ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Özden ERÇİN

MULTI-OBJECTIVE BEE COLONY OPTIMIZATION TO TUNING PID CONTROLLER

DEPARTMENT OF COMPUTER ENGINEERING

ADANA, 2011

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MULTI-OBJECTIVE BEE COLONY OPTIMIZATION TO TUNING PID CONTROLLER

Özden ERÇİN

MSc THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Master of Science by the board of jury on .../..../......

This MSc Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.

Registration Number:

Prof. Dr. İlhami YEĞİNGİL
Director
Institute of Natural and Applied Sciences

ABSTRACT

MSc THESIS

MULTI-OBJECTIVE BEE COLONY OPTIMIZATION TO TUNING PID CONTROLLER

Özden ERÇİN

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

Supervisor :Asst.Prof.Dr. Ramazan ÇOBAN
Yıl: 2011, Sayfa: 91
Jury :Asst.Prof.Dr. Ramazan ÇOBAN
:Assoc.Prof.Dr. İlyas EKER
:Assoc.Prof.Dr. Zekeriya TÜFEKÇİ

In this study, a novel intelligent design method for closed-loop auto-tuning of a proportional-integral-derivative (PID) controller based on Bee Inspired Swarm Algorithms is proposed, in which PID controller parameters can be tuned concurrently. The set of trade-off optimal solutions, called Pareto-set optimization solutions, of the conflicting objective functions are able to be found. Moreover the research presents an investigation for the development of system identification using Artificial Bee Colony (ABC).

Bees Algorithm (BA), developed by D.T. Pham (2006) and Artificial Bee Colony, developed by Dervis Karaboga (2005) are a subfield of Swarm Intelligence and was inspired by swarming patterns occurring in nature such as the food foraging behavior of honeybees. In the present study, the problem of identifying the PID controller parameters is considered as an optimization problem. The Multi-Objective Bees Algorithm (MOBA) and the Multi-Objective Artificial Bee Colony (MOABC) algorithm have been employed to determine the PID parameters. The PID controller is designed using the MOBA and the MOABC algorithms. The results of all designs are compared and analyzed. Simulation results demonstrate that the proposed method using the MOBA and the MOABC has a better control system performance. The results obtained show good stability, set-point tracking performance and robustness against disturbance.

Key Words: Multi-objective optimization, Artificial Bee Colony, Bees Algorithm, PID tuning, System Identification

ÖZ

YÜKSEK LİSANS TEZİ

ÇOK AMAÇLI ARI KOLONİSİ OPTİMİZASYONU KULLANARAK PID KONTROLÖRÜN AYARLANMASI

Özden ERÇİN

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman :Yrd.Doç.Dr. Ramazan ÇOBAN
Yıl: 2011, Sayfa: 91
Jüri :Yrd.Doç.Dr. Ramazan ÇOBAN
:Doç.Dr. İlyas EKER
:Doç.Dr. Zekeriya TÜFEKÇİ

Bu çalışmada, PID kontrolörün parametrelerinin ayarlanması için, arı algoritmalarına dayalı bir method önerilmiştir. D.T. Pham (2006) tarafından geliştirilen arı algoritması ve Derviş Karaboğa (2005) tarafından geliştirilen yapay arı kolonisi, bal arılarının yiyecek arama davranışları gibi doğada oluşan sürü modellerinden esinlenmiş algoritmalarıdır. Bu çalışmada, PID kontrolör parametrelerinin belirlenmesi problemi bir optimizasyon problemi olarak kabul edilmektedir. Çok-amaçlı arı algoritması (MOBA) ve çok-amaçlı yapay arı kolonisi (MOABC) algoritmaları PID kontrolör parametrelerini belirlemek için kullanılmıştır. Farklı derecelerdeki sistemler için sonuçlar karşılaştırılmıştır ve analiz edilmiştir. Simülasyon sonuçları çok-amaçlı arı algoritması ve çok-amaçlı yapay arı kolonisi yöntemleri kullanılarak, parametre ayarlama problemlerinde daha iyi performas özelliklerine sahip olduğunu göstermektedir. Elde edilen sonuçlar, gürültüye karşı iyi bir kararlılık ve dayanıklılık göstermiştir.

Ayrıca, bu çalışmada yapay arı kolonisinin (ABC), sistem tanılama problemlerinde başarı ölçütleri incelenmiştir. Elde edilen sonuçlar, sistem tanıma problemlerinde arı algoritmasının başarıyla kullanabileceğini göstermiştir.

Anahtar Kelimeler: Çok Amaçlı Optimizasyon, Yapay Arı Kolonisi, Arı Algoritması, PID Parametrelerinin Ayarlaması, Sistem Tanılama

ACKNOWLEDGEMENTS

The work presented in this thesis was carried out under the supervision of Asst.Prof.Dr. Ramazan ÇOBAN. My gratitude and sincere appreciation goes out to him for his assistance, valuable contribution and guidance throughout the study. His challenging questions and imaginative input greatly benefited the work. His leadership and motivation have been a source of inspiration and a model for me to follow.

I am very grateful to Assoc.Prof.Dr. İlyas EKER for his endless support.

I would like to thank and acknowledge the support from Cukurova University Library for my research project.

I owe my deepest thanks to Çukurova University Computer Engineering Department that enabled me to benefit form all facilities of the department during my thesis study.

A special note of thanks to my parents for their unwavering support and guidance. Finally, thanks to my wife Saime for her tolerance and patience while I spent many hours glued in front of the PC. She has shown great understanding and given selfless support. She is always supported and encouraged me, I am very grateful to her all times. This thesis is dedicated to my newborn daughter Beren with all my heart.

CONTENTS **PAGE**

ABSTRACT I
ÖZ II
ACKNOWLEDGEMENTS III
CONTENTS IV
LIST OF TABLES VI
LIST OF FIGURES VIII
LIST OF ABBREVIATONS XII
LIST OF SYMBOLS XIV
l. INTRODUCTION 1
2. HONEY BEE ALGORITHMS 9
2.1. Artificial Bee Colony Algorithm 11
2.2. Bees Algorithm 16
2.3. Comparison of Honey Bee Algorithms for Numerical Function Optimization 19
3. SYSTEM IDENTIFICATION 21
3.1. Linear System Identification 22
3.2. Applying Artificial Bee Colony Algorithm in System Identification 23
4. PID CONTROLLER DESIGN 27
4.1. Controller Design Methods 28
4.1.1. Iterative or Manual Tuning Method 28
4.1.2. Ziegler-Nichols Frequency Domain Method 29
4.1.3. Ziegler-Nichols Time Domain Method 29
4.1.4. Cohen-Coon Method 30
4.1.5. Root Locus Method 31
4.2. PID Characteristic Parameters 31
4.3. Principle of Multi-objective Optimization (Pareto Optimality) 36
4.3.1. Weighted Sum Method 38
4.3.2. Applying Artificial Bee Colony and Bees Algorithm in PID Controller Design....... 39

5. DISCUSSION OF SIMULATION RESULTS 41
5.1. Results for System Identification.. 41
5.1.1. Example 1: First Order SISO Linear System Identification 41
5.1.2. Example 2: Third Order SISO Linear System Identification........ 44
5.1.3. Example 3: Fifth Order SISO Linear System Identification......... 47
5.1.4. Example 4: Seventh Order SISO Linear System Identification 50
5.1.5. Example 5: DC Motor Identification 54
5.2. Results for PID Controller Tuning... 58
5.2.1. Example 1: PID Controller Tuning for Third Order Linear Plant . 59
5.2.2. Example 2: PID Controller Tuning for Fourth Order Linear Plant 65
5.2.3. Example 3: PID Controller Tuning for DC Motor Plant 71
5.2.4. PID Controller Design with Gaussian White Noise 75
6. CONCLUSION.. 83
6.1. Conclusion.. 83
6.2. Future Work... 84
REFERENCES 85
CURRICULUM VITAE 91

LIST OF TABLES PAGE

Table 2.1. Mean and standard deviations of the results obtained by the ABC and the BA algorithms for Rosenbrock function 20

Table 4.1. PID characteristic parameters 34

Table 5.1. Simulation results of the estimated $G_1(s)$ linear plant......................... 42

Table 5.2. RMS errors for $G_1(s)$ linear plant.. 44

Table 5.3. Simulation results of the estimated $G_2(s)$ linear plant......................... 45

Table 5.4. RMS errors for $G_2(s)$ linear plant.. 46

Table 5.5. Simulation results of the estimated $G_3(s)$ linear plant......................... 48

Table 5.6. RMS errors for $G_3(s)$ linear plant.. 49

Table 5.7. Simulation results of the estimated $G_4(s)$ linear plant......................... 52

Table 5.8. RMS errors for $G_4(s)$ linear plant.. 53

Table 5.9. Parameters of the Motor (Ong, 1998) 55

Table 5.10. The real and estimated parameters of the DC motor......................... 57

Table 5.11. RMS errors for DC Motor 57

Table 5.12. Comparison of simulation results of the PID controlled systems (Hsiao et al., 2004) 60

Table 5.13. Simulation results of the proposed algorithms for different error criteria 60

Table 5.14. Simulation results of the proposed algorithm with increasing rise time 65

Table 5.15. Comparison of simulation results of the PID controlled systems (Hsiao et al., 2004) 66

Table 5.16. Simulation results of the proposed algorithms for different error criteria 66

Table 5.17. Simulation results of the proposed algorithm with increasing rise time 71

Table 5.18. Simulation results of Motor Speed PID control (Nasri et al., 2007).. 73

LIST OF FIGURES **PAGE**

Figure 2.1. Pseudo code of the bee algorithms 11
Figure 2.2. Detailed flow chart of the artificial bee colony algorithm 15
Figure 2.3. Detailed flow chart of the bees algorithm 19
Figure 4.1. Plant step response in Z-N method 30
Figure 4.2. Block diagram of a PID controller 32
Figure 4.3. Rise time, Settling time, Maximum overshoot 36
Figure 4.4. Illustration of Pareto front for a bi-objective optimization problem .. 38
Figure 4.5. Block diagram of a MOABC-PID controller 39
Figure 4.6. Block diagram of a MOBA-PID controller 40
Figure 5.1. Step responses of the plant $G_1(s)$ 43
Figure 5.2. Sinusoidal responses of the plant $G_1(s)$ 43
Figure 5.3. Step responses of the plant $G_2(s)$ 46
Figure 5.4. Sinusoidal responses of the plant $G_2(s)$ 46
Figure 5.5. Step responses of the plant $G_3(s)$ 49
Figure 5.6. Sinusoidal responses of the plant $G_3(s)$ 49
Figure 5.7. Step responses of the plant $G_4(s)$ 53
Figure 5.8. Sinusoidal responses of the plant $G_4(s)$ 53
Figure 5.9. A DC Motor equivalent circuit 55
Figure 5.10. Step response of the DC motor 57
Figure 5.11. Sinusoidal response of the DC motor 57
Figure 5.12. Comparison of step responses of the plant $G_1(s)$ for the MOBA 61
Figure 5.13. Comparison of step responses of the plant $G_1(s)$ for the MOABC... 61
Figure 5.14. Multi-objective optimization Pareto-sets of the plant $G_1(s)$
for the MOBA 62
Figure 5.15. Multi-objective optimization Pareto-sets of the plant $G_1(s)$
for the MOABC 62
Figure 5.16. Convergence graph of the plant $G_1(s)$ in the MOBA method 63
Figure 5.17. Convergence graph of the plant $G_1(s)$ in the MOABC method 63

Figure 5.18. Step responses of the plant G1(s) with increasing rise time for the MOBA 64
Figure 5.19. Step responses of the plant $G_1(s)$ with increasing rise time for the MOABC........ 64
Figure 5.20. Comparison of step responses of the plant $G_2(s)$ for the MOBA 67
Figure 5.21. Comparison of step responses of the plant $G_2(s)$ for the MOABC... 67
Figure 5.22. Multi-objective optimization Pareto-sets of the plant $G_2(s)$ for the MOBA 68
Figure 5.23. Multi-objective optimization Pareto-sets of the plant $G_2(s)$ for the MOABC........ 68
Figure 5.24. Convergence graph of the plant $G_2(s)$ in the MOBA method 69
Figure 5.25. Convergence graph of the plant $G_2(s)$ in the MOABC method........ 69
Figure 5.26. Step responses of the plant $G_2(s)$ with increasing rise time for the MOBA 70
Figure 5.27. Step responses of the plant $G_2(s)$ with increasing rise time for the MOABC........ 70
Figure 5.28. Comparison of step responses of the DC motor for the MOBA 72
Figure 5.29. Comparison of step responses of the DC motor for the MOABC.... 72
Figure 5.30. Multi-objective optimization Pareto-sets of the DC motor for the MOBA 73
Figure 5.31. Multi-objective optimization Pareto-sets of the DC motor for the MOABC........ 74
Figure 5.32. Convergence graph of the DC motor by using the MOBA method . 74
Figure 5.33. Convergence graph of the DC motor by using the MOABC method75
Figure 5.34. The PID controller in the closed-loop with Gaussian White Noise.. 75
Figure 5.35. Comparison of Gaussian white noise step responses of the plant $G_1(s)$ for the MOBA for σ^2=0.0025........ 76
Figure 5.36. Comparison of Gaussian white noise step responses of the plant $G_1(s)$ for the MOBA for σ^2=0.025 76
Figure 5.37. Comparison of Gaussian white noise step responses of the plant $G_1(s)$ for the MOABC for σ^2=0.0025........ 77

Figure 5.38. Comparison of Gaussian white noise step responses of the plant $G_1(s)$ for the MOABC for σ^2=0.025 77
Figure 5.39. Comparison of Gaussian white noise step responses of the plant $G_2(s)$ for the MOBA for σ^2=0.0025 78
Figure 5.40. Comparison of Gaussian white noise step responses of the plant $G_2(s)$ for the MOBA for σ^2=0.025 78
Figure 5.41. Comparison of Gaussian white noise step responses of the plant $G_2(s)$ for the MOABC for σ^2=0.0025 79
Figure 5.42. Comparison of Gaussian white noise step responses of the plant $G_2(s)$ for the MOABC for σ^2=0.025 79
Figure 5.43. Comparison of Gaussian white noise step responses of the plant $G_3(s)$ for the MOBA for σ^2=0.0025 80
Figure 5.44. Comparison of Gaussian white noise step responses of the plant $G_3(s)$ for the MOBA for σ^2=0.025 80
Figure 5.45. Comparison of Gaussian white noise step responses of the plant $G_3(s)$ for the MOABC for σ^2=0.0025 81
Figure 5.46. Comparison of Gaussian white noise step responses of the plant $G_3(s)$ for the MOABC for σ^2=0.025 81

LIST OF ABBREVIATONS

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
BA	Bees Algorithm
DC	Direct Current
DCS	Distributed Control System
HBA	Honey Bee Algorithm
HBMO	Honey-Bee Mating Optimization
GA	Genetic Algorithm
IAE	Integral Absolute Error
ISE	Integral Square Error
ITAE	Integral Time Absolute Error
ITSE	Integral Time Square Error
LTI	Linear Time-Invariant
PD	Proportional, Derivative
PI	Proportional, Integral
PID	Proportional, Integral and Derivative
MSE	Mean Square Error
MOBA	Multi-Objective Bees Algorithm
MOABC	Multi-Objective Artificial Bee Colony
SISO	Single Input Single Output
VBA	Virtual Bee Algorithm
ZN	Ziegler-Nichols

LIST OF SYMBOLS

r	The reference input signal
e	The error signal
u	The control signal
y	The output signal
a	Abandonment threshold
y_{max}	The maximum value of y
y_{ss}	The steady-state value of y
t_r	Rise time
t_s	Settling time
M_p	Maximum overshoot
J^B	The value of the objective function
D	Dimension, Number of parameters
Φ	Vector of non negative weights
G_p	A Linear Time-Invariant plant’s transfer function
G_c	PID Controller transfer function
K_p	Proportional gain constant of PID controller
K_d	Derivative gain constant of PID controller
K_i	Integral gain constant of PID controller
n	Number of scout bees
m	Number of sites selected for neighbourhood search
e_s	Number of top-rated (elite) sites among m selected sites
nep	Number of bees recruited for the selected sites
nsp	Number of bees recruited for the other $(m-e_s)$ selected sites
ngh	The initial size of each patch
sc	Shrinking constant
C_m	Motor torque
I_a	Rotor circuit current
K_e	Electrical constant
K_m	Mechanical constant

L_a	Rotor circuit inductance
R_a	Rotor circuit resistance
U_a	Input voltage
B	Damping ratio
e_v	Electromotive voltage
J	Rotor moment of inertia
Ω	Rotor speed
x_i	Solution of i
p_i	Probability values
fit_i	Fitness value of the solution i
SN	Number of solutions
Φ_{ij}	Random number
MCN	Maximum cycle number
$U(k)$	External input
$\hat{Y}(k)$	Output of the model
$Y(k)$	Output of the system
$e(k)$	Error signal
$\alpha_0 \dots \alpha_n$	Numerator system parameters
$\beta_0 \dots \beta_n$	Denumerator system parameters
J_e	The measure of the error
K_{CR}	The critical gain value
P_{CR}	Ultimate period
T_d	Dead time
K	Process gain
$\lambda_1 \dots \lambda_N$	Weighting coefficient
$f_1 \dots f_N$	The objective functions
NP	Onlooker bees
$iter$	Iteration number
σ^2	Variance of gaussian white noise
d	Disturbace

1. INTRODUCTION

After the Industrial Revolution, when people have kept on inventing all kinds of new machines to relieve his daily work, the importance of ways to control these technical processes increased. First, people were the overall controllers of the processes in for example a factory, but with the increasing complexity and increasing pays this was no longer possible. People started a search for technical expedient resources to ease their controlling job. During the mechanization period, development of ways was mainly involved to measure things in processes. People could use this information for better control of the process. When the Automation started, people gave away a part of his most easy controlling tasks to technical expedient resources, which were often electrical circuits. Nowadays processes are controlled by electronics and computers at much higher levels and this development still goes on. Some information is required about the process in order to develop controllers for a process. That is to say, the system has to be identified and has to be put in some kind of model. The kind of model relies on the kind of controller which one wants to implement. Process analysis is another application of models: try to predict the behaviour of a process in certain circumstances.

Using the physical equations which belong to the components of the process is one way to construct a model. After determining the equations, the system can be represented by several interconnected boxes with equations in it. The transfer is described by these equations from an input of the box to the output. This model is called “White Box Modelling” (Eykhoff, 1974). It does not mean that this model never describes the system exactly. The physical equations describe a simplified situation and only hold under certain conditions and assumptions. White Box Modelling is very time-consuming or even impossible for more complex systems. During the last few decades a new way of system identification called as “Black Box Identification” has been developed (Ljung, 1999). The different inputs of the process are excited with some kind of (time) signal and the outputs are measured. These measured data are put into a computer algorithm that calculates a certain type of model. There is no direct relation between the resulting model and the physical reality, therefore the name “Black Box Modelling”. Also these models do not entirely describe a process and have limited validity. For instance although nearly all processes are nonlinear and of high order, the resulting models are usually linear and of a finite (small) order. Since a system is only identified for certain frequencies or range of frequencies, the model is only valid for or near these working points. During measurement and identification all kinds of noise appear which deteriorate the identification appear, such as noise in the sensors, quantization errors in the computer algorithms etc.

The primary role of system identification in control system design is reduction in uncertainty. This can be performed by using more accurate modelling techniques (such as non-linear models), or by determination of the quality of the limited model. To develop controllers, especially modern / robust controllers, this information can be used. So by underlying modern robust control design techniques (control oriented system identification methods) the framework and assumptions are matched by this idea. Robust controllers have a better performance than more classical controllers by making explicit use of prior information about the model error. It is useful if, in addition to verification of the model with the true system, mathematical methods for the determination of model errors and uncertainties are available.

Black Box identification has two approaches that can be distinguished. In the “traditional” approach the system can be described by one single model, so the system is in the set of models which is considered. Although the noise is not fixed, it is described as a stochastic quantity. Therefore, this is called a stochastic approach of system identification (Van den Bosch and Van der Klauw, 1994). All other identification methods in which the stochastic setting is not used are called deterministic. The resulting models are generally non-parametric and often of high order. Most recent developments in system identification methods focus on deterministic approaches which are strongly related to robust control in the way that these algorithms yield models with an upperbound for the model error.

A mathematical model of the plant must be obtained in the initial phase of design process. Using a numerical process known as system identification is one way to obtain a model. This process involves acquiring data from a plant and then numerically analyzing stimulus and response data to estimate the parameters of the plant.

These models can be used for control system design, fault detection or adaptive guidance (Ogata, 1995). In turn system identification parameter estimation is a common criterion for control system, especially for sensitive or adaptive control system design. In fact, if the estimated parameters used in the system model for controller design do not coincide with the actual process parameters, a closed loop control system may be unstable or exhibit unacceptable transient response characteristics. So, parameters estimation technique that is accurate and reliable is critical for the design and development of high-performance control systems in which the estimated parameters are often used in the self-sensing, field orientation, motion control and other advanced algorithms.

The conventional system identification schemes are actually local search techniques. If the search space is not differentiable or linear in the parameters, these techniques do not often achieve in the search for the global optimum. On the other hand, these techniques can iterate only once on each datum received. Using artificial intelligence algorithm, a better solution could be provided as an alternative strategies. To achieve this aim most commonly used intelligence algorithms are used to point out the capabilities.

Over the past decade, honey bees algorithms and their other variants have been a topic of research. Inspired by the foraging behavior of honeybees, honey bees algorithms have proven to be an effective and useful stochastic search technique (Krusienski and Jenkins, 2005). Therefore, it has been applied to a wide variety of problems related to search optimization, routing, clustering, scheduling. Honey bees algorithms have gone through various changes and different variants have been introduced to solve the problems more effectively. It has also been combined with other different artificial intelligence algorithms to create hybrid optimization algorithms (Moore and Venayagamoorthy, 2006). These algorithms have been published in different literature and applied to different practical applications. In this thesis, two problems have been studied: (1) system identification and (2) PID controller design.

The conventional PID controllers are extensively used in industry because of their simple operating algorithm, ease of using, good robustness, high reliability, stabilization and zero steady state error (Lu et al., 2007). It is widespread in use and universally accepted because of its simple operating algorithm, the relative ease with which the controller effects can be adjusted, the broad range of applications where it has reliably produced excellent control performances, and the familiarity with which it is perceived amongst researchers and practitioners within the process control community. Industrial PID control schemes based on the classical control theory have been widely used for miscellaneous process control systems for many years. They have been preferred for their functional simplicity, good robust performance and easy implementation in a wide range of operating conditions; furthermore, PID controller principle is easier to understand than other traditional controllers for the majority of industrial processes. In PID tuning rules, there are a number of objectives such as stability, regulating performance, tracking performance, robustness and noise attenuation. It does not mean that objectives are consistent or commensurable. However, since the performance of a PID controller completely depends on the tuning of its parameters many industrial plants are often confronted with many problems such as higher order, time delays and nonlinearities (Kwok et al., 1993).

Designing and tuning a Proportional Integral Derivative (PID) controller seems to be conceptually intuitive. However, if there are multiple (and often conflicting) objectives which have to be achieved such as transient behaviour and high stability have to be achieved, it can be hard to tune in practice. Generally, initial designs obtained by all means need to be adjusted repeatedly through computer simulations until the closed-loop system performs or compromises as desired. The development of intelligent software tools provide that engineers can achieve the best overall PID control for the entire operating envelope. Since the invention of PID, numerous tuning rules which differ in flexibility, complexity and amount of process knowledge have been developed. During the development of the tuning rule, an important point that seems to miss out is in practical process control industrial environments. It is obvious that there is a need to achieve satisfactory control performances without adopting complex control architectures, to guarantee the best cost/benefit ratio.

Many tuning rules accepted by industry are now incorporated into the hardware modules. However, due to the fact that modelling errors, process variations and human errors exist, user can intervent in tuning of PID controllers. So, a tuning rule which is simple to understand and quick to apply is needed by user. Needless to say, it cannot be assumed that all users are highly educated in control theory. That is why the classical Ziegler-Nichols tuning rule (Ziegler and Nichols, 1942) is still commonly used.

There have been several tuning methods proposed for the tuning of process control loops, with the most popular method being that of Ziegler and Nichols (1942). Other methods are the methods of Cohen and Coon (1953), Åström and Hägglund (1984), De Paor and O'Malley (1989), Zhuang and Atherton (1993), Venkatashankar and Chidambaram (1994), Poulin and Pomerleau (1996) and Haung and Chen (1996). Despite this large range of tuning techniques, to date there still seems to be no general consensus as to which tuning method works best for most applications (Lipták, 1995). Some methods are based heavily on experience, while others are based more on mathematical considerations.

Process control practitioners mostly preferred the Ziegler-Nichols method. Since control personnel are reluctant to learn new techniques which they perceive as being complicated, time consuming and laborious to implement, alternative methods are often not applied in practice. Also, there is no sufficient performance of some commonly used techniques in the presence of strong nonlinear characteristics within the control channel (Åström and Hägglund, 2004).

In many papers, different PID control methods have been applied to determine three parameters of PID controller for the given processes (Bagis, 2007). Several algorithms such as manual tuning, Ziegler-Nichols, Cohen-Coon, etc. have their own advantages and disadvantages. The major drawback of the manual tuning method is that it requires experienced personnel. Some shortcomings of the Ziegler-

Nichols method are the resulting in large overshoot and oscillatory responses. Besides, controller settings necessitate very aggressive tuning and also further fine tuning. This method has also poor performance for processes with a dominant delay and closed loop system is very sensitive to parameter variations, so parameters of the step response may be hard to determine due to measurement noise. A common disadvantage of the Cohen-Coon method is that it can only be used for first order models including large process delays. It is almost not possible to achieve optimal performance by using classical search and optimisation methods.

In the last decades, design engineers have focused on evolutionary based approaches to improve the existing design theories and find the best design results to tune the parameters of PID controllers. The main weakness of Genetic Algorithm (GA) among evolutionary based approaches is a lack of guarantee that global optimum is found within limited period of time and slower speed of convergence (Bagis, 2007). A disadvantage of Ant Colony Optimization (ACO) which is another evolutionary based approach is difficulty of theoretical analysis, sequences of random decisions and probability distribution changes by iteration (Hsiao et al., 2004). In addition, a long convergence time is a significant drawback of it but convergence is guaranteed. The properly selection of the PID parameters is so important that the closed loop system must meets design specifications. The design specifications can include minimum or no overshoot, minimal rise time, minimal steady state error and settling time in the step response of the closed loop system. Multi-Objective Bees Algorithm (MOBA) has been used successfully to solve many problems and applied to constrained and unconstrained single objective function optimizations (Pham and Ghanbarzadeh, 2007). In this work, the MOBA was applied to optimize the parameters of PID controllers. To indicate the effectiveness and efficiency of the proposed optimization method, the step responses of closed loop systems were compared with those of the existing methods in the literature such as Ziegler–Nichols, genetic algorithm and ant colony optimization.

This study proposes the development of a tuning technique that would be suitable for optimizing the control of processes operating in a single input single output (SISO) process control loop. The SISO topology has been selected for this study because it is the most fundamental of control loops and the theory developed for this type of loop can be easily extended to more complex loops.

The thesis has been divided into 6 chapters. Chapter 1 introduces to the topic and outlines the objectives of the research work carried out. In Chapter 2, honey bee algorithms have been covered. This chapter explains the basics of the algorithm and how it has been applied to the numerical function optimization problem. In Chapter 3, system identification has been explained. This chapter introduces to the problem of system identification and traditional and modern techniques used to solve it. In Chapter 4, PID controller design is explained. This chapter introduces to the problem and traditional and modern techniques used in PID controller design.

In the next two chapters, case studies carried out during the research and the results obtained from them have been presented. In Chapter 5, studies and results of system identification have been presented. This chapter shows the comparison of results obtained from system identification, and is presented as figures and tabulated data. Additionally, results obtained for PID controller design are presented. These results are also presented as figures and tabulated data and show a comparison of different algorithms as applied to the examples.

Conclusion of the thesis and future work is presented in Chapter 6.

2. HONEY BEE ALGORITHMS

Instinctive ability that colonies of social insects such as ants and bees have is called as swarm intelligence (Nakrani and Tovey, 2004). By functioning collectively and interacting primitively amongst members of the group, the colonies of insects are able to solve problems beyond capability of individual members by means of this highly organized behavior. In a honey bee colony, for example, environment in search of flower patches (food sources) is explored honey bees and then the food source is indicated to the other bees of the colony when they return to the hive. Such a colony is described by self-organization, adaptiveness and robustness (Teodorovic and Dell'orco, 2005).

A behavioral model of self-organization is proposed for a colony of honey bees by Seeley (1995). Foraging bees visiting flower patches return to the hive with nectar as well as a profitability rating of respective patches in the behavioral model. The nectar collected by foraging bees provides feedback on the current status of nectar flow into the hive (Camazine and Sneyd, 1991). The profitability rating is a function of nectar bounty, nectar quality and distance from the hive. A response threshold is set by the feedback for an enlisting signal which is known as waggle dance, the length of which is dependent on both the profitability rating and the response threshold. On the dance floor where can be observed by individual foragers the waggle dance is performed. The forager bees can randomly select a dance to observe and follow from which they can learn the location of the flower patch and leave the hive to forage.

Honey bees live in a colony in the nature and they forage and store honey in their constructed colony. Pheromone and "waggle dance" are communication of honey bees (Bastürk and Karaboga, 2006). For example, a chemical message (pheromone) may be released by an alarming bee to stimulate attack response in other bees. Furthermore, bees will communicate the location of the food source by performing the so called waggle dances as a signal system when they find a good food source and bring some nectar back to the hive. Such signaling dances differ from species to species, however, using directional dancing with varying strength, more bees will be recruited so as to communicate the direction and distance of the found food resource.

For multiple food sources such as flower patches, in studies it is shown that forager bees are able to be allocated among different flower patches so as to maximize their total nectar intake. A bee colony typically has to collect and store extra nectar, about 15 to 50 kg in order to survive the winter. From the evolution point of view, the efficiency of nectar collection is consequently very important. Experimental studies which include the important work have also been carried out by Camazine and Sneyd (1991) and lately by Quijano and Passino (2007a). If the natural behaviour of bee colonies is learnt, various algorithms can be designed (Quijano and Passino, 2007a, 2007b).

Over the last decade or so, nature-inspired bee algorithms have started to emerge as a promising and powerful optimization tool.The exact dates are difficult to pinpoint when the bee algorithms were first formulated. Several groups of researchers developed them over a few years independently.

From the literature survey, in around 2004 Nakrani and Tovey (2004) first formulated the Honey Bee Algorithm (HBA) to study a method to allocate computers among different clients and web-hosting servers. Later in 2004 and earlier 2005, by Yang (2005) a Virtual Bee Algorithm (VBA) was developed to solve numerical optimization problems. Although only functions with two parameters were given as examples, VBA can optimize both functions and discrete problems. Slightly later in 2005, a Honey-Bee Mating Optimization (HBMO) algorithm which was subsequently applied to reservoir modelling and clustering was presented by Afshar et al. (2007). Around the same time, an Artificial Bee Colony (ABC) algorithm was developed by Karaboga (2007) in Turkey for numerical function optimization and a comparison study was carried in 2007. These bee algorithms are nowadays becoming more and more popular (Kang et al., 2011).

The communication or broadcasting ability of a bee to some neighbourhood bees shows the essence of the bee algorithms so a bee can be known and followed to the best source, locations or routes to complete the optimization task. The detailed implementation will depend on the actual algorithms, and they may differ slightly and vary with different variants. However, the pseudo code in Figure 2.1 can summarize the essence of all the bee algorithms.

```
Objective function $f(x)$, $x=(x_1, .... x_n)^T$ & constraints
Encode $f(x)$ into virtual nectar levels
Define dance routine (strength, direction) or protocol
while (criterion)
    for loop over all $n$ dimensions
    Generate new solutions
    Evaluate the new solutions
    end for
    Communicate and update the optimal solution set
end while
Decode and output the best results
```

Figure 2.1. Pseudo code of the bee algorithms

2.1 Artificial Bee Colony Algorithm

A minimal model of foraging behavior of a honeybee colony was developed based on the reaction-diffusion equations by Tereshko (2000). This model provides the emergence of collective intelligence of honey bee swarms. There are three essential components of this model: food sources, employed foragers and unemployed foragers and two leading modes of the honey bee colony behavior are defined by this model: recruitment to a food source and abandonment of a source. A forager bee evaluates a number of properties related with the food source such as its taste of its nectar, closeness to the hive, richness of the energy and the ease or difficulty of extracting this energy in order to choose a food source. An employed forager is employed at a specific food source in order to carry information about this specific source and share it with other bees waiting in the hive. The information contains the distance between hive and food source, the direction and the profitability of the food source. Unemployed forager is a forager bee looking for a food source to exploit. It can be a scout who looks for the environment arbitrarily or an onlooker who attempts to find a food source through the information given by employed bee

(Tereshko, 2000). The main steps of the ABC algorithm are given below (Basturk and Karaboga, 2009a):

Step 1. Generate the population of solutions (positions of food sources) randomly x_i, $i = 1. . .SN$

Step 2. Evaluate the generated population

Step 3. Cycle = 1

Step 4. Repeat

Step 5. Produce new solutions υ_i for the employed bees by using Equation (2.2) and evaluate them

Step 6. Apply the greedy selection process

Step 7. Calculate the probability values p_i for the solutions x_i by Equation (2.1)

Step 8. Produce the new solutions υ_i for the onlookers from the solutions x_i selected depending on p_i and evaluates them

Step 9. Apply the greedy selection process

Step 10. Determine the abandoned solution for the scout and replace it with a new randomly produced solution x_i by Equation (2.3).

Step 11. Record the best solution achieved so far

Step 12. Cycle = Cycle + 1

Step 13. Until Cycle = Maximum cycle number

In the ABC algorithm, there are three flocks of bees: onlooker, employed, and scout. A colony consists of the onlooker bees plus employed bees. If an employed bee abandons its food source, it becomes a scout bee. The number of solutions (population) to problem is equal to the number of the onlooker bees or the employed bees. A possible solution to the optimization problem is presented by the position of a food source and the quality (fitness) is measured with the amount of nectar of the associated food source. The number of food source equals the number of employed bees. At the first step, initial population $P(G = 0)$ of SN solutions (food source positions) is generated randomly by the ABC. SN denotes the size of population. Each solution x_i ($i = 1, 2, ..., SN$) is presented by using a D-dimensional vector. Here,

D denotes the number of optimization parameters. After initialization, the employed bees, the onlooker bees and the scout bees repeatedly search for all food sources until a predetermined number of iterations (*Cycle=1, 2, ..., MCN*). An employed bee starts neighborhood search firstly depending on the local information (visual information) and tests the nectar amount (fitness value) of the new source (new solution), the position of new sources replace the previous one if better than previous position, otherwise keep the position of previous one. After the search process is completed by all employed bees, the nectar information of the food sources and their position information are shared with the onlooker bees on the dance area. An onlooker bee evaluates the nectar information taken from all employed bees and then it chooses a food source by using a selection probability related to its nectar amount.

An artificial onlooker bee selects a food source depending on the selection probability value associated with that food source, p_i, calculated by the following expression (Basturk and Karaboga, 2009a, 2009b):

$$p_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \tag{2.1}$$

where fit_i is the fitness value of the ith solution which is proportional to the nectar amount of the food source in the position i. SN is the number of solutions which is equal to the number of employed bees (BN).

So as to produce an applicant food position from the old one in memory, the ABC uses the following expression (Basturk and Karaboga, 2009a, 2009b):

$$v_{ij} = x_{ij} + \Phi_{ij}(x_{ij} - x_{kj}) \tag{2.2}$$

where $k \in \{1, 2, ..., SN\}$ and $j \in \{1, 2, ..., D\}$ are randomly selected indexes. The index k is determined randomly but it has to be different from i. Φ_{ij} is a random number and chosen between [-1, 1]. A bee compares two food locations visually by using this parameter. As seen from Equation (2.2), as long as the change between the positions of x_{ij} and x_{kj} diminishes, the perturbation on the position x_{ij} decreases. Therefore, the search comes close to optimum solution in the search space.

The scout bees replace new food sources, which are produced randomly in their dynamic ranges, with the ones which the employed bees abandon. In ABC, during a predetermined number of cycles if a position cannot be improved further then that food source is thought to be abandoned. Here, since the only one source is abandoned in each cycle, one employed bee becomes a scout bee. The parameter so-called *"limit"* is an important one for abandonment which is the value of predetermined number of cycles [3]. If the abandoned source is x_i and $j \in \{1, 2,..., D\}$, then a new food source is discovered by the scout to be replaced with x_i. The description of this operation is given as (Basturk and Karaboga, 2009a, 2009b)

$$x_i^j = x_{\min}^j + rand(0,1)(x_{\max}^j - x_{\min}^j) \tag{2.3}$$

After the artificial bee produces and then evaluates each candidate source position v_{ij}, its performance is compared with that of its old one. If the nectar of the new food source is equal to or better than the old one, it replaces with the old one in the memory. Otherwise, the old one in the memory is retained. In other words, a greedy selection mechanism is engaged as the selection operation between the old and the candidate one. In the ABC, there are three control parameters such as the number of food sources which is equal to the number of employed or onlooker bees (*SN*), the value of limit, the maximum cycle number (*MCN*). In a robust search process, exploration and exploitation processes have to be performed together. In the ABC algorithm, the scout bees control the exploration process while onlookers and employed bees carry out the exploitation process in the search space. The detailed flow chart of the artificial bee colony algorithm is shown in Figure 2.2.

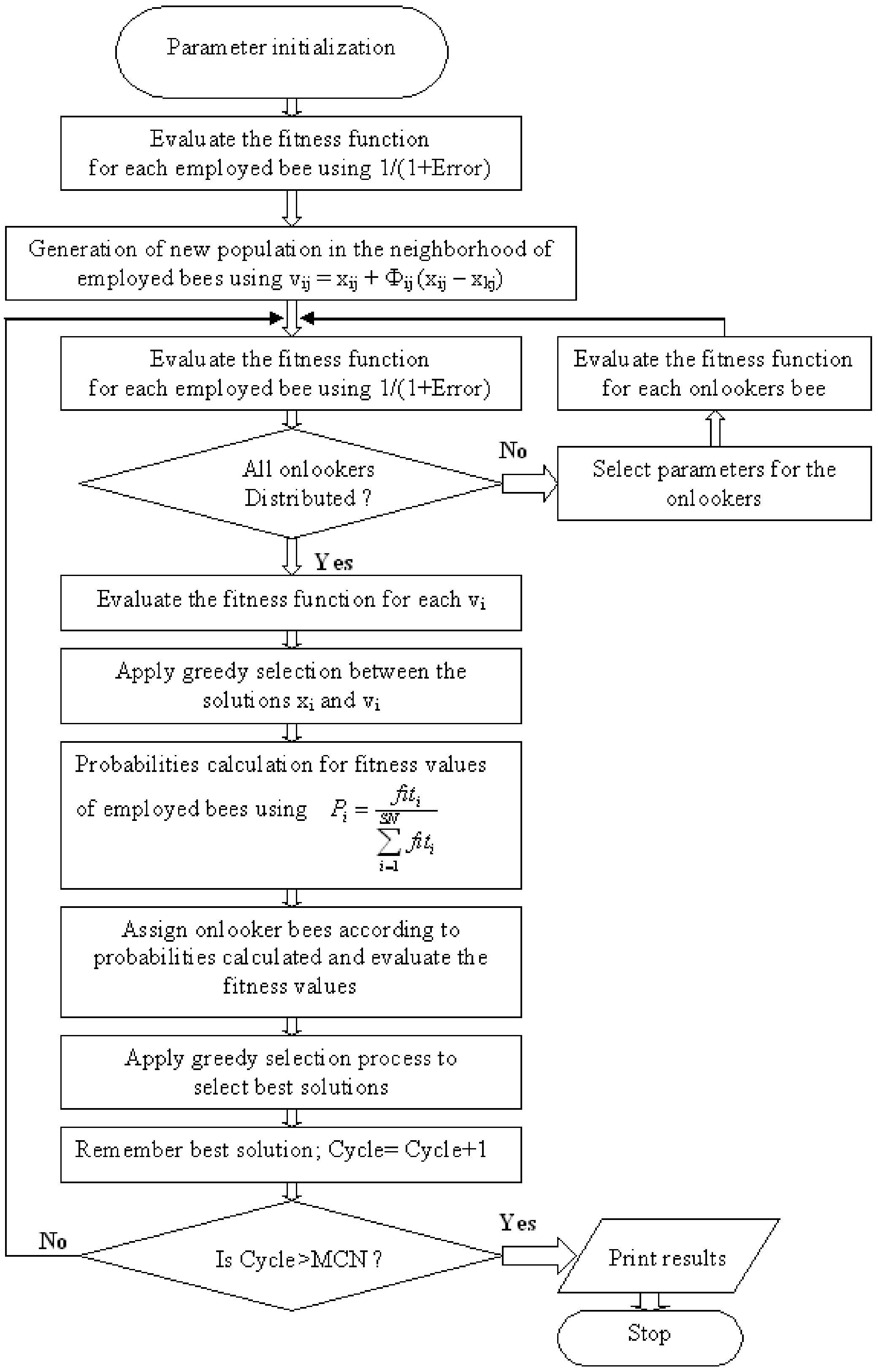
Parameter initialization
Evaluate the fitness function for each employed bee using 1/(1+Error)
Generation of new population in the neighborhood of employed bees using $v_{ij} = x_{ij} + \Phi_{ij}(x_{ij} - x_{kj})$
Evaluate the fitness function for each employed bee using 1/(1+Error)
Evaluate the fitness function for each onlookers bee
All onlookers Distributed ?
No
Select parameters for the onlookers
Yes
Evaluate the fitness function for each v_i
Apply greedy selection between the solutions x_i and v_i
Probabilities calculation for fitness values of employed bees using $P_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i}$
Assign onlooker bees according to probabilities calculated and evaluate the fitness values
Apply greedy selection process to select best solutions
Remember best solution; Cycle= Cycle+1
No
Is Cycle>MCN ?
Yes
Print results
Stop

Figure 2.2. Detailed flow chart of the artificial bee colony algorithm

2.2 Bees Algorithm

The main steps of the bees algorithm are summarized in this section. A number of parameters in the algorithm need to be set in advance: *n* denotes number of scout bees, *m* denotes number of sites selected for neighborhood search (out of n visited sites), *e* denotes number of top-rated (elite) sites among *m* selected sites, *nep* denotes number of bees recruited for the selected sites, *nsp* denotes number of bees recruited for the other (*m-e*) selected sites, *ngh* denotes the initial size of each patch (a patch is a site in the search space that includes the visited site and its neighborhood), *sc* denotes shrinking constant, *a* denotes abandonment threshold (Pham et al, 2006a).

```
    Step 1. Initialise the bee population with n random solutions
    Step 2. Evaluate the fitness of the bee population
    Step 3. While (stopping criterion not met)
    Step 4. Select sites (m) for neighbourhood search
    Step 5. Recruit bees for selected sites (more bees for best e sites), evaluate
fitnesses, select the fittest bee from each site and shrink patches
    For (k = 1; k <= e; k++) // Elite sites
      For (Bee = 1; Bee <= nep; Bee++) // More bees for e elite sites
          BeesPositionInNgh( ) = GenerateRandomValueInNgh(from x + ngh
          to x - ngh);
       Evaluate fitness = Bee(i);
         // Evaluate the fitness of recruited Bee(i)
       If (Bee(i) is better than Bee(i - 1))
    RepresentativeBee = Bee(i);
For (k = e+1; k <= m; k++) // Other selected sites (m - e)
  For (Bee = 1; Bee <= nsp; Bee++)
     // Fewer bees for other selected sites (m - e)
          BeesPositionInNgh( ) = GenerateRandomValueInNgh(from x + ngh
          to x - ngh);
```

```
            Evaluate fitness = Bee($i$);
                // Evaluate the fitness of recruited Bee($i$)
            If (Bee($i$) is better than Bee($i$ - 1))
                RepresentativeBee = Bee($i$);
// Shrink all patches
For (patch = 1; patch <= $m$; patch++)
    $ngh$(patch) = $ngh$(patch)/$sc$;
```

Step 6. If no improvement exists on a site *k* after a given number of iterations, then save the fitness of *k*, abandon the site and assign the bee to random search.

```
    // Site abandonment procedure
    If (Iteration > $a$)
    If (No improvement on the site)
        Save the fitness of site $k$;
        Abandon the site;
        Bee($k$) = GenerateRandomValue(All search space);
```

Step 7. Remaining bees are assigned to search randomly and their fitnesses are evaluated.

```
    // ($n$-$m$) bees assigned randomly to search the all solution space.
```

Step 8. End while

At the beginning of the algorithm *n* scout bees are randomly distributed in the search space. The evaluation of the sites visited by the scout bees using the fitness function (i.e. the performance of the candidate solutions) is in step 2.

The *m* non-dominated sites are assigned as "*selected sites*" and preferred for neighborhood search in step 4. If there exist more than *m* non-dominated sites in the population, the first *m* will be chosen as it is impossible to tell the difference between them. If there are less than *m* non-dominated sites, from the dominated sites that have been dominated just once, the rest will be chosen and this subroutine is continued until an adequate number of sites have been chosen. The algorithm enforces searches in the neighbourhood of the selected sites, assigning more bees to the best *e* sites in

Steps 5. Selection of the best sites can be made directly according to the fitnesses associated with the neighbourhood of the selected sites. Alternatively, fitness values can be used in order to determine the probability of sites being selected. Searches in the neighborhood of the best *e* sites that represent the most promising solutions are made more detailed by recruiting more bees for them than for the other selected sites. This differential recruitment is a key operation of the bees algorithm together with scouting. For each patch, only the one bee which has found the site with the highest fitness (the fittest bee) will be selected to form part of the next bee population. When there does not exist any progress in the neighborhood search, the patch size is diminished. The purpose of this strategy is to make the local search more exploitative, to search more intensely the surrounding of the local optimum. Therefore this step is named as the “shrinking method” (Pham et al., 2007).

In Step 6, if the points visited near a selected site *k* are all inferior in quality to that site, after the number of iterations has reached the abandonment threshold *a*, then the location of the site is memorized and the site abandoned. The bee that found the site is then assigned to random search namely, made to scout for new potential solutions (Pham et al, 2006b).

In Step 7, the rest of the bees in the population are also sent randomly around the search space in order to look for new potential solutions. At the end of each iteration, there are two parts of colony to its new population. These are representative bees from the selected patches and scout bees appointed to attitude random searches. These steps are repeated till a stopping criterion is satisfied (Pham et al, 2006c).

The researchers believe that the algorithm presented here more closely mimics the foraging behaviour of honey bees than other similarly named optimisation algorithms. The detailed flow chart of the bees algorithm is shown in Figure 2.3.

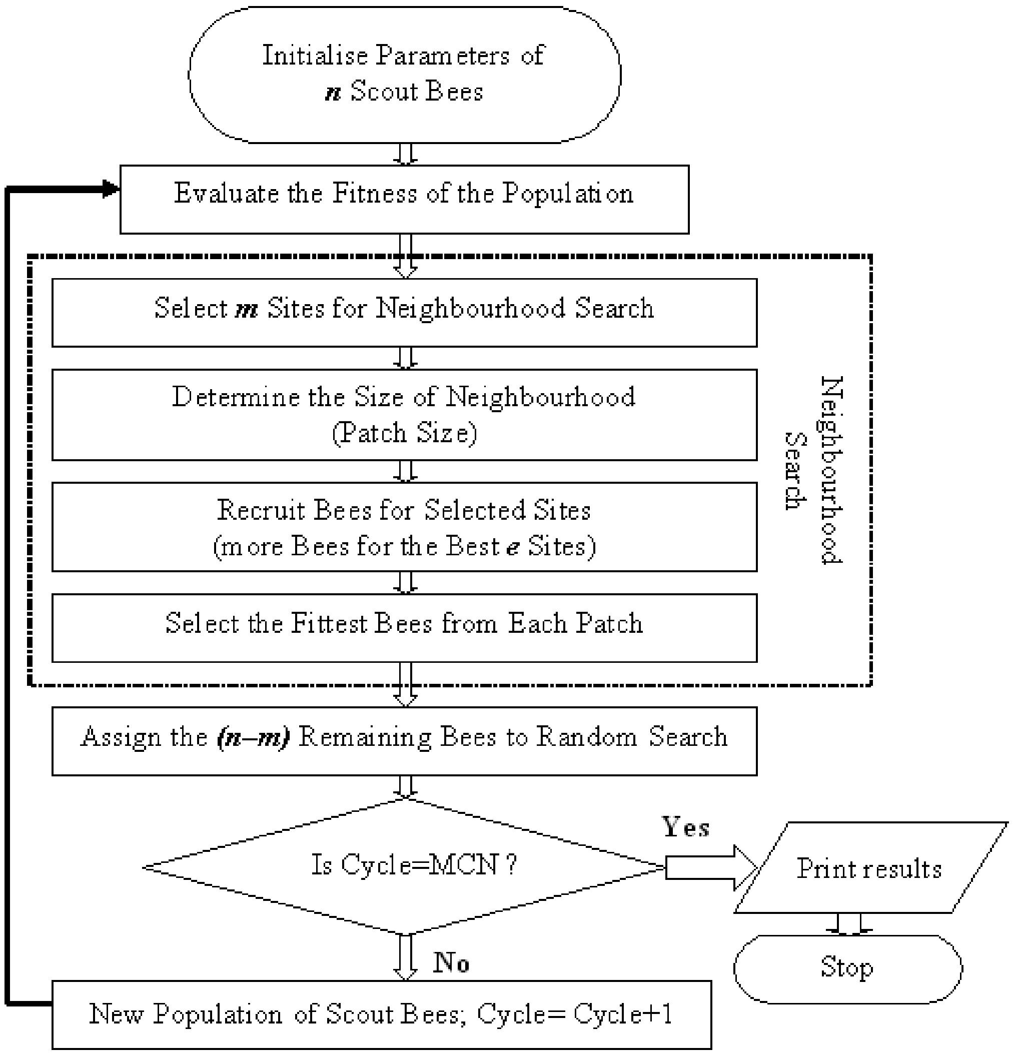
Initialise Parameters of *n* Scout Bees
Evaluate the Fitness of the Population
Select *m* Sites for Neighbourhood Search
Determine the Size of Neighbourhood (Patch Size)
Recruit Bees for Selected Sites (more Bees for the Best *e* Sites)
Select the Fittest Bees from Each Patch
Neighbourhood Search
Assign the *(n–m)* Remaining Bees to Random Search
Is Cycle=MCN ?
Yes
Print results
Stop
No
New Population of Scout Bees; Cycle= Cycle+1

Figure 2.3. Detailed flow chart of the bees algorithm

2.3 Comparison of Honey Bee Algorithms for Numerical Function Optimization

In the experiments, results of the basic BA and ABC algorithms are compared to each other on unimodal Rosenbrock function with different dimensions. For the BA, n=40, e=1, m=3, nep=50, nsp=10, maximum iteration number was 500, initial ngh was 6, and shrinking constant (sc) was 1.05. For the ABC algorithm, number of food sources was 20 and maximum cycle number was 2500. Limit control parameter was 100 for all problems. Each of the experiments was repeated 30 times with different random seeds, and the mean function values of the best solutions and the standard deviations throughout the optimization runs was recorded under different dimensions. In the experiments, mean and standard deviations of 30 runs with different random seeds for 3, 5, 7 and 10 dimensions were reported on Table 2.1 for Rosenbrock function. The test function is Rosenbrock function whose value is 0 at its global minimum (1,1,…,1) in Equation (2.4) (Karaboga and Basturk, 2009). Initialization range for the function is [−30,30] (Karaboga, 2007). The global optimum is inside a long, narrow, parabolic shaped flat valley. Since it is difficult to converge the global optimum, the variables are strongly dependent, and the gradients generally do not point towards the optimum, this problem is repeatedly used to test the performance of the optimization algorithms (Karaboga and Basturk, 2007).

$$f(\vec{x}) = \sum_{i=1}^{D} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \tag{2.4}$$

Table 2.1. Mean and standard deviations of the results obtained by the ABC and the BA algorithms for Rosenbrock function

Algorithm Type	**Dimension (*D*)**				
		3	**5**	**7**	**10**
ABC	**Mean**	2.9905 e-2	9.6872 e-2	1.1202 e-1	1.7071 e-1
	Std	5.4600 e-3	1.7686 e-2	2.0452 e-2	3.1168 e-2
BA	**Mean**	1.6846 e-3	7.6970 e-1	2.1588	19.3714
	Std	3.0756 e-4	1.4052 e-1	3.9414 e-1	3.5367

From the test results, the ABC is superior over the BA. The ABC algorithm preserves producing reasonable results even for high dimensions. The ABC employs less control parameter to be tuned with respect to the BA.

3. SYSTEM IDENTIFICATION

The system identification problem deals with the determination of a mathematical model for a system or a process by observing the input-output data. Historically, system identification has been needed in designing a suitable control process for an unknown system (black box problem) or an incompletely known system (gray box problem). In most practical systems, such as industrial processes, the actual parameter values within a known model structure are unknown. This type of problems, which axe examples of the gray box variety, are more accurately called as system parameter identification problems. System model and parameter identification are applicated not only in engineering but also in other fields like economics, medicine, biology and chemistry (Hsia, 1977). The need for more accurate knowledge of system parameters has increased with recent advances in adaptive and optimal control. The system parameters are updated periodically according to the control system requirement in adaptive control. On the contrary, many recently developed methods of system identification make use of such fields as optimal control theory.

Least squares, the maximum Ukelihood, and the minimum variance methods are available parameter estimation techniques which are more widely used in literature . Researchers and control engineers were interested in the results obtained by their fellows in the computer science field (Astrom and Wittenmark, 1997). They were scrutinizing the artificial intelligence models developed by their fellows because of complex problems unyielding to traditional mathematical techniques. The system identification problem is one of these problems. The problem of system identification with its hard multimodality, nonlinearity and constraints is especially unsuitable for traditional mathematical techniques, and the results obtained using these techniques are deficient for most real life applications. Henceforth, the models developed by their fellows were used by control engineers to solve system identification problems. As the results obtained were encouraging, many artificial intelligence models became the method of choice for many control engineers.

A new wave of artificial intelligence models is the computational intelligence techniques that are biologically inspired or nature inspired techniques. Recent research have shown promising results in their applications in many control engineering problems, and specifically, system identification problems (Liu et al., 2003).

3.1 Linear System Identification

A linear time-invariant discrete-time system is defined using the following linear equations in vector-matrix form:

$$X(k+1) = AX(k) + BU(k) \tag{3.1}$$

$$Y(k) = CX(k) + DU(k) \tag{3.2}$$

where the coefficients A, B, C and D are properly dimensioned matrices. The notation k represents time index. $U(k)$ is the input vector, $Y(k)$ is the output vector, and $X(k)$ is the state vector:

$$U(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ \bullet \\ \bullet \\ \bullet \\ u_r(k) \end{bmatrix}, \quad Y(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \\ \bullet \\ \bullet \\ \bullet \\ y_p(k) \end{bmatrix}, \quad X(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \bullet \\ \bullet \\ \bullet \\ x_n(k) \end{bmatrix}. \tag{3.3}$$

In system identification, the main task is to find a suitable model structure of a system with unknown parameters, given some prior knowledge about the system and input-output observations. Employing the artificial bee colony algorithm for identification, one can exploit their ability to learn the system behavior and requires a reduced amount of knowledge such as observed input-output data and order of the system. The proposed method approximates the system using the observed input-output data pairs and order of the system. In its estimating process, since the identification model is parallel to the system being identified, both of them get the same external input $U(k)$. For the same input, the output of the model $\hat{Y}(k)$ is compared with the output of the system $Y(k)$. Therefore, the error signal $e(k)$ is produced by the difference between the output of the identification model and the output of the system in the following way:

$$e(k) = Y(k) - \hat{Y}(k) \tag{3.4}$$

3.2 Applying Artificial Bee Colony Algorithm in System Identification

A discrete time signal is a sequence

$$u = \{u(0), u(1), \mathbf{K}, u(k) \text{ and } y = \{y(0), y(1), \mathbf{K}, y(k), \mathbf{K}\} \tag{3.5}$$

For a single-input, single-output system of order n, causal LTI systems difference equation can be written as follows:

$$\begin{aligned} y[k] = -b_1 y[k-1] - b_2 y[k-2] - \mathbf{L} - b_n y[k-n] \\ + a_0 u[k] + a_1 u[k-1] + a_2 u[k-2] + \mathbf{L} + a_n u[k-n] \end{aligned} \tag{3.6}$$

where β_1, β_2, ... β_n, α_0, α_1, ... α_n are the system parameters. Applying the Z-transform to Equation (3.6), we can obtain the following:

$$\left(1 + b_1 z^{-1} + \mathbf{L} + b_n z^{-m}\right) Y(z) = \left(a_0 + a_1 z^{-1} + \mathbf{L} + a_n z^{-n}\right) U(z) \qquad m \le n \tag{3.7}$$

Hence, the transfer function, $G(z)$, can be defined as

$$G(z) = \frac{Y(z)}{U(z)}, \tag{3.8}$$

or

$$G(z) = \left(a_0 + a_1 z^{-1} + \mathbf{L} + a_n z^{-m}\right) / \left(1 + b_1 z^{-1} + \mathbf{L} + b_n z^{-n}\right) \qquad m \le n \tag{3.9}$$

where $\alpha_0 \neq 0$. The concept of a transfer function can be extended to a linear system having *p* inputs and *r* outputs, in which case a transfer function matrix, *G*(*z*), is defined. It has a dimension *r x p.* In the identification problem, the parameters in each element of the matrix *G*(*z*) need to be found. The signal y(k) can be constructed by iterative computation, given the input signal and initial conditions. After discrete system transfer function has been formed, linear system equation is written as input and output form. The initial conditions are set to zero. Measured input-output data is used for parameter estimation. At the beginning of parameter estimation, input and output data is known and real system parameters are assumed as unknown. Using initial conditions and obtained real system data, system parameters are estimated with the ABC Algorithm. An error between real system output and estimated system output is defined as error function. The estimation of the system parameters is achieved as a result of minimizing the error function by the ABC algorithm. The sum squared error function is used here as an error criterion:

$$J_e = \frac{1}{2} \sum_k e(k)^2 \tag{3.10}$$

where J_e is the measure of the error. For one training epoch the root-mean-square error (*RMSE*) is represented as follows:

$$RMSE = \sqrt{\frac{\sum_{k=1}^{r} e(k)^2}{r}} \tag{3.11}$$

where *r* represents the number of data.

On the other hand, the ABC algorithm searched based on the well-known *Jury stability criterion* within the stability boundary for parameters estimation (Jury and Blanchard, 1961). Roots of a transfer function in the z-plane must be located to inside the unit circle $|z| \leq 1$ for stability. A zero order hold element for discretization is used.

4. PID CONTROLLER DESIGN

In process control a dominant role is played by PID-controllers and simple and transparent design procedures are demanded by industrial application. PID controller design is based on a direct relationship between the parameters of the controller and the process model. Simple test results like step or pulse response of the plant should be used to ensure a common acceptance for the controller design even for staff with lower theoretical background and knowledge (Ang et al, 2005).

Several approaches have been applied in the past to evaluate design procedures for an optimal frequency response using the conception of unity gain approach and gain adjustment. But universally valid design equations were not found until now. Thus one of the most challenging issues in control engineering education and application is still controller design (Nise, 2004).

It is considered that PID controller is the most common control technique that is extensively used in control applications. The PID controller have been used in daily life by a huge number of applications and control engineers. On the other hand, many research papers, number of master and doctoral theses and books have been written on PID controller design subject. PID control offers an easy method of controlling a process by varying its parameters. PID works well in industrial applications such as slow industrial manipulators were large components of joint inertia added by actuators. Since the invention of PID control in 1910, and Ziegler-Nichols (ZN) tuning method in 1942, PID controllers became dominant and popular issues in control theory because of simplicity of implementation, simplicity of design, and the ability to be used in a widespread range of applications. Moreover, PID controllers are available at low cost. Consequently, if the parameters are tuned properly, it provides robust and reliable performance for most systems. PID variations (P, PD, and PI) are widely used in more than 90% to 95% of industrial control applications. However, there is own limitation of the PID controller; if the requirement is reasonable and the process parameters variation are limited, the PID performances can give only satisfactory performance (Seborg et al, 2004).

Setting the PID parameters is called tuning in literature. Several methods have been proposed for determining the PID controller gains during the last six decades. The information concerning the open loop characteristics such as Cohen-Coon method is used by some of those methods. The Nyquist curve plotting of the plant such as Ziegler-Nichols tuning method is used by other methods. A prior knowledge about the system is needed to be known by all of these tuning formulas. PID control technique is applied to control and enhance the system characteristics such as reducing the overshoot, speed up rising time, and eliminating the steady state error. There is specific criteria for each one of the PID parameters to enhance the characteristics of the controlled system (Lin et al, 2008).

4.1 Controller Design Methods

There are not only many different architectures or configurations for control systems but also many different general approaches to expressing the design goals and objectives for controller design. One example is the optimal controller paradigm: the goal is to determine a controller that minimizes a single cost function or objective. Other approaches such as multicriterion optimization, several different cost functions are specified and the goal is to identify controllers that perform mutually well on these goals. The purpose of this chapter is to develop tuning method for describing design specifications, and to explain how these various approaches can be described using tuning methods.

4.1.1 Iterative or Manual Tuning Method

It is considered that iterative or manual tuning is an experimental method and it is used to determine the PID controller parameters. An experimental procedure using tuning can be outlined as follows:

1. Integral and derivative gains equal zero.
2. Proportional gain is tuned to give the desired response, neglect the steady state error.

3. Increase K_P gain by small increment and adjust the derivative gain K_D to decrease the damping.

4. Adjust the integral gain K_I to remove the steady state error.

5. Replicate the previous steps until acquiring the desired response.

This method concerned as a time consuming method because it depends on trial and error approach.

4.1.2 Ziegler–Nichols Frequency Domain Method

This method is based on the closed loop system response. Initially K_I and K_D gains are set to zero. The proportional gain is increased until the process oscillation occurs. It reaches the critical gain value K_{CR} at which the output of the loop starts to oscillate. Using the value of a critical or ultimate gain K_{CR} and the oscillation or ultimate period P_{CR}, the value of PID parameter K_P, K_I and K_D are given in terms of the ultimate gain and ultimate period (Ziegler and Nichols, 1942):

$$\begin{aligned} K_p &= 0.6K_{CR} \\ K_I &= \frac{2K_P}{P_{CR}} \Rightarrow T_I = 0.5P_{CR} \\ K_D &= \frac{K_P P_{CR}}{8} \Rightarrow T_D = P_{CR}/8 \end{aligned} \tag{4.1}$$

4.1.3 Ziegler–Nichols Time Domain Method

The Ziegler-Nichols step response and frequency response methods are the classical tuning methods for PID controllers. They were presented already in 1942, but they are still widely used in the process industry as the basis for controller tuning. The step response method is based on an open-loop step response test of the process, hence requiring the process to be stable. It depends on the characteristic of the open loop step response of the system. Two parameters are determined: the maximum point of the slop of the step response, and the intersection between the tangent and the time-axis. Four steps to determine *a*, T_d, and PID parameters are described as follows (Hang et al, 1991):

1. Design the control system at the open loop state.

2. Plot the step response as shown in Figure 4.1.

3. Draw tangent line crossing the middle point of the slop of the step response.

4. Determine the PID parameters according to the following relations:

$$\begin{aligned} K_P &= 1.2/a \\ T_I &= 2T_d \\ T_D &= 0.5T_d \end{aligned} \tag{4.2}$$

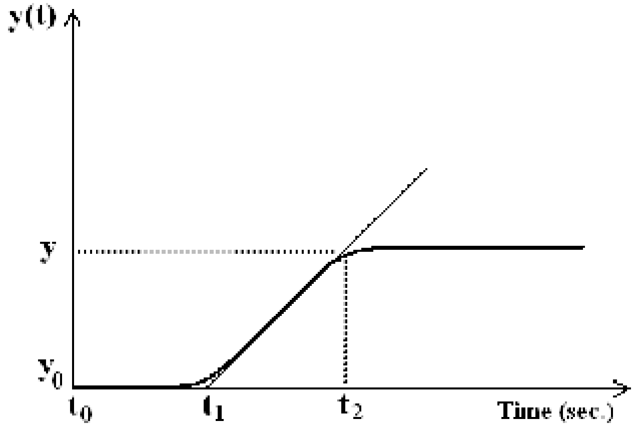

Figure 4.1. Plant step response in Z-N method

where time constant $a = t_2 - t_1$ and the dead time T_d is defined as $t_1 - t_0$.

4.1.4 Cohen-Coon Method

The procedure to find the PID parameters in this method is the same as Ziegler-Nichols time domain method. The Cohen-Coon method's main objective is load disturbance rejection. The PID parameters are calculated according to the following formulas (Cohen and Coon, 1953):

$$\begin{aligned} K_P &= (0.25+1.35T_d/a)/K \\ T_I &= T_d(2.5+0.46T_d/a)/(1+0.61T_d/a) \\ T_D &= 0.37T_d/(1+0.19T_d/a) \end{aligned} \tag{4.3}$$

T_d and a are defined as the same in the Z-N, based on the dead time and the time constant respectively. K is a process gain.

4.1.5 Root Locus Method

Root locus method is a good technique to design the PID parameters. It's a graphical technique that gives a description of the control system as various parameters change, such as overshoot and rising time (Shamsuzzoha and Skogestad, 2010). This method is used to analyze the relationship between the poles, gains, and stability of the system.

Root locus means in control theory, the location of the poles and zeros of transfer function. Pole location determines system stability. If the roots of transfer function in the right half plan of the continuous system or inside the circle of discrete systems, it indicates that the system is unstable, where if these roots in the left half plan this means the system is stable. In addition, when root location on jw axis, the system is considered marginal stable.

4.2 PID Characteristic Parameters

PID controller widely used in industrial control systems is composed of proportional control action, integral control action and derivative control action. There are many forms of PID controller implementations such as a stand-alone controller or Distributed Control System (DCS). Figure 4.2 is a simple diagram showing the schematic of the PID controller and it is known as non-interacting form or parallel form.

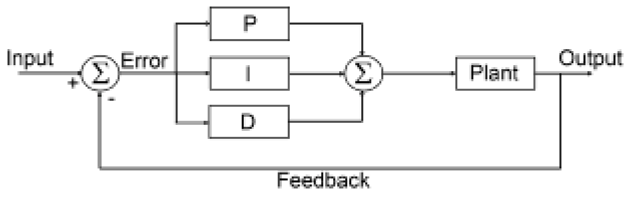

Figure 4.2. Block diagram of a PID controller

The parallel controllers are mostly preferred for higher order systems. The transfer function of PID controller in Laplace transform is defined for a continuous system as

$$G_c(s) = K_P + \frac{K_I}{s} + K_D s \tag{4.4}$$

The proportional controller response is proportional to the control error. The controller error is defined as the difference between the set point and the process output. The proportional controller output is the multiplication of the system error signal and the proportional gain. Proportional term can be mathematically expressed as

$$P_{term} = K_p \times Error \tag{4.5}$$

The integral control applies a control signal to the system which is proportional to the integral of the error. The offset introduced by the proportional control is removed by the integral action but a phase lag is added into the system. Integral term can be mathematically expressed as

$$I_{term} = K_I \times \int Error\, dt \tag{4.6}$$

There is a proportion between the derivative controller output and the rate of change of the error. Derivative control is used to decrease and eliminate overshoot of system response and introduce a phase lead action that removes the phase lag introduced by the integral action.

$$D_{term} = K_D \times \frac{d(Error)}{dt} \tag{4.7}$$

Combining these three types of control together, transfer function of continuous PID controller is formed as

$$G_c(s) = \frac{K_D s^2 + K_P s + K_I}{s} \tag{4.8}$$

where K_P, K_I and K_D are the proportional, integral and derivative gains, respectively. The control signal to the plant is given by

$$u(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt} \tag{4.9}$$

Proportional action K_P improves the system rising time, and reduces the steady state error. This means the larger proportional gain, the larger control signal become to correct the error. However, the higher value of K_P produces large overshoot and the system may be oscillating; therefore, integral action K_I is used to eliminate the steady state error. Despite the integral control, reducing the steady state error, it may make the transient response worse. Therefore, derivative gain K_D will have the effect of increasing the damping in system, reducing the overshoot, and improving the transient response.

As discussed previously, each one of the three gains of the classical PID control has an effect of the response of the closed loop system. Table (4.1) summarizes the effects of each of PID control parameters. It will be known that any changing of one of the three gains will affect the characteristic of the system response.

Table 4.1. PID characteristic parameters

Closed-Loop Response	**Rise Time**	**Overshoot**	**Settling Time**	**Steady State Error**
Increasing Kp	Fast	Increase	Small / No effect	Decrease
Increasing Ki	Fast	Increase	Increase	Decrease
Increasing Kd	Small / No effect	Decrease	Decrease	Small / No effect

To design the proposed controller, four important characteristics of the output of the system are used. These four characteristics are briefly defined below and illustrated in Figure 4.3. In addition, it will be defined as:

Rise time (t_r) is defined as the time required for the step response to rise from 10% to 90% of the set point.

Settling time (t_s) is defined as the time required for the step response to stay within 2% of the set point.

Maximum overshoot (M_p) characterizes what maximum peak value will be reached over the set point. If y_{max} designate the maximum value of y and y_{ss} show the steady-state value of it, the maximum overshoot will be expressed as:

$$M_p = y_{\max} - y_{ss} \tag{4.10}$$

Steady state error (*SSE*): it expresses the final difference between the process variable and the set point. *IAE*, *MSE*, *ISE*, *ITAE*, and *ITSE* are typically and popular integral error criteria. Some error criteria usually have to be minimized to get the PID tuning parameters optimal or near optimal. The Integral Absolute Error (*IAE*) in the controlled variable is formulated by (Seborg et al, 2004)

$$IAE = \int_0^t |e(t)|dt = \int_0^t |r(t) - y(t)|dt \tag{4.11}$$

Now that large errors penalized by the *ISE* criterion results in the most-aggressive settings and persistent errors penalized by the *ITAE* criterion results in the most-conservative settings, moderate settings are produced between *ISE* and *ITAE* criteria by the *IAE* criterion. The Mean of the Squared Error (*MSE*), Integral Square Error (*ISE*), Integral Time Absolute Error (*ITAE*), and Integral Time Square Error (*ITSE*) are also given as follows (Seborg et al, 2004):

$$MSE = \frac{1}{t}\int_0^t e(t)^2\,dt = \frac{1}{t}\int_0^t (r(t) - y(t))^2\,dt \tag{4.12}$$

$$ISE = \int_0^t e(t)^2\,dt = \int_0^t (r(t) - y(t))^2\,dt \tag{4.13}$$

$$ITAE = \int_0^t t|e(t)|dt = \int_0^t t|r(t) - y(t)|dt \tag{4.14}$$

$$ITSE = \int_0^t te(t)^2\,dt = \int_0^t t(r(t) - y(t))^2\,dt \tag{4.15}$$

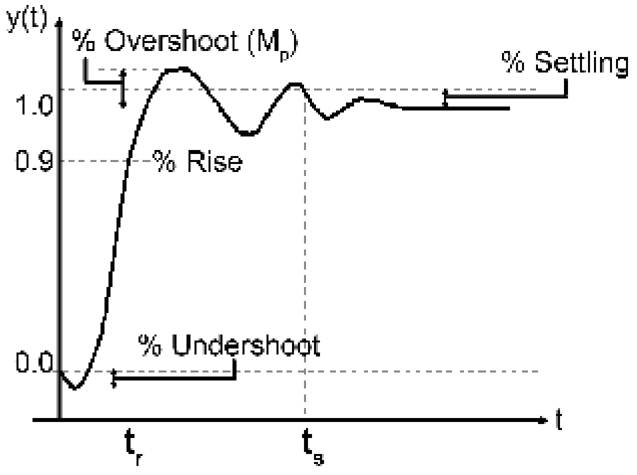

Figure 4.3. Rise time, Settling time, Maximum overshoot

4.3 Principle of Multi-objective Optimization (Pareto Optimality)

The multi-objective optimization is used to minimize all the objective design criteria functions simultaneously. The general multi-objective optimization requiring the optimization of *j* objectives can be written as follows (Ngatchou et al, 2005):

$$\underset{x \in D}{Min}\{ f_1(x) = y_1, f_2(x) = y_2, ..., f_j(x) = y_j \} \tag{4.16}$$

In Equation (4.16), $f_j(x)$ are the *j* th objective design criteria functions and *x* indicates the design parameters chosen and *D* indicates the set of possible design parameters. There are response surface functions $f_j(x)=y_j$ of each response. When there exists a vector of non negative weights $\Phi=[\lambda_1.....\lambda_j]^T$, an efficient solution is supported. Unique global optimum *x* is expressed in the following formula (Ngatchou et al, 2005):

$$\underset{x \in D}{Min} \sum_{j=1}^{N} l_j f_j(x) \tag{4.17}$$

If there does not exist a conflict between the objective functions in Equation (4.16), then a solution *x,* called an *ideal solution*, can be found where every objective function obtains its minimum. Generally, there is not an ideal single solution which is optimal with respect to each objective function. The objective functions are mostly in conflict, that's why the reduction of one objective function usually causes to increase another objective functions. Consequently, Pareto optimal solution is the result of the multi-objective optimization and this solution is possible to improve any of the objective function by increasing at least one of the other objective functions. Pareto optimality cannot improve any criterion without deteriorating a value of at least one other criterion. There are generally a lot of Pareto optimal solutions. There is an equally acceptable solution of the problem for every Pareto optimal point. Nevertheless, the aim is generally desirable to obtain one point as a solution.

A solution vector $x^* \in X$ is called *Pareto optimal* when there does not exist another solution which dominates it in Equation (4.18). That is to say, solution can improve in one of the objectives if it affects at least one other objective.

$$\exists x \in X : f_i(x) \leq f_i(x^*) \wedge f(x) \neq f(x^*); \quad \forall i = \{1,2,...,p\} \tag{4.18}$$

The corresponding objective vector $f(x^*)$ is said to be a *Pareto dominant* vector. A solution vector x_1 dominates another feasible solution x_2, ($x_1 > x_2$) such as

$$f_1(x_1) \leq f_i(x_2) \wedge \exists\, j : f_j(x_1) \leq f_j(x_2); \quad \forall i, j = \{1,2,....,p\} \tag{4.19}$$

If there doesn't exist any solution that dominates x_1, then x_1 is non-dominated. A set of non-dominated feasible solutions $\{x^* \mid \neg\exists x : x > x^*\}$ is called the *Pareto optimal set*. The set of objective vectors which are image of a Pareto set $\{F(x^*) \mid \neg\exists x : x > x^*\}$ is said to be on the *Pareto front* (Ngatchou et al, 2005). A Pareto front for a bi-objective optimization problem is illustrated in Figure 4.4.

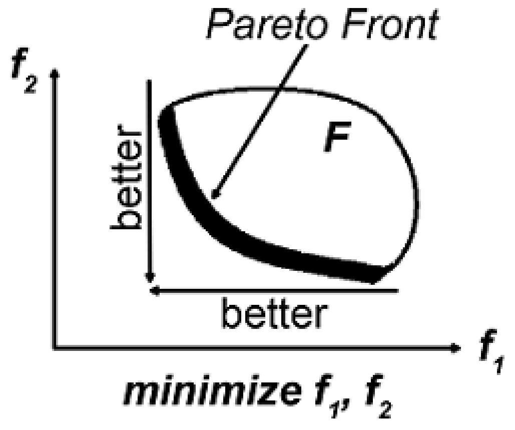

Figure 4.4. Illustration of Pareto front for a bi-objective optimization problem

4.3.1 Weighted Sum Method

The weighted sum method is a traditional, popular method that parametrically changes the weights among objective functions to obtain the Pareto front. Let us consider we have the objective functions f_1, f_2, ..., f_N. This method takes each objective function and multiplies it by a fraction of one, the "weighting coefficient" which is represented by λ_N. The modified functions are then added together to obtain a single cost function, which can easily be solved using any method. Mathematically, the new function is written as

$$F(x) = l_1 f_1(x) + l_2 f_2(x) + \mathbf{L} + l_N f_N(x) \tag{4.20}$$

The method is easy to implement and guarantees finding the Pareto optimal set, provided the objective function space is convex. However, a uniformly distributed set of weights does not necessarily find a uniformly distributed Pareto optimal set, which makes it difficult to obtain a Pareto solution in a desired region of the objective space. The important issue arises in assigning the weighting coefficients (λ_1, λ_2, … λ_3) because the solution is strongly dependent on the chosen weighting coefficients.

4.3.2 Applying Artificial Bee Colony and Bees Algorithm in PID Controller Design

Tuning the parameters of the PID controllers using the multi-objective artificial bee colony and bees algorithm is an optimization problem which needs to be solved in such a way that output of the system attains the desired level in the shortest time as far as possible preventing a high overshoot at the same time. In feedback control loop denoted by Figure 4.5 and Figure 4.6, G_c presents the PID controller that is governed by Equation (4.21), and G_p presents the system to be controlled. In Figure 4.5 and Figure 4.6, *r* denotes the reference input signal, *e* denotes the error signal, *u* denotes the control signal, *y* denotes the output signal, G_p denotes a Linear Time-Invariant (LTI) system, G_c denotes the PID Controller. Using the reference signal $r(t)$ and system output $y(t)$ the error signal are defined as $e(t) = r(t) - y(t)$.

$$G_c(s) = \frac{K_D s^2 + K_P s + K_I}{s} \tag{4.21}$$

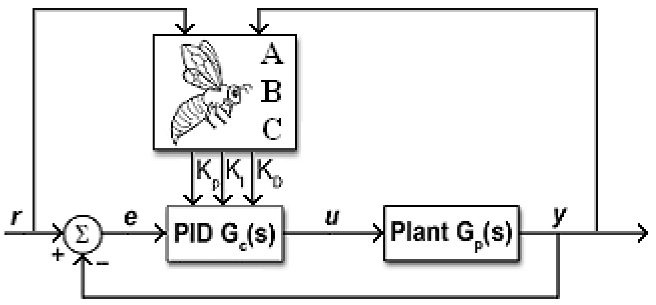

Figure 4.5. Block diagram of a MOABC-PID controller

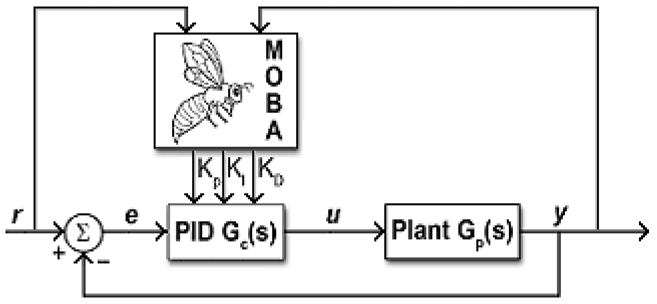

Figure 4.6. Block diagram of a MOBA-PID controller

Optimization criteria which are used to evaluate fitness are to be chosen in applying optimization method. Many indexes of PID controller affecting performance of the transient response can be combined into one objective function composed of the weighted sum of objectives. The set of objective function is represented by Equation (4.22):

$$J^B = \min(\Phi F) \tag{4.22}$$

where J^B denotes the value of the objective function found by the bees, $F = [f_1\ f_2\ f_3\ f_4\ f_5\ f_6\ f_7]^T$ denotes vector of objective functions, f_1 denotes the first objective function including the settling time (t_s), f_2 denotes the second objective function including rise time (t_r), f_3 denotes the third objective function including maximum overshoot (M_p), f_4 denotes the fourth objective function including Integral Absolute Error (*IAE*), f_5 denotes the fifth objective function including Integral Time Absolute Error (*ITAE*), f_6 denotes the sixth objective function including Integral Square Error (*ISE*), f_7 denotes the seventh objective function including Integral Time Square Error (*ITSE*), $\Phi = [\lambda_1\ \lambda_2\ \lambda_3\ \lambda_4\ \lambda_5\ \lambda_6\ \lambda_7]$ denotes vector of non negative weights. The important issue arises in assigning the weighting coefficients ($\lambda_1, \lambda_2, \ldots \lambda_7$) because the solution is strongly dependent on the chosen weighting coefficients.

5. DISCUSSION OF SIMULATION RESULTS

5.1 Results for System Identification

In this subsection, two main examples are utilized in order to illustrate the efficiency of the proposed algorithm: (i) the ABC algorithm in system identification for comparing different linear SISO plants with different order, (ii) the ABC algorithm in system identification for a DC motor. Here, there are four control parameters in the ABC algorithm: The first parameter is the number of food sources which is equal to the number of employed and also onlooker bees (*NP*), the second one is the number of parameters of the problem to be estimated (*D*), the third one is the value of limit parameter (*limit*), and the fourth one is the maximum cycle number (*MCN*). The value of limit is generally chosen as *NP/2×D* (Basturk and Karaboga, 2010).

5.1.1 Example 1: First Order SISO Linear System Indentification

In the ABC algorithm, the values of the control parameters are chosen as D=2, NP=20, limit=20, MCN=500 for the first order plant. The transfer function of the 1st-order plant is given in the s-domain as

$$G_1(s) = \frac{3}{30s+1} \tag{5.1}$$

The transfer function of the 1st-order plant (sampling time = 1.0 second) is given in the z-domain as

$$G_1(z) = \frac{0.098352}{z - 0.967216} \tag{5.2}$$

The difference equation of the G_1 plant is as follows:

$$y[k] = 0.967216y[k-1] + 0.098352u[k-1] \tag{5.3}$$

A training set consisting of 400 data for the first order plant is obtained by using a random input whose amplitude is uniformly distributed in the interval [-2.0, 2.0] for zero initial conditions. For avoiding a similar particular solution, all parameters to be estimated are initialized randomly over the range [-10.0, 10.0]. The proposed algorithm is run 10 times for G_1 plant. For G_1 plant, each parameter is calculated and the simulation results are presented in Table 5.1. The results in Table 5.1 are found by using 20 bees for first order plant.

Table 5.1. Simulation results of the estimated G_1(s) linear plant

Plant	**Parameters**	**Real System**	**Estimated System (ABC Algorithm)**
G_1(s)	**β_1**	**0.967216**	**0.967216**
	α_0	**0.098352**	**0.098352**

The performance of the ABC algorithm is tested with the unit step input and also the following input sequence consisting of mixtures of sinusoids and constant signals:

$$\begin{aligned}
&u(k) = \sin(pk/25), \quad k < 250\\
&u(k) = 1.0, \quad 250 \le k < 500\\
&u(k) = -1.0, \quad 500 \le k < 750\\
&u(k) = 0.3\sin(pk/25) + 0.1\sin(pk/32) + 0.6\sin(pk/10), \quad 750 \le k < 1000
\end{aligned} \tag{5.4}$$

The step response of $G_1(s)$ plotted with the best values of the parameters estimated by the ABC in 10 runs is shown in Figures 5.1. Also the sinusoidal input response of $G_1(s)$ plotted with the best values of the parameters estimated by the ABC in 10 runs is shown in Figures 5.2.

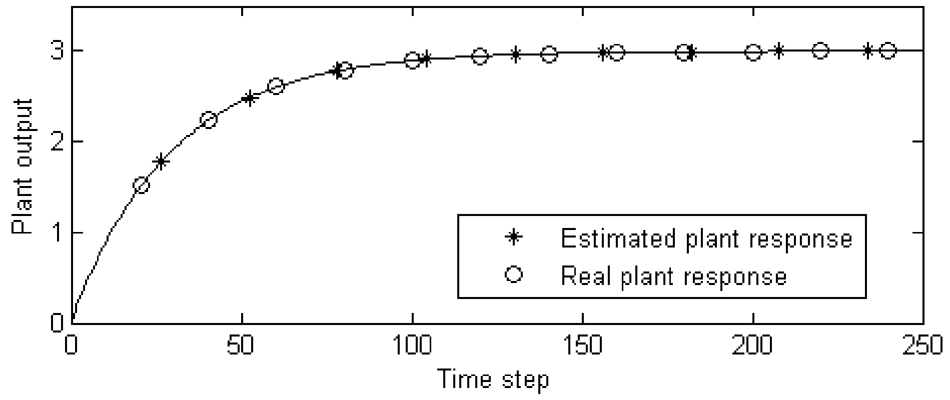

Figure 5.1. Step responses of the plant $G_1(s)$

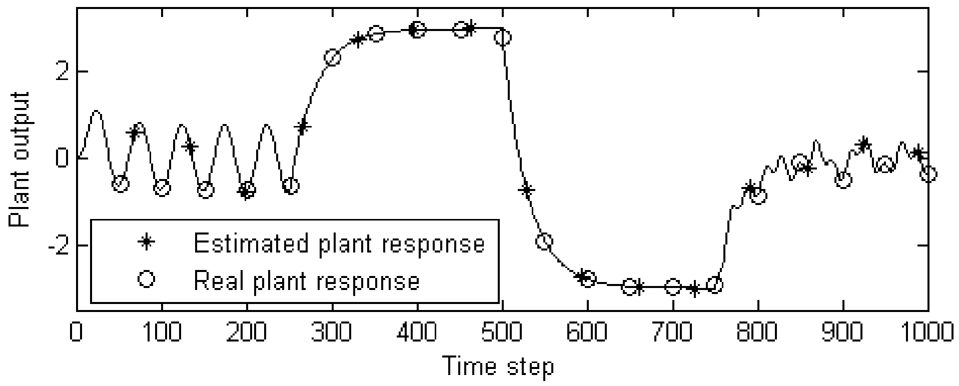

Figure 5.2. Sinusoidal responses of the plant $G_1(s)$

The RMS errors for unit step input and sinusoidal input are presented in Table 5.2. The results show that the value of the RMS error is quite smaller. The ABC algorithm shows satisfactorily performance.

Table 5.2. RMS errors for $G_1(s)$ linear plant

Plant	RMS Error For Step Response Input	RMS Error For Sine wave Input
$G_1(s)$ (First Order)	**0.0**	**0.0**

5.1.2 Example 2: Third Order SISO Linear System Indentification

In the ABC algorithm, the values of the control parameters are chosen as D=6, NP=50, limit=150, MCN=2000 for the third order plant. The transfer function of the 3rd-order plant is given in the s-domain as

$$G_2(s) = \frac{750}{s^3 + 36s^2 + 205s + 750} \tag{5.5}$$

The transfer function of the 3rd-order plant (sampling time = 0.1 second) is given in the z-domain as

$$G_2(z) = \frac{0.057176\mathrm{z}^2 + 0.107891z + 0.009899}{z^3 \text{ -}1.414464\mathrm{z}^2 + 0.616755z - 0.027324} \tag{5.6}$$

The difference equation of the G_2 plant is as follows:

$$\begin{aligned} y[k] =& 1.414464y[k-1] - 0.616755y[k-2] + 0.027324y[k-3] \\ &+ 0.057176u[k-1] + 0.107891u[k-2] + 0.009899u[k-3] \end{aligned} \tag{5.7}$$

A training set consisting of 2000 data for the third order plant is obtained by using a random input whose amplitude is uniformly distributed in the interval [-2.0, 2.0] for zero initial conditions. For avoiding a similar particular solution, all parameters to be estimated are initialized randomly over the range [-10.0, 10.0]. The proposed algorithm is run 10 times for G_2 plant. For G_2 plant, each parameter is calculated and the simulation results are presented in Table 5.3. The results in Table 5.3 are found by using 50 bees for third order plant.

Table 5.3. Simulation results of the estimated $G_2(s)$ linear plant

Plant	**Parameters**	**Real System**	**Estimated System (ABC Algorithm)**
$G_2(s)$	**β_1**	**1.414464**	**1.114346**
	β_2	**-0.616755**	**-0.188177**
	β_3	**0.027324**	**-0.151084**
	α_0	**0.057176**	**0.057323**
	α_1	**0.107891**	**0.124714**
	α_2	**0.009899**	**0.042447**

The performance of the ABC algorithm is tested with the unit step input and also the following input sequence consisting of mixtures of sinusoids and constant signals:

$$\begin{aligned}
&u(k) = \sin(pk/25), \qquad k < 250 \\
&u(k) = 1.0, \qquad 250 \le k < 500 \\
&u(k) = -1.0, \qquad 500 \le k < 750 \\
&u(k) = 0.3\sin(pk/25) + 0.1\sin(pk/32) + 0.6\sin(pk/10), \qquad 750 \le k < 1000
\end{aligned} \tag{5.8}$$

The step response of $G_2(s)$ plotted with the best values of the parameters estimated by the ABC algorithm in 10 runs is shown in Figures 5.3. Also the sinusoidal input response of $G_2(s)$ plotted with the best values of the parameters estimated by the ABC algorithm in 10 runs is shown in Figures 5.4.

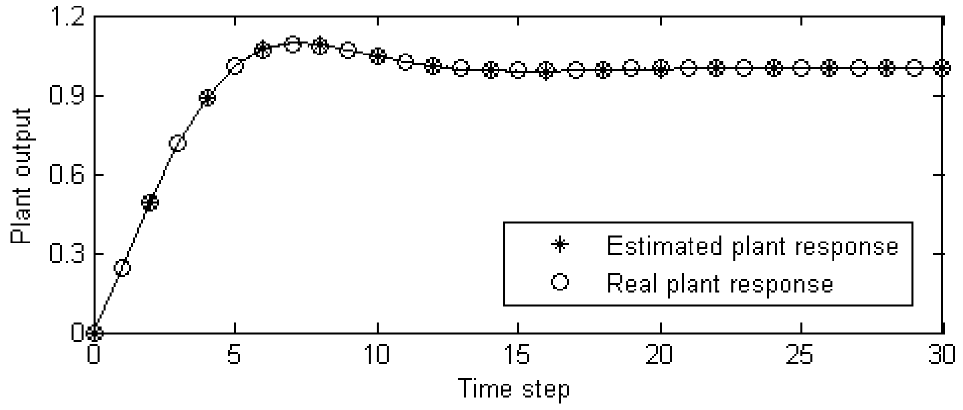

Figure 5.3. Step responses of the plant $G_2(s)$

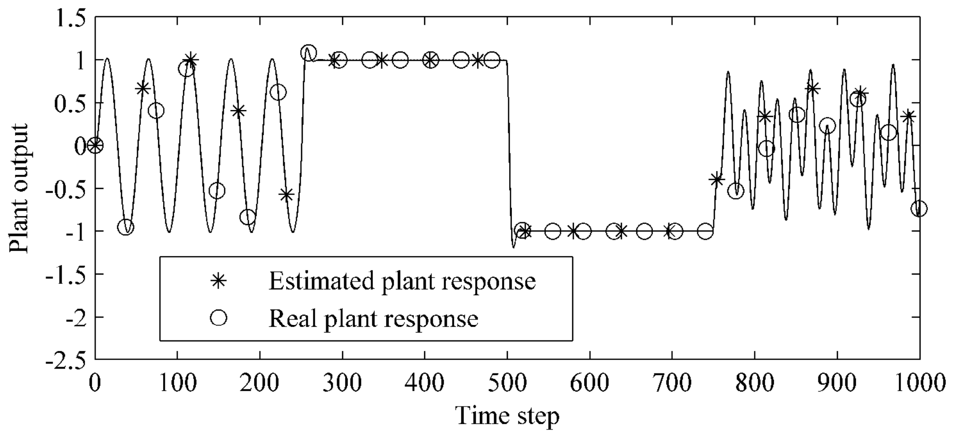

Figure 5.4. Sinusoidal responses of the plant $G_2(s)$

The RMS errors for unit step input and sinusoidal input are presented in Table 5.4. The simulation results show that the value of the RMS error is quite smaller and the simulation results have demonstrated the effectiveness of the proposed algorithm.

Table 5.4. RMS errors for $G_2(s)$ linear plant

Plant	**RMS Error For Step Response Input**	**RMS Error For Sine wave Input**
G_2(s) (Third Order)	**0.0019**	**0.0027**

5.1.3 Example 3: Fifth Order SISO Linear System Indentification

In the ABC algorithm, the values of the control parameters are chosen D=10, NP=50, limit=250, MCN=2000 for the fifth order plant. The transfer function of the 5th-order plant is given in the s-domain as

$$G_3(s) = \frac{-6.35\times10^{-6}s^4 + 4.933\times10^{-5}s^3 + 2812s^2 + 1.172\times10^4 s + 1.953\times10^4}{s^5 + 32.5s^4 + 475s^3 + 3625s^2 + 1.422\times10^4 s + 1.914\times10^4} \tag{5.9}$$

The transfer function of the 5th-order (sampling time = 0.1 second) is given in the z-domain as

$$G_3(z) = \frac{0.225545\mathrm{z}^4 + 0.071233z^3 - 0.490390\mathrm{z}^2 + 0.197875z + 0.035971}{z^5 \text{ - } 2.380647\mathrm{z}^4 + 2.335256z^3 - 1.204551z^2 + 0.328146z - 0.038774} \tag{5.10}$$

The difference equation of the G_3 plant is as follows:

$$\begin{aligned} y[k] = {} & 2.380647y[k-1] - 2.335256y[k-2] + 1.204551y[k-3] \\ & - 0.328146y[k-4] + 0.038774y[k-5] + 0.225545u[k-1] \\ & + 0.071233u[k-2] - 0.490390u[k-3] + 0.197875u[k-4] \\ & + 0.035971u[k-5] \end{aligned} \tag{5.11}$$

A training set consisting of 2000 data for the fifth order plant is obtained by using a random input whose amplitude is uniformly distributed in the interval [-2.0, 2.0] for zero initial conditions. For avoiding a similar particular solution, all parameters to be estimated are initialized randomly over the range [-10.0, 10.0]. The proposed algorithm is run 10 times for G_3 plant. For G_3 plant, each parameter is calculated and the simulation results are presented in Table 5.5. The results in Table 5.5 are found by using 50 bees for fifth order plant.

Table 5.5. Simulation results of the estimated $G_3(s)$ linear plant

Plant	**Parameters**	**Real System**	**Estimated System (ABC Algorithm)**
$G_3(s)$	**β_1**	**2.380647**	**1.305520**
	β_2	**-2.335256**	**-0.984349**
	β_3	**1.204551**	**0.283957**
	β_4	**-0.328146**	**-0.043367**
	β_5	**0.038774**	**-0.034219**
	α_0	**0.225545**	**0.225264**
	α_1	**0.071233**	**0.313395**
	α_2	**-0.490390**	**-0.140650**
	α_3	**0.197875**	**0.047479**
	α_4	**0.035971**	**0.033275**

The performance of the ABC algorithm is tested with the unit step input and also the following input sequence consisting of mixtures of sinusoids and constant signals:

$$\begin{aligned}
&u(k) = \sin(pk/25), \qquad k < 250\\
&u(k) = 1.0, \qquad 250 \le k < 500\\
&u(k) = -1.0, \qquad 500 \le k < 750\\
&u(k) = 0.3\sin(pk/25) + 0.1\sin(pk/32) + 0.6\sin(pk/10), \qquad 750 \le k < 1000
\end{aligned} \tag{5.12}$$

The step response of $G_3(s)$ plotted with the best values of the parameters estimated by the ABC algorithm in 10 runs is shown in Figures 5.5. Also the sinusoidal input response of $G_3(s)$ plotted with the best values of the parameters estimated by the ABC algorithm in 10 runs is shown in Figures 5.6.

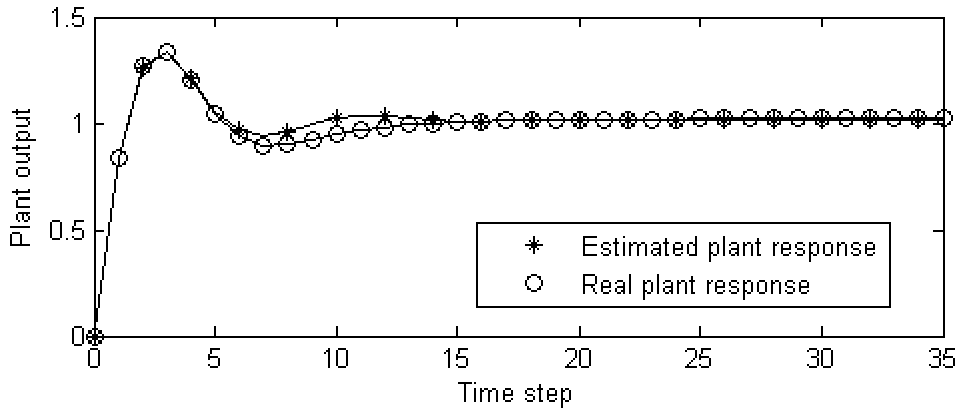

Figure 5.5. Step responses of the plant $G_3(s)$

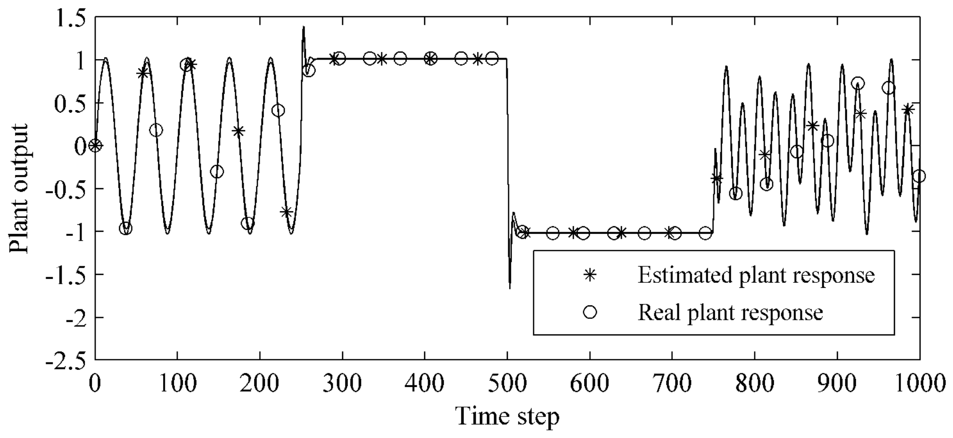

Figure 5.6. Sinusoidal responses of the plant $G_3(s)$

The RMS errors for unit step input and sinusoidal input are presented in Table 5.6. The simulation results show that the value of the RMS error is quite smaller for $G_3(s)$ linear plant.

Table 5.6. RMS errors for $G_3(s)$ linear plant

Plant	**RMS Error For Step Response Input**	**RMS Error For Sine wave Input**
$G_3(s)$ (Fifth Order)	**0.0104**	**0.0345**

5.1.4 Example 4: Seventh Order SISO Linear System Indentification

In the ABC algorithm, the values of the control parameters are chosen as D=14, NP=50, limit=350, MCN=2000 for the seventh order plant. The transfer function of the 7th-order plant is given in the s-domain as

$$G_4(s) = \frac{Y_4(s)}{U_4(s)} \tag{5.13}$$

where

$$\begin{aligned} Y_4(s) &= 1.435\times10^{-5}s^6 + 6.232\times10^{-6}s^5 + 8.882\times10^{-5}s^4 \\ &\quad -1.699\times10^{-5}s^3 + 1.671\times10^{-4}s^2 + 17.98s - 17.98 \\ U_4(s) &= s^7 + 5.234s^6 + 19.7s^5 + 45.92s^4 + 76.52s^3 \\ &\quad 84.09s^2 + 57.11s + 17.98 \end{aligned}$$

The transfer function of the 7th-order plant (sampling time = 1.0 second) is given in the z-domain as

$$G_4(z) = \frac{Y_4(z)}{U_4(z)} \tag{5.14}$$

where

$$\begin{aligned} Y_4(z) &= 0.008550z^6 + 0.102751z^5 - 0.123576z^4 - 0.502080z^3 \\ &\quad - 0.248905z^2 - 0.026393z - 0.000277 \\ U_4(z) &= z^7 - 0.363682z^6 + 0.257812z^5 - 0.166412z^4 + 0.096288z^3 \\ &\quad - 0.047987z^2 + 0.019244z - 0.005333 \end{aligned}$$

The difference equation of the G_4 plant is as follows:

$$\begin{aligned} y[k] &= 0.363682y[k-1] - 0.257812y[k-2] + 0.166412y[k-3] \\ &\quad - 0.096288y[k-4] + 0.047987y[k-5] - 0.019244y[k-6] \\ &\quad + 0.005333y[k-7] + 0.008550u[k-1] + 0.102751u[k-2] \\ &\quad - 0.123576u[k-3] - 0.502080u[k-4] - 0.248905u[k-5] \\ &\quad - 0.026393u[k-6] - 0.000277u[k-7] \end{aligned} \tag{5.15}$$

A training set consisting of 2000 data for the seventh order plant is obtained by using a random input whose amplitude is uniformly distributed in the interval [-2.0, 2.0] for zero initial conditions. For avoiding a similar particular solution, all parameters to be estimated are initialized randomly over the range [-10.0, 10.0]. The proposed algorithm is run 10 times for G_4 plant. For G_4 plant, each parameter is calculated and the simulation results are presented in Table 5.7. The results in Table 5.7 are found by using 50 bees for seventh order plant.

The performance of the ABC algorithm is tested with the unit step input and also the following input sequence consisting of mixtures of sinusoids and constant signals:

$$\begin{aligned} &u(k) = \sin(pk/25), \qquad k < 250 \\ &u(k) = 1.0, \qquad 250 \le k < 500 \\ &u(k) = -1.0, \qquad 500 \le k < 750 \\ &u(k) = 0.3\sin(pk/25) + 0.1\sin(pk/32) + 0.6\sin(pk/10), \qquad 750 \le k < 1000 \end{aligned} \tag{5.16}$$

Table 5.7. Simulation results of the estimated $G_4(s)$ linear plant

Plant	**Parameters**	**Real System**	**Estimated System (ABC Algorithm)**
$G_4(s)$	β_1	**0.363682**	**0.260398**
	β_2	**-0.257812**	**-0.497459**
	β_3	**0.166412**	**0.169752**
	β_4	**-0.096288**	**-0.060764**
	β_5	**0.047987**	**0.012406**
	β_6	**-0.019244**	**0.004800**
	β_7	**0.005333**	**-0.005562**
	α_0	**0.008550**	**0.008500**
	α_1	**0.102751**	**0.103899**
	α_2	**-0.123576**	**-0.110693**
	α_3	**-0.502080**	**-0.485597**
	α_4	**-0.248905**	**-0.327795**
	α_5	**-0.026393**	**-0.207619**
	α_6	**-0.000277**	**-0.094800**

The step response of $G_4(s)$ plotted with the best values of the parameters estimated by the ABC algorithm in 10 runs is shown in Figures 5.7. Also the sinusoidal input response of $G_4(s)$ plotted with the best values of the parameters estimated by the ABC algorithm in 10 runs is shown in Figures 5.8.

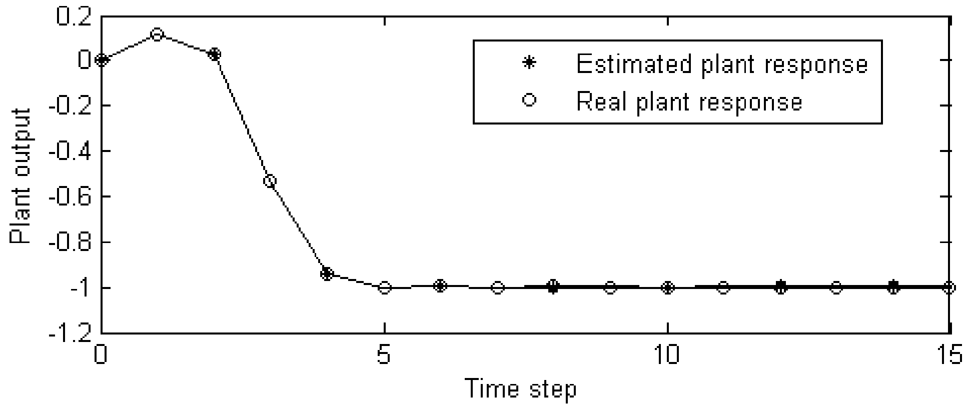

Figure 5.7. Step responses of the plant $G_4(s)$

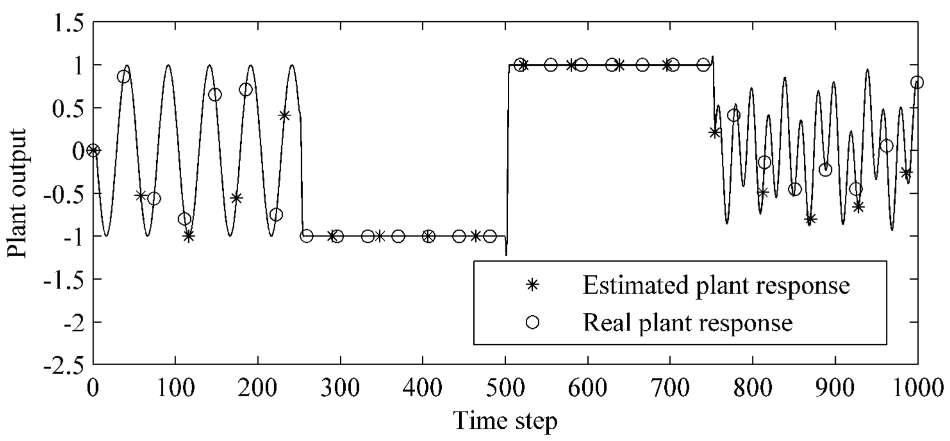

Figure 5.8. Sinusoidal responses of the plant $G_4(s)$

The RMS errors for unit step input and sinusoidal input are presented in Table 5.8. The simulation results show that the value of the RMS error is quite smaller for $G_4(s)$ linear plant.

Table 5.8. RMS errors for $G_4(s)$ linear plant

Plant	**RMS Error For Step Response Input**	**RMS Error For Sine wave Input**
$G_4(s)$ (seventh Order)	**0.0021**	**0.0017**

5.1.5 Example 5: DC Motor Indentification

An identification design method is presented for a permanent magnet DC motor. Simplified mathematical model of DC motor has been used in order to build the DC motor's transfer function. There are differential equations of the electrical part and mechanical part in DC motor model and also it consists in the interconnection between them.

Using simplified equivalent electromechanical diagram of the DC motor, illustrated in Figure 5.9, the mathematical model is written as (Ong, 1998).

$$U_a(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + e_v(t) \tag{5.17}$$

$$e(t) = K_e \Omega(t) \tag{5.18}$$

$$C_m(t) = K_m i_a(t) \tag{5.19}$$

$$C_m(t) = J \frac{d\Omega(t)}{dt} + B\Omega(t) \tag{5.20}$$

where C_m denotes motor torque (Nm), I_a denotes rotor circuit current (A), K_e denotes electrical constant, K_m denotes mechanical constant, L_a denotes rotor circuit inductance (H), R_a denotes rotor circuit resistance (Ohm), U_a denotes input voltage (V), B denotes damping ratio (Nms), e_v denotes electromotive voltage (V), J denotes rotor moment of inertia (kgm^2), Ω denotes rotor speed (rad/s).

The transfer function of the motor model is obtained to allow the control of speed by the voltage input from the characteristic equations of the DC motor. It is given by

$$\frac{\Omega(s)}{U_a(s)} = \frac{K_m}{L_a J s^2 + (R_a J + L_a B)s + (R_a B + K_e K_m)} \tag{5.21}$$

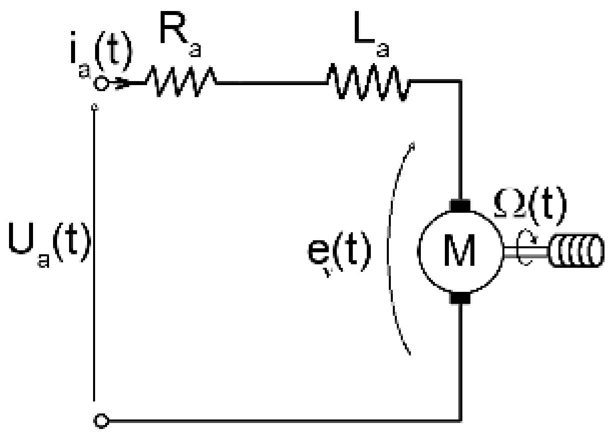

Figure 5.9. A DC Motor equivalent circuit

This transfer function makes possible to simulate motor behavior to various inputs. The specifications of the motor used for simulation are given in Table 5.9.

Table 5.9. Parameters of the Motor (Ong, 1998)

Parameters	**Value**
Armature circuit Resistance (R_a)	21.2 ohm
Armature circuit Inductance (L_a)	0.052 H
Back-Emf constant (K_m)	0.1433 Kg-m/A
Coefficient of friction (B)	$1\text{x}10^{-4}$ Nms
Moment of Inertia (J)	$1\text{x}10^{-5}$ kgm^2
Torque constant (K_e)	0.1433 V/rad/s

A training set consisting of 400 data is obtained using a random input whose amplitude is uniformly distributed in the interval [-2.0, 2.0] for zero initial conditions. Simulations are carried out by using employed and onlooker bees *NP*=20, the maximum cycle number *MCN*=500 for the DC motor. For avoiding a similar particular solution, all parameters are initialized randomly over the range [-10.0, 10.0]. The proposed algorithm is run 10 times for the DC motor parameter estimation. The DC motor transfer function in s domain is given by

$$G_{DC\_MOTOR}(s) = \frac{0.1443}{5.2\times10^{-7}\mathrm{s}^2 + 2.172\times10^{-4}s + 0.0227} \tag{5.22}$$

The discrete time transfer function of the 2nd-order DC motor model (sampling time 0.001 second) is given by

$$G_{DC\_MOTOR}(z) = \frac{0.12z + 0.1044}{\mathrm{z}^2 - 1.623z + 0.6586} \tag{5.23}$$

Its difference equation is given by

$$y[k] = 1.623y[k-1] - 0.6586y[k-2] + 0.12u[k-1] + 0.1044u[k-2] \tag{5.24}$$

The plant given in Equation (5.22) is tested with a unit step input and also the input sequence consisting of mixtures of sinusoids and constant signals given in Equation (5.16) to show the effectiveness and performance of the proposed method. Comparative graph of the actual and the simulated dynamic response with the identified parameters is illustrated in Figure 5.10 and Figure 5.11. Figure 5.10 and 5.11 show a considerable agreement between the actual plant response and identified plant response using the estimated parameters. The real parameters and estimated parameters of the DC motor are given in Table 5.10. The *RMS* errors for step and sinusoidal inputs are given in Table 5.11.

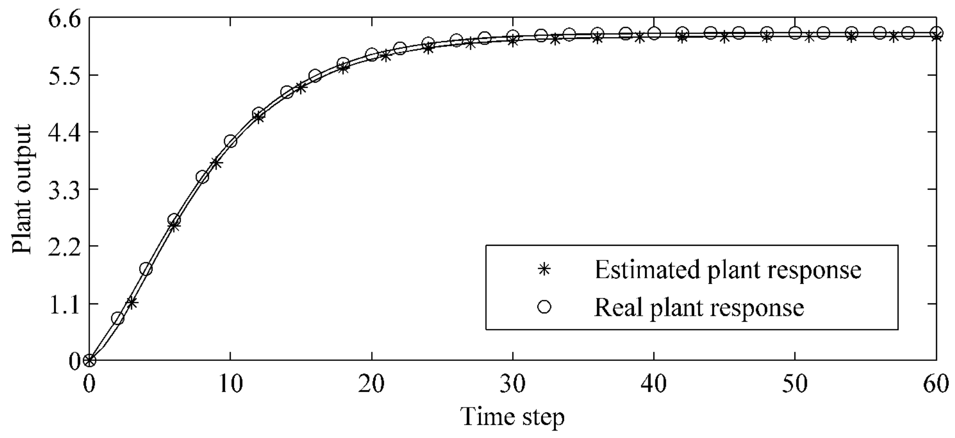

Figure 5.10. Step response of the DC motor

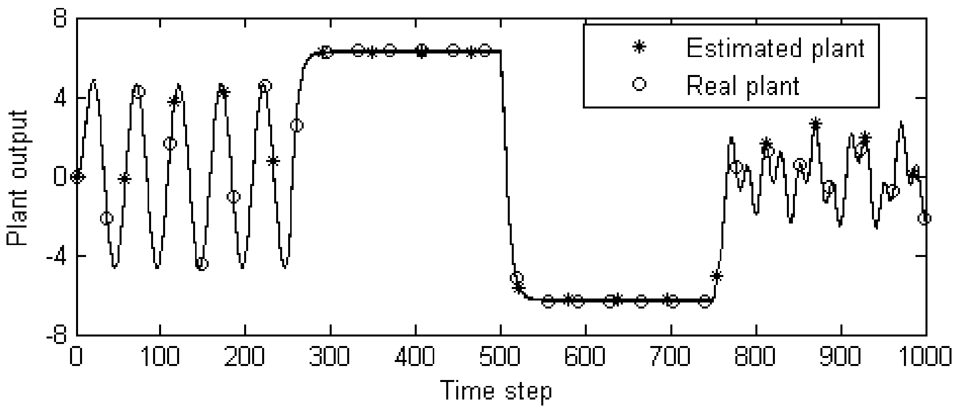

Figure 5.11. Sinusoidal response of the DC motor

Table 5.10. The real and estimated parameters of the DC motor

Plant	Parameters	Real System	Estimated System (ABC Algorithm)
$G_{DC\_MOTOR}(s)$	β_1	**1.6230**	**1.5940**
	β_2	**-0.6586**	**-0.6332**
	α_0	**0.1200**	**0.0101**
	α_1	**0.1044**	**0.2343**

Table 5.11. *RMS* errors for DC Motor

Plant	RMS Error For Unit Step Input	RMS Error For Sine wave Input
$G_{DC\_MOTOR}(s)$ (Second Order)	**0.0722**	**0.0844**

5.2 Results for PID Controller Tuning

The performance of the MOBA and the MOABC were tested with two plants with different order. The objective function which should be minimized was composed of the objective functions f_1, f_2, f_3, and f_6 which include the settling time, the rise time, the maximum overshoot and the integral square error, respectively. The vector of weights is defined as $\Phi = [0.000001 \quad 0.0001 \quad 1 \quad 0 \quad 0 \quad 0.0001 \quad 0]$. Throughout the optimization process, the MOBA and the MOABC use the step reference input and closed loop step response of the process.

The tuning algorithm looks for the optimal parameters for the PID controller to satisfy the desired system specifications by using the changed closed loop control performance according to the adjusted controller parameters at the each iteration. The closed loop response was compared with a step change of a number of simulated systems in order to demonstrate the effectiveness of the presented method. For PID controller tuning, two various processes with different order are used.

In control system applications, a weighted combination of different performance characteristics such as rise time, settling time, maximum overshoot and integral of the square of the error is the chosen performance criterion. The desired system response needs minimal rise time, minimal settling time with a small or no overshoot in the step response of the closed loop system. Hence, the objective function *F* is defined using the performance indices consisting of integral of the square of the error (*ISE*), rise time *(t_r)*, settling time (t_s) and percentage overshoot (M_p).

$$F = l_1(t_s) + l_2(t_r) + l_3(M_p) + l_4(IAE) + l_5(ITAE) + l_6(ISE) + l_7(ITSE) \quad (5.25)$$

In Equation (5.25), the weighting factors are the variables of λ_1, λ_2, λ_3, λ_4, λ_5, λ_6 and λ_7. By adjusting these factors, the most convenient PID controller parameters can be provided in order to achieve the desirable closed loop characteristics of the system. For the predetermined control objectives the performance of the PID controller can be significantly improved. To obtain better solution, weighting factors are defined as $\lambda_1=0.000001$, $\lambda_2=0.0001$, $\lambda_3=1$, $\lambda_4=0$, $\lambda_5=0$, $\lambda_6=0.0001$, $\lambda_7=0$ and the best results obtained for the parameters of the PID controlled systems optimized by *ISE* error criteria. Weighting factor variables depends on being chosen error criteria such as $\lambda_4=0.0001$ for *IAE* error criteria, $\lambda_5=0.0001$ for *ITAE* error criteria, $\lambda_6=0.0001$ for *ISE* error criteria, $\lambda_7=0.0001$ for *ITSE* error criteria. The closed loop PID controller was tuned for the values K_P, K_I and K_D first by using Ziegler-Nichols method, genetic algorithm and ant colony algorithm. In addition, the closed loop PID controller was tuned by using the MOBA and the MOABC.

5.2.1 Example 1: PID Controller Tuning for Third Order Linear Plant

For PID controller tuning, third order plant are used as follows (Bagis, 2007):

$$G_1(s)=\frac{4.228}{(s+0.5)(s^2+1.64s+8.456)} \tag{5.26}$$

The results were found by using 200 scout bees. For avoiding a similar particular solution, the initial populations were generated at random within the range $0.0 \le K_P \le 3.0$, $0.0 \le K_I \le 3.0$, $0.0 \le K_D \le 3.0$. The results in Table 5.12 show that the value of the maximum overshoot is quite smaller, nearly zero percent and the values of the rise time, the settling time for *ISE* error criteria obtained by the MOABC and the MOBA are much less than the values of the other methods. The results of the other methods in Table 5.12 were taken from existing literature (Bagis, 2007).

Furthermore, the step responses of $G_1(s)$ tested with the optimum values of the parameters K_P, K_I and K_D which are obtained by the MOABC and the MOBA are presented in Table 5.13 according to some error criteria. Table 5.13 shows the values of the parameters adopted for the ABC and the BA. The values were decided empirically for the ABC and the BA.

Table 5.12. Comparison of simulation results of the PID controlled systems (Hsiao et al., 2004)

$G_1(s)$	Algorithm Parameters	PID Parameters	Ziegler Nichols	Genetic Algorithm	Ant Colony Optimization	Proposed Algorithms
BA	n=200; e=20; m=80 nep=60; nsp=40 ngh=6.0 sc=2.0 iter=2000 runtime=30	K_P	2.19	1.637	2.517	2.784
		K_I	2.126	0.964	2.219	1.001
		K_D	0.565	0.387	1.151	2.087
		f_1:t_s	6.6	5.97	6.51	6.4691
		f_2:t_r	0.8	2.45	0.627	0.4959
		f_3:M_p	%16.46	%3	%16	%0.1676
		f_6:ISE	0.785	0.588	0.684	0.508
ABC	n=200 limit=100 iter=2000 runtime=30	K_P	2.19	1.637	2.517	2.6213
		K_I	2.126	0.964	2.219	0.8719
		K_D	0.565	0.387	1.151	2.4816
		f_1:t_s	6.6	5.97	6.51	6.5249
		f_2:t_r	0.8	2.45	0.627	0.4553
		f_3:M_p	%16.46	%3	%16	%0.0513
		f_6:ISE	0.785	0.588	0.684	0.400

Table 5.13. Simulation results of the proposed algorithms for different error criteria

$G_1(s)$	Algorithm Parameters	PID Parameters	For ISE	For IAE	For ITAE	For ITSE	For MSE
BA	n=200; e=20; m=80 nep=60; nsp=40 ngh=6.0 sc=2.0 iter=2000 runtime=30	K_P	2.7844	2.5783	2.5562	2.9078	2.7418
		K_I	1.0012	0.9380	0.7160	0.9959	0.9171
		K_D	2.0873	1.6281	0.5932	1.7206	2.3044
		f_1:t_s	6.4691	6.8121	10.3541	6.8534	6.5094
		f_2:t_r	0.4959	0.6026	0.7980	0.5384	0.4701
		f_3:M_p	%0.1676	%0.0644	%0.0032	%0.0279	%0.0185
ABC	n=200 limit=100 iter=2000 runtime=30	K_P	2.6213	2.4046	2.7860	2.7193	2.7895
		K_I	0.8719	0.8973	0.9345	0.9745	0.9425
		K_D	2.4816	1.3700	2.2043	1.8512	2.2173
		f_1:t_s	6.5249	5.7213	6.5438	6.6587	6.5190
		f_2:t_r	0.4553	0.7335	0.4804	0.5366	0.4782
		f_3:M_p	%0.0513	%0.0564	%0.0026	%0.0891	%0.0245

The step responses of $G_1(s)$ plotted with the optimum values of the parameters K_P, K_I and K_D which are obtained by the MOBA are shown in Figures 5.12. Furthermore, the step responses of $G_1(s)$ plotted with the optimum values of the parameters K_P, K_I and K_D which are obtained by the MOABC are shown in Figures 5.13. Step response results of $G_1(s)$ process obtained by using Ziegler Nichols, genetic algorithm and ant colony optimization algorithm are represented for comparison purposes.

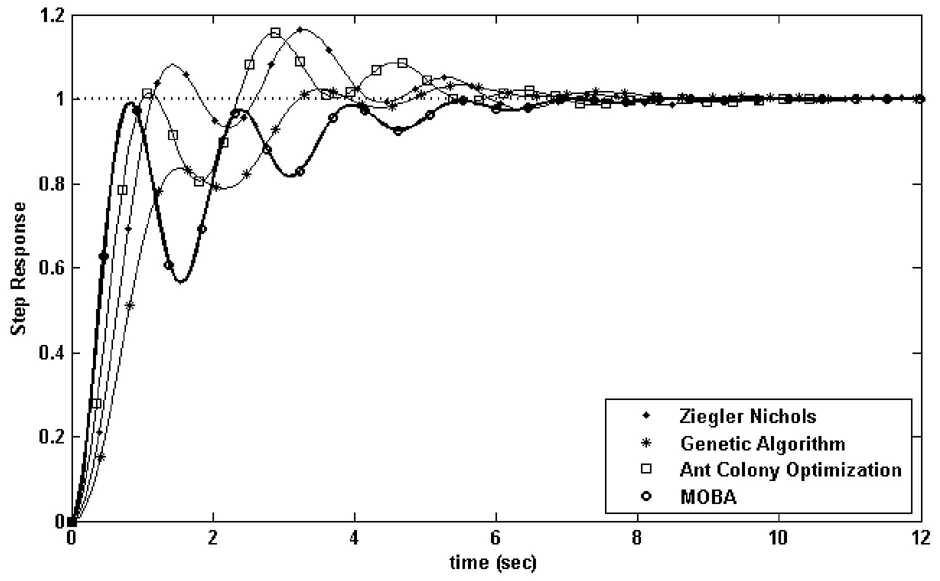

Figure 5.12. Comparison of step responses of the plant $G_1(s)$ for the MOBA

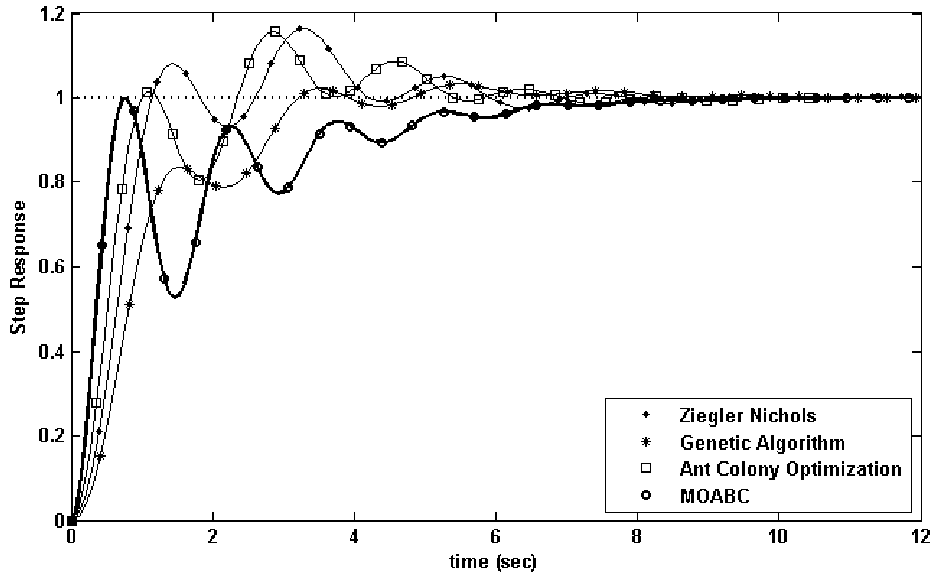

Figure 5.13. Comparison of step responses of the plant $G_1(s)$ for the MOABC

Figures 5.14 and Figures 5.15 illustrate the graphs of the obtained three-dimensional Pareto optimal fronts consisting of the settling time, overshoot and ISE error criteria for the step response of $G_1(s)$ process related with each transfer function. Thus, a well distributed set of non-dominated solutions along the Pareto-optimal front can be found. The MOABC gives better responses than those produced by using the other methods. Thus, it can be considered that the MOABC improves the optimal system performance of the PID controllers satisfactorily. Evaluation of the objective function on the above mentioned $G_1(s)$ plant is presented in Figures 5.16 and Figures 5.17. It is also observed that the objective function value decreases substantially and smoothly.

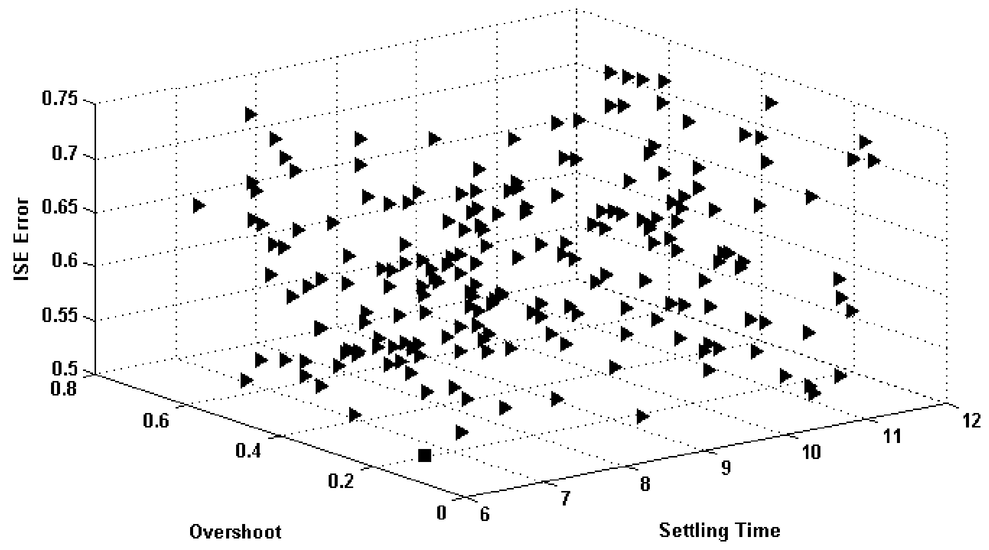

Figure 5.14. Multi-objective optimization Pareto-sets of the plant $G_1(s)$ for the MOBA

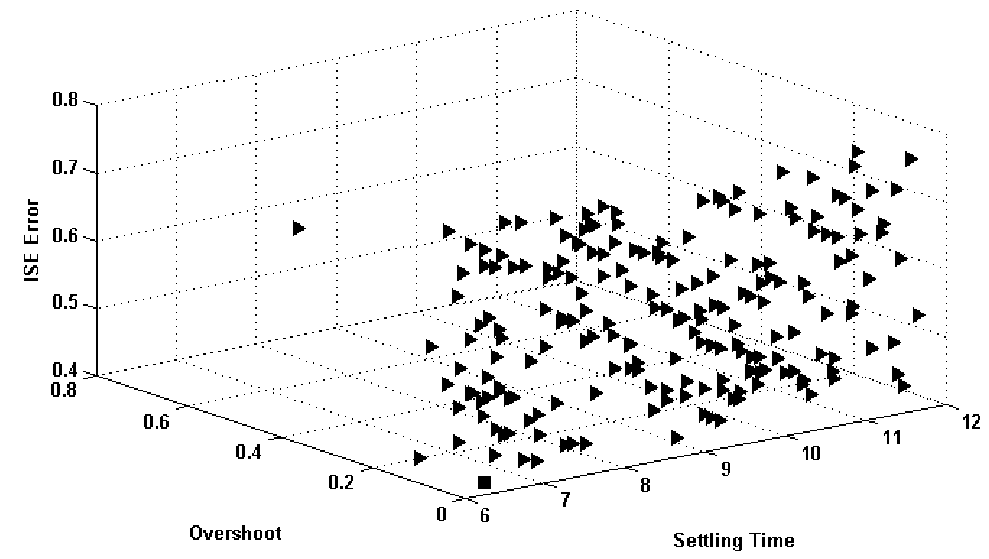

Figure 5.15. Multi-objective optimization Pareto-sets of the plant $G_1(s)$ for the MOABC

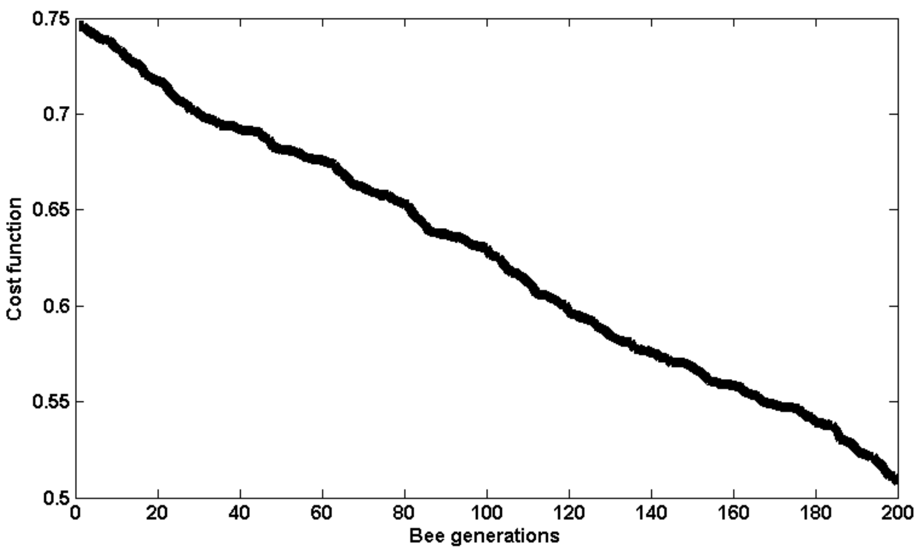

Figure 5.16. Convergence graph of the plant $G_1(s)$ in the MOBA method

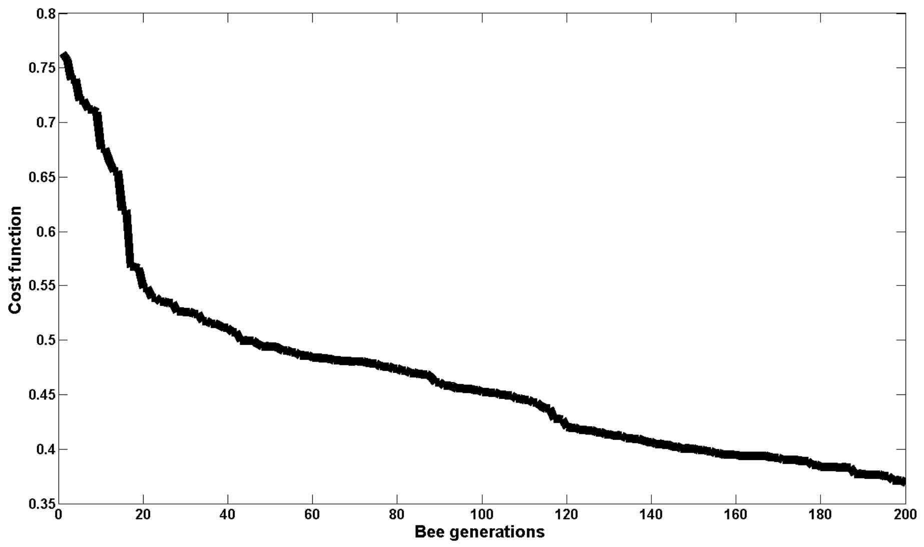

Figure 5.17. Convergence graph of the plant $G_1(s)$ in the MOABC method

As seen in Figures 5.12 and 5.13, the controlled systems show oscillations, especially much more in the plant $G_1(s)$. Sometimes oscillation effects stability of the controlled plants. It also causes undesirable situations. In order to cope with this problem generally some of the design specifications are modified in control system design. The smoother responses were achieved with slight concessions to the rise time. This time the vector of weights was defined as Φ = [0.000001 0.000001 1 0

0 0.0001 0]. Increasing the rise time caused longer settling time for the plant $G_1(s)$. Nevertheless, the controlled systems gave fast response without overshoot and oscillation as seen in Figure 5.18 and Figure 5.19. The obtained results are presented in Table 5.14.

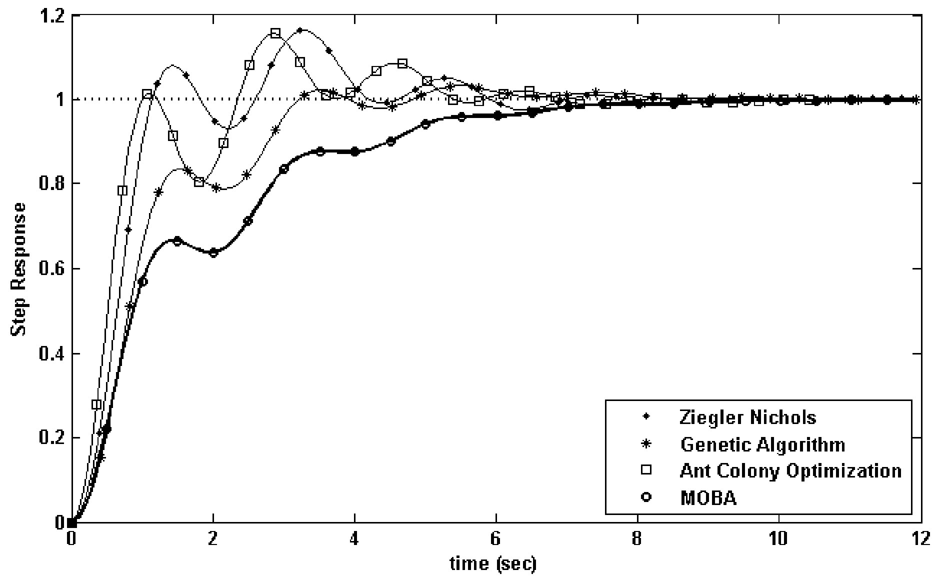

Figure 5.18. Step responses of the plant $G_1(s)$ with increasing rise time for the MOBA

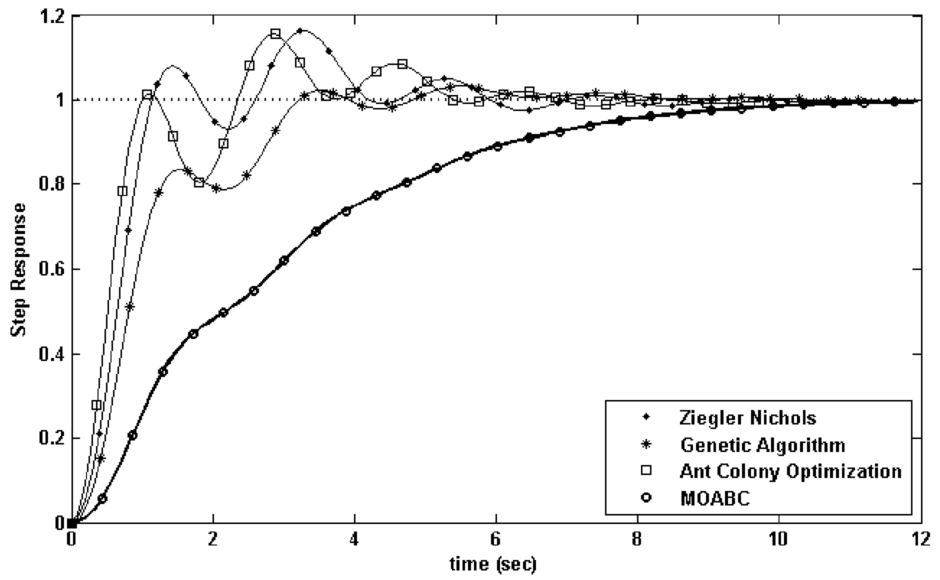

Figure 5.19. Step responses of the plant $G_1(s)$ with increasing rise time for the MOABC

Table 5.14. Simulation results of the proposed algorithm with increasing rise time

$G_1(s)$	Algorithm Parameters	PID Parameters	For ISE
BA	n=200; e=20; m=80; nep=60; nsp=40; ngh=6.0; sc=2.0 iter=2000 runtime=30	K_P	1.3104
		K_I	0.5825
		K_D	0.4597
		f_1:t_s	6.9200
		f_2:t_r	4.1637
		f_3:M_p	%0.0254
ABC	n=200 limit=100 iter=2000 runtime=30	K_P	0.6138
		K_I	0.3521
		K_D	0.1120
		f_1:t_s	9.4256
		f_2:t_r	5.6385
		f_3:M_p	%0.0887

5.2.2 Example 2: PID Controller Tuning for Fourth Order Linear Plant

For PID controller tuning, a fourth order plant is used as follows (Bagis, 2007):

$$G_2(s) = \frac{27}{(s+1)(s+3)^3} \tag{5.27}$$

The results were found by using 200 scout bees. For avoiding a similar particular solution, the initial populations were generated at random within the range *$0.0 \le K_P \le 4.0$, $0.0 \le K_I \le 4.0$, $0.0 \le K_D \le 4.0$*. The results in Table 5.15 show that the value of the maximum overshoot is quite smaller, nearly zero percent and the values of the rise time, the settling time for *ISE* error criteria obtained by the MOABC and the MOBA are much less than the values of the other methods. The results of the other methods in Table 5.15 were taken from existing literature (Bagis, 2007).

Furthermore, the step responses of $G_2(s)$ tested with the optimum values of the parameters K_P, K_I and K_D which are obtained by the MOABC and the MOBA are presented in Table 5.16 according to some error criteria. Table 5.16 shows the values of the parameters adopted for the ABC and the BA. The values were decided empirically for the ABC and the BA.

Table 5.15. Comparison of simulation results of the PID controlled systems (Hsiao et al., 2004)

$G_2(s)$	Algorithm Parameters	PID Parameters	Ziegler Nichols	Genetic Algorithm	Ant Colony Optimization	Proposed Algorithms
BA	n=200; e=20; m=80 nep=60; nsp=40 ngh=6.0 sc=2.0 iter=2000 runtime=30	K_P	3.072	1.772	2.058	2.2103
		K_I	2.272	1.061	1.137	1.1038
		K_D	1.038	0.772	0.746	1.2640
		f_1:t_s	5.1	2.91	4.34	3.8226
		f_2:t_r	0.7	1.2	0.971	0.8730
		f_3:M_p	%32.53	%1.17	%6.62	%0.0069
		f_6:ISE	0.66	0.7311	0.708	0.618
ABC	n=200 limit=100 iter=2000 runtime=30	K_P	3.072	1.772	2.058	2.2974
		K_I	2.272	1.061	1.137	1.1017
		K_D	1.038	0.772	0.746	1.2176
		f_1:t_s	5.1	2.91	4.34	3.9734
		f_2:t_r	0.7	1.2	0.971	0.8547
		f_3:M_p	%32.53	%1.17	%6.62	%0
		f_6:ISE	0.66	0.7311	0.708	0.514

Table 5.16. Simulation results of the proposed algorithms for different error criteria

$G_2(s)$	Algorithm Parameters	PID Parameters	For ISE	For IAE	For ITAE	For ITSE	For MSE
BA	n=200 e=20 m=80 nep=60 nsp=40 ngh=6.0 sc=2.0 iter=2000 runtime=30	K_P	2.2103	2.3414	2.0473	1.6778	2.4385
		K_I	1.1038	1.1708	1.0933	0.9626	1.1429
		K_D	1.2640	1.3930	0.9872	0.7340	1.5330
		f_1:t_s	3.8226	3.5923	3.8710	3.9130	4.7147
		f_2:t_r	0.8730	0.8027	0.9763	1.2478	0.7533
		f_3:M_p	%0.0069	%0.0326	%0	%0.000063	%0.0071
ABC	n=200 limit=100 iter=2000 runtime=30	K_P	2.2974	2.1449	2.0105	1.6602	2.4195
		K_I	1.1017	1.0811	1.0578	0.9633	1.1206
		K_D	1.2176	1.1892	1.0262	0.6928	1.4801
		f_1:t_s	3.9734	3.8912	3.9170	3.8488	4.8280
		f_2:t_r	0.8547	0.9111	1.0026	1.2461	0.77
		f_3:M_p	%0	%0.0031	%0.000051	%0	%0

The step responses of $G_2(s)$ plotted with the optimum values of the parameters K_P, K_I and K_D which are obtained by the MOBA are shown in Figures 5.20. Furthermore, The step responses of $G_2(s)$ plotted with the optimum values of the parameters K_P, K_I and K_D which are obtained by the MOABC are shown in Figures 5.21. Step response results of $G_2(s)$ process obtained by using Ziegler Nichols, genetic algorithm and ant colony optimization algorithm are represented for comparison purposes.

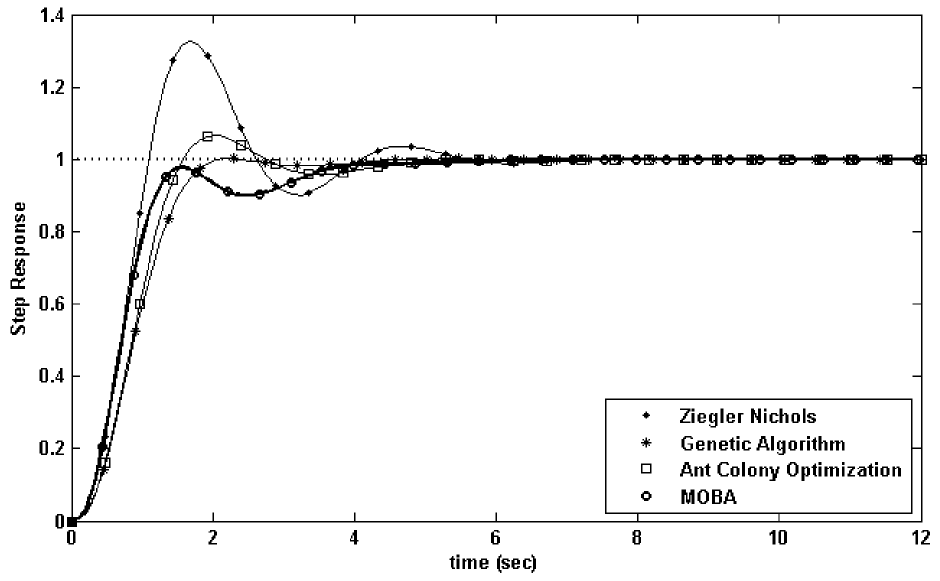

Figure 5.20. Comparison of step responses of the plant $G_2(s)$ for the MOBA

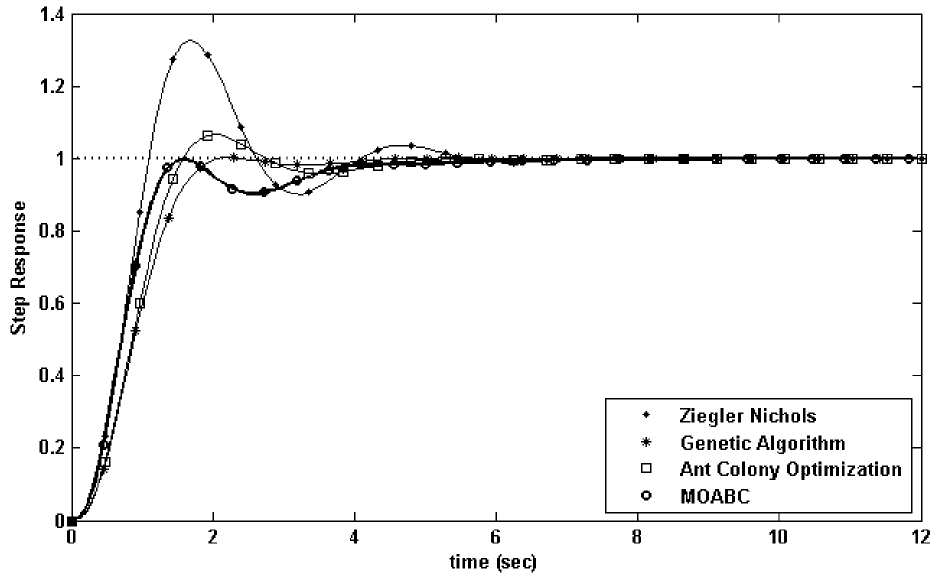

Figure 5.21. Comparison of step responses of the plant $G_2(s)$ for the MOABC

Figures 5.22 and Figures 5.23 illustrate the graphs of the obtained three-dimensional Pareto optimal fronts consisting of the settling time, overshoot and ISE error criteria for the step response of $G_2(s)$ process related with each transfer function. Thus, a well distributed set of non-dominated solutions along the Pareto-optimal front can be found. The MOABC gives better responses than those produced by using the other methods. Thus, it can be considered that the MOABC improves the optimal system performance of the PID controllers satisfactorily. Evaluation of the objective function on the above mentioned $G_2(s)$ plant is presented in Figures 5.24 and Figures 5.25. It is also observed that the objective function value decreases substantially and smoothly.

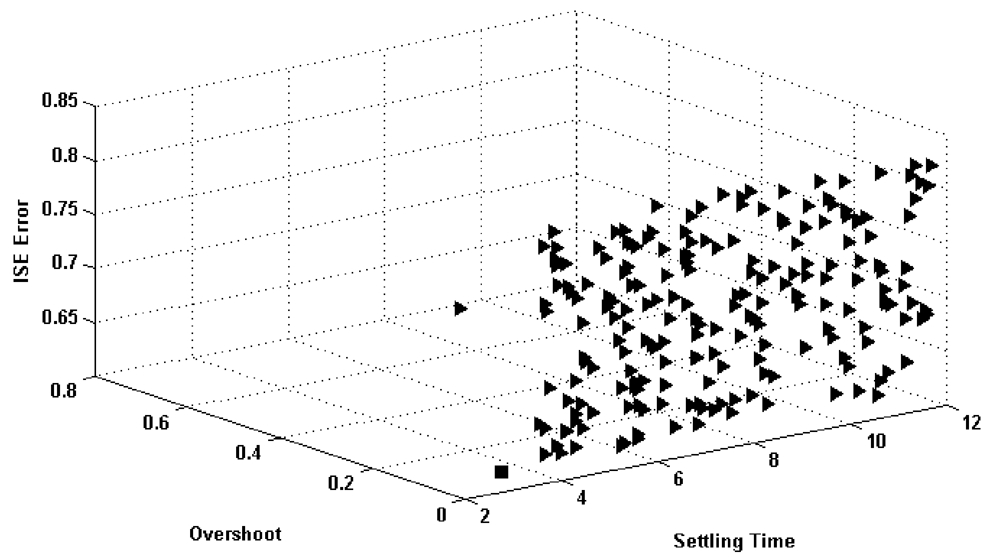

Figure 5.22. Multi-objective optimization Pareto-sets of the plant $G_2(s)$ for the MOBA

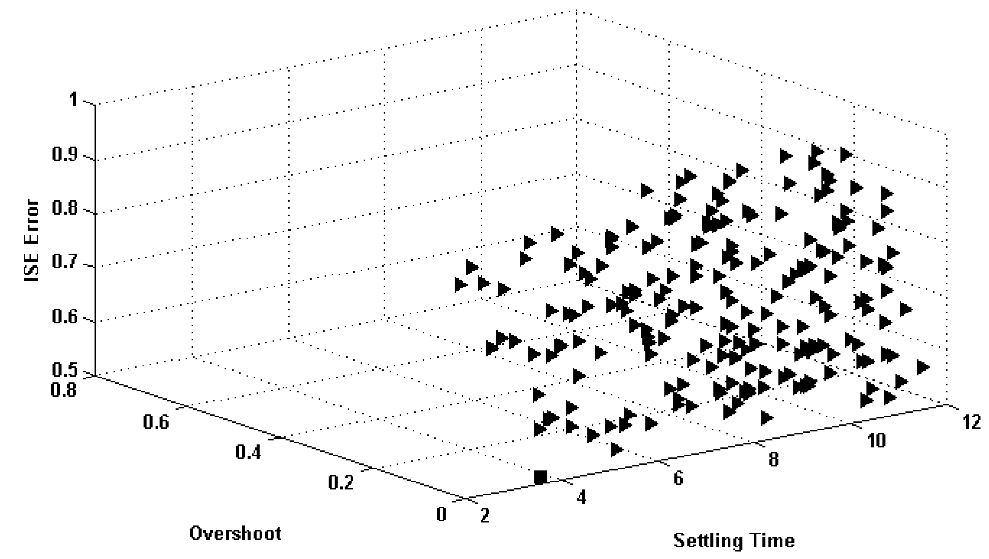

Figure 5.23. Multi-objective optimization Pareto-sets of the plant $G_2(s)$ for the MOABC

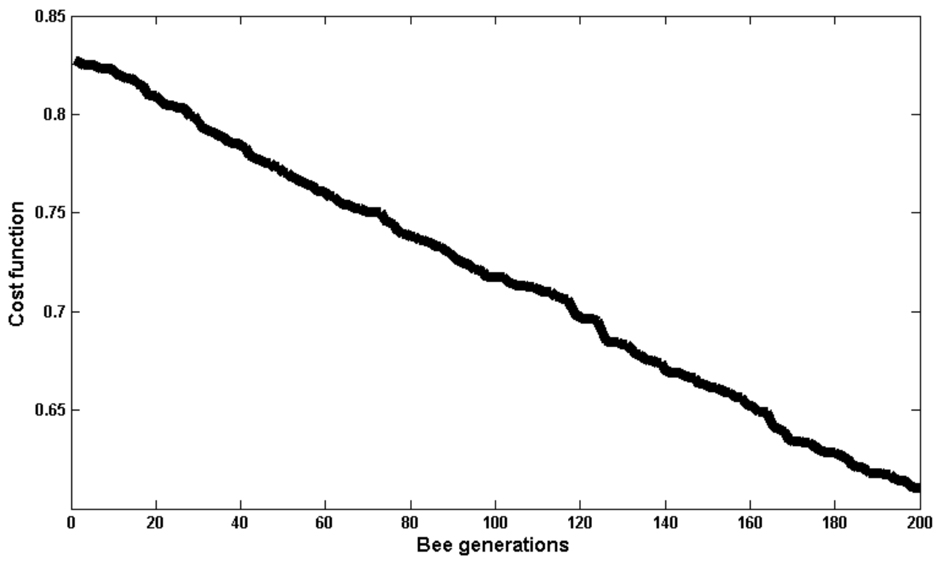

Figure 5.24. Convergence graph of the plant $G_2(s)$ in the MOBA method

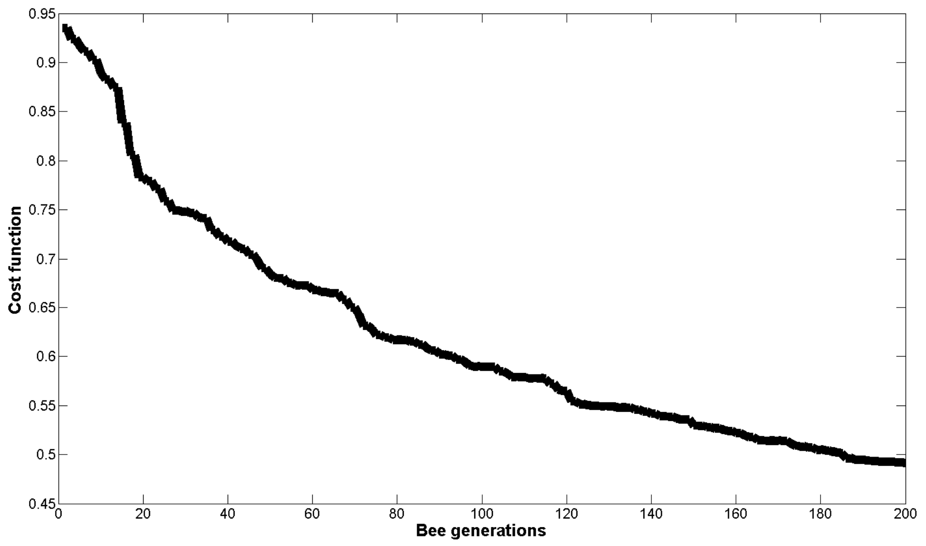

Figure 5.25. Convergence graph of the plant $G_2(s)$ in the MOABC method

As seen in Figures 5.20 and 5.21, the controlled systems show oscillations, especially much more in the plant $G_2(s)$. The smoother responses were achieved with slight concessions to the rise time. This time the vector of weights was defined as $\Phi = [0.000001 \quad 0.000001 \quad 1 \quad 0 \quad 0 \quad 0.0001 \quad 0]$. Increasing the rise time caused shorter settling time for the plant $G_2(s)$. Nevertheless, the controlled systems gave fast response without overshoot and oscillation as seen in Figure 5.26 and Figure 5.27. The obtained results are presented in Table 5.17.

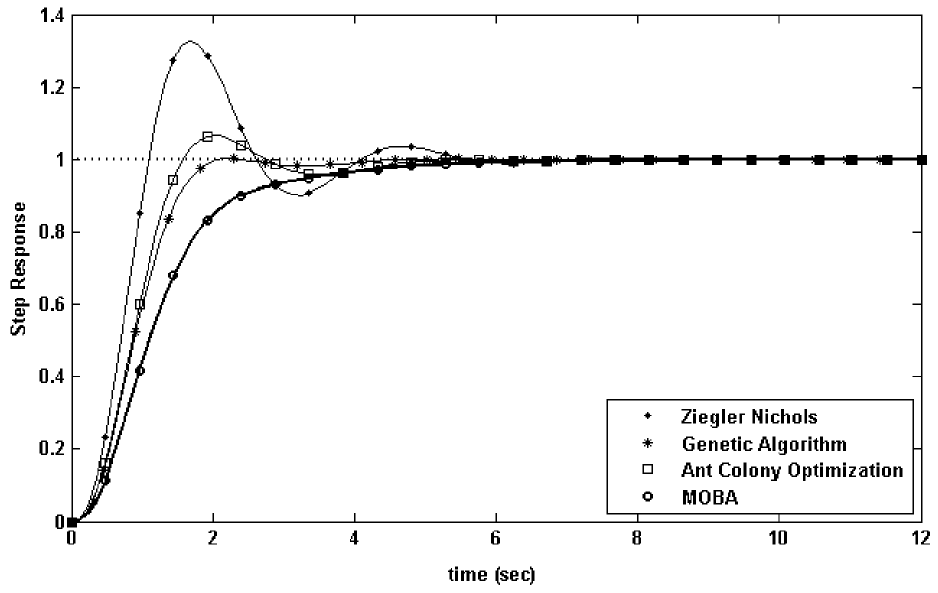

Figure 5.26. Step responses of the plant $G_2(s)$ with increasing rise time for the MOBA

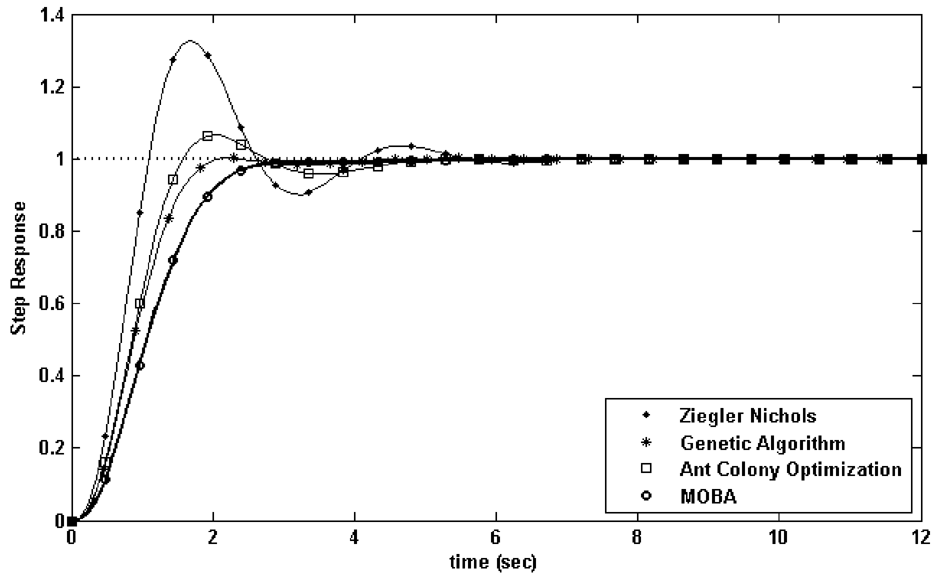

Figure 5.27. Step responses of the plant $G_2(s)$ with increasing rise time for the MOABC

Table 5.17. Simulation results of the proposed algorithm with increasing rise time

$G_2(s)$	Algorithm Parameters	PID Parameters	For ISE
BA	n=200; e=20; m=80; nep=60; nsp=40; ngh=6.0; sc=2.0 iter=2000 runtime=30	K_P	1.2835
		K_I	0.7422
		K_D	0.5432
		f_1:t_s	4.7018
		f_2:t_r	1.9513
		f_3:M_p	%0
ABC	n=200 limit=100 iter=2000 runtime=30	K_P	1.3927
		K_I	0.8561
		K_D	0.5236
		f_1:t_s	2.5920
		f_2:t_r	1.4961
		f_3:M_p	%0

5.2.3 Example 3: PID Controller Tuning for DC Motor Plant

For PID controller tuning, a DC motor plant is used as follows (Nasri et al., 2007):

$$G_3(s) = \frac{\Omega(s)}{U_a(s)} = \frac{0.1433}{5.2\times10^{-7}s^2 + 2.172\times10^{-4}s + 0.0227} \tag{5.28}$$

In this example, the objective function was composed of *f₁*, *f₂*, *f₃*, and *f*₆ which include the settling time, rise time, maximum overshoot and integral square error, respectively. The vector of weights was defined as Φ = [0.000001 0.0001 1 0 0 0.0001 0]. Simulations were carried out by using 200 scout bees and the initial populations were generated at random within the range $0.0 \le K_P \le 100.0$, $0.0 \le K_I \le 100.0$, $0.0 \le K_D \le 0.05$. Also, the initial populations were calculated between these ranges for genetic algorithm.

The plant given in Equation (5.28) was tested with a unit step input to show the effectiveness and performance of the proposed method. Three other approaches such as Ziegler-Nichols, genetic algorithm and ant colony algorithm were applied in order to make comparison and show the performance of the MOABC and the MOBA. The step response of the DC motor is depicted in Figure 5.28 and Figure 5.29. The obtained simulation results are given in Table 5.

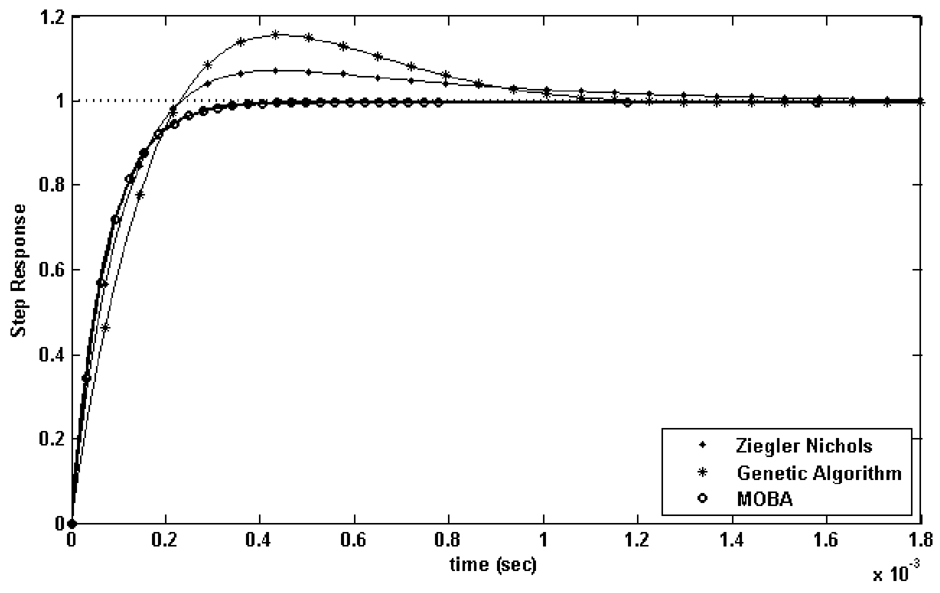

Figure 5.28. Comparison of step responses of the DC motor for the MOBA

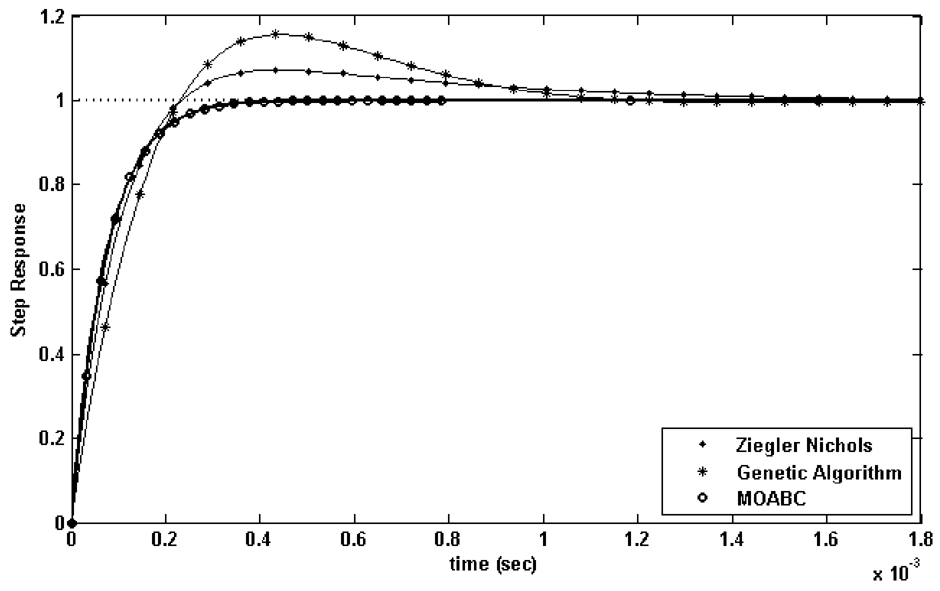

Figure 5.29. Comparison of step responses of the DC motor for the MOABC

Table 5.18. Simulation results of Motor Speed PID control (Nasri et al., 2007)

$G_3(s)$	Algorithm Parameters	PID Parameters	Ziegler Nichols	Genetic Algorithm	Proposed Algorithms
BA	n=200; e=20; m=80 nep=60; nsp=40 ngh=6.0 sc=2.0 iter=2000 runtime=30	K_P	70.556	93.1622	19.5190
		K_I	50	38.6225	52.9022
		K_D	0.039567	0.027836	0.04944
		f_1:t_s	$11x10^{-4}$	$9.83x10^{-4}$	$2.94x10^{-4}$
		f_2:t_r	$1.57x10^{-4}$	$1.71x10^{-4}$	$1.62x10^{-4}$
		f_3:M_p	%7.166	%15.609	%0
ABC	n=200 limit=100 iter=2000 runtime=30	K_P	70.556	93.1622	21.8463
		K_I	50	38.6225	48.4252
		K_D	0.039567	0.027836	0.0492
		f_1:t_s	$11x10^{-4}$	$9.83x10^{-4}$	$2.84x10^{-4}$
		f_2:t_r	$1.57x10^{-4}$	$1.71x10^{-4}$	$1.61x10^{-4}$
		f_3:M_p	%7.166	%15.609	%0

The simulation results on the plant and the average values of standard performance measures where the objective function depends on the standard performance measures such as rise time, settling time and maximum overshoot are summarized in Table 5.18. Figure 5.30 and Figure 5.31 presents the distribution of the non-dominated solutions in Pareto-optimal front. The convergence of the objective function is depicted in Figure 5.32 and Figure 5.33. It can be seen from the figure that the objective function value decreases considerably.

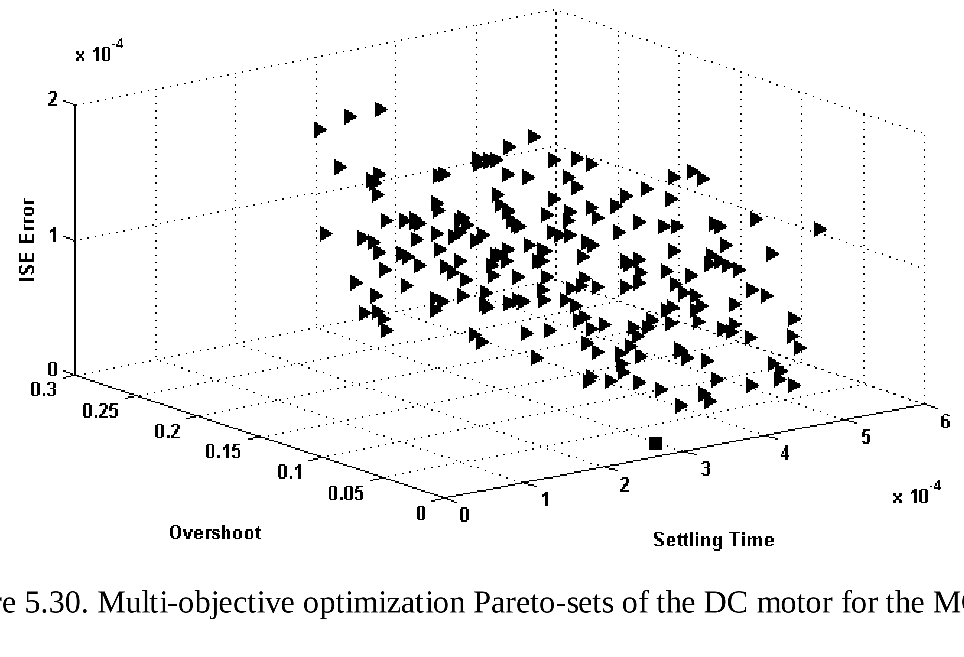

Figure 5.30. Multi-objective optimization Pareto-sets of the DC motor for the MOBA

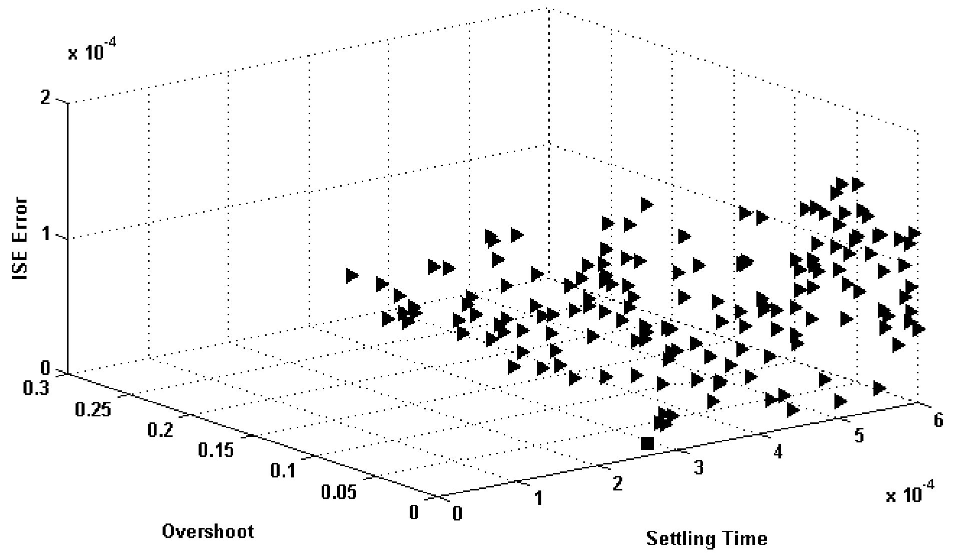

Figure 5.31. Multi-objective optimization Pareto-sets of the DC motor for the MOABC

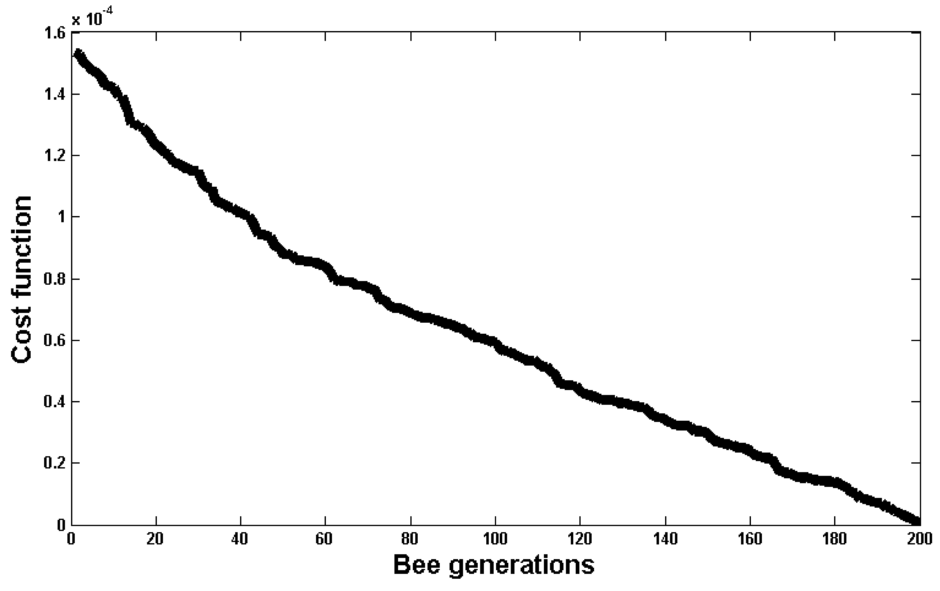

Figure 5.32. Convergence graph of the DC motor by using the MOBA method

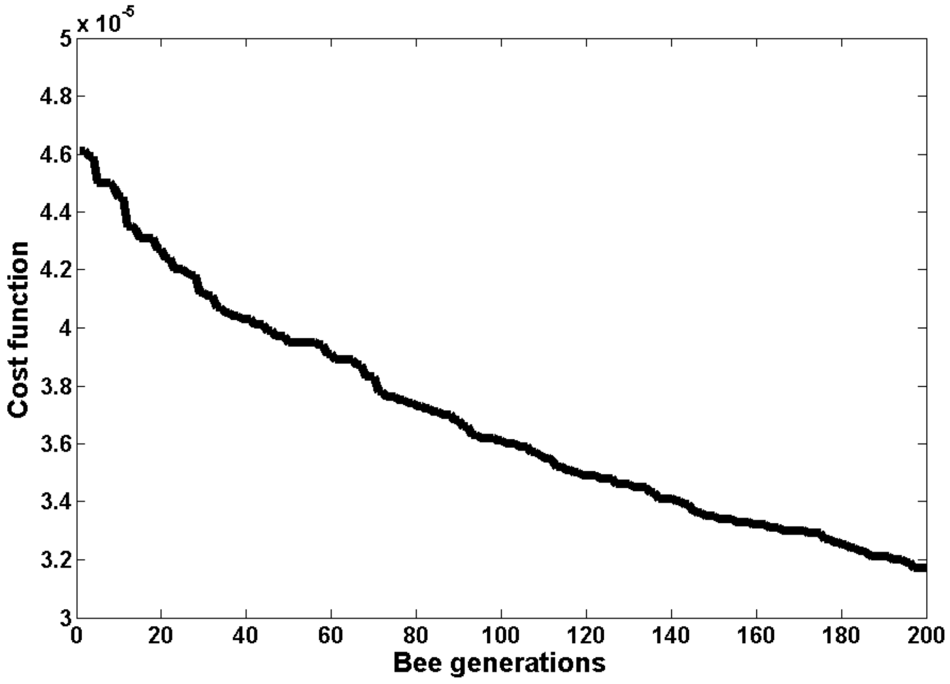

Figure 5.33. Convergence graph of the DC motor by using the MOABC method

5.2.4 PID Controller Design with Gaussian White Noise

In order to evaluate the effect of a noise disturbance, we have performed simulations where the PID controller parameters have been tested in the presence of Gaussian noise acting on the output of the system in Figure 5.34. The above PID controller design examples are tested for two different variances σ^2=0.0025 and σ^2=0.025.

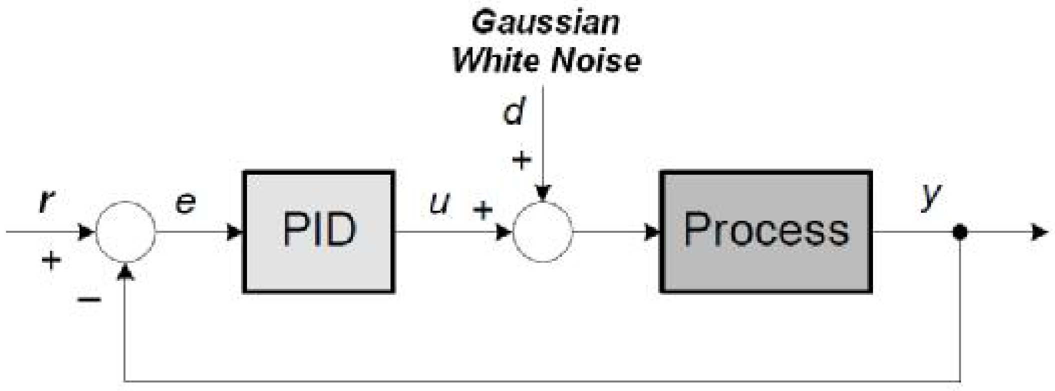

Figure 5.34. The PID controller in the closed-loop with Gaussian White Noise

The closed loop responses of the third order linear plant in the Example 1 for different variances of gaussian white noise are illustrated in Figure 5.35, 5.36 for the MOBA and in Figure 5.37, 5.38 for the MOABC.

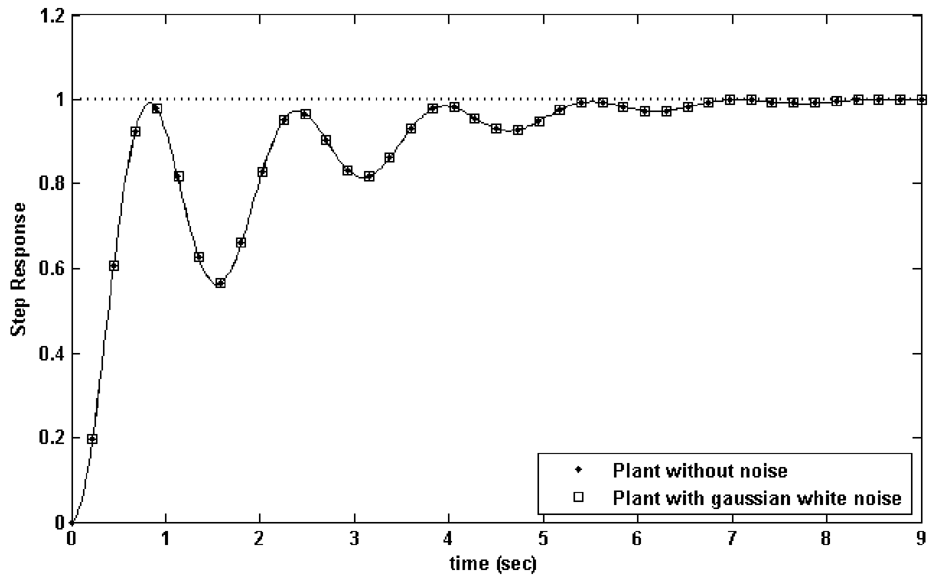

Figure 5.35. Comparison of Gaussian white noise step responses of the plant $G_1(s)$ for the MOBA for σ^2=0.0025

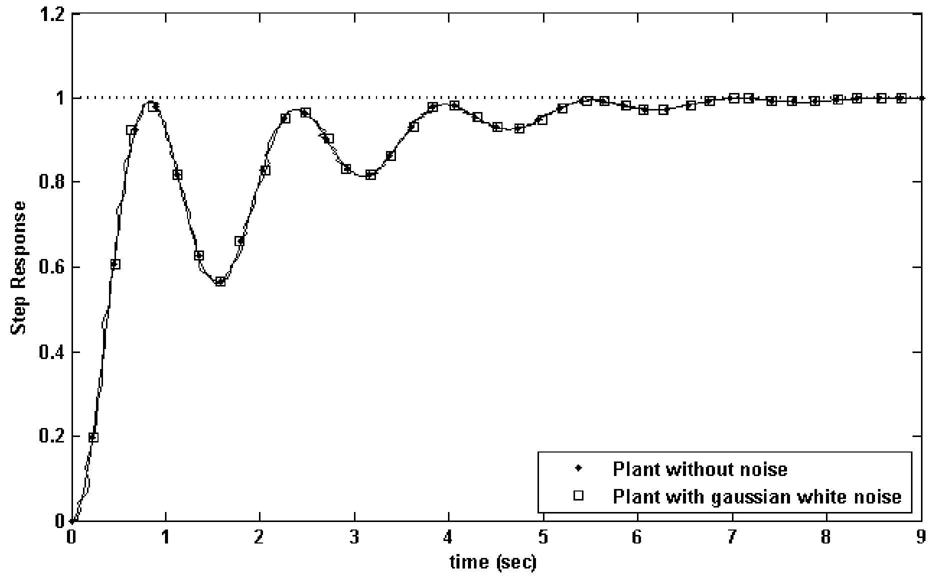

Figure 5.36. Comparison of Gaussian white noise step responses of the plant $G_1(s)$ for the MOBA for σ^2=0.025

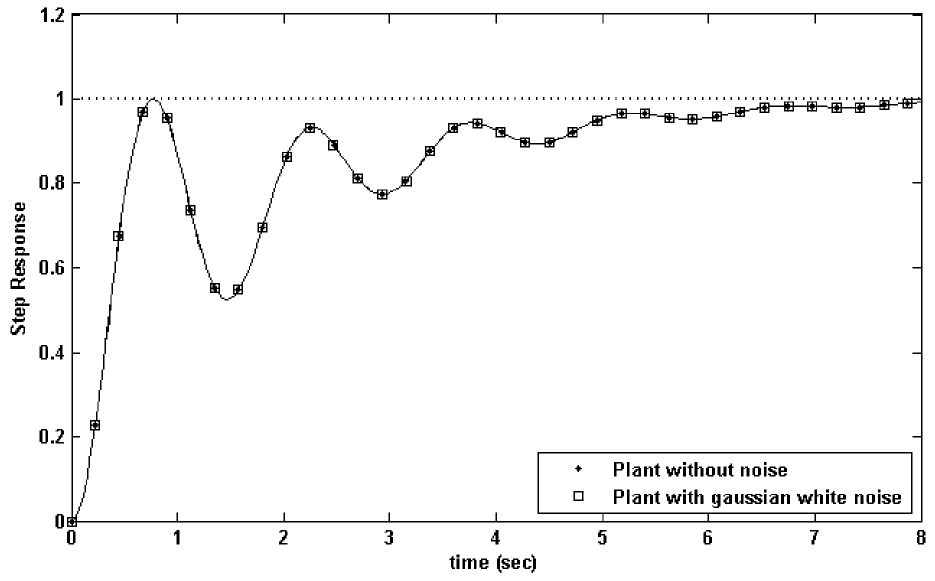

Figure 5.37. Comparison of Gaussian white noise step responses of the plant $G_1(s)$ for the MOABC for σ^2=0.0025

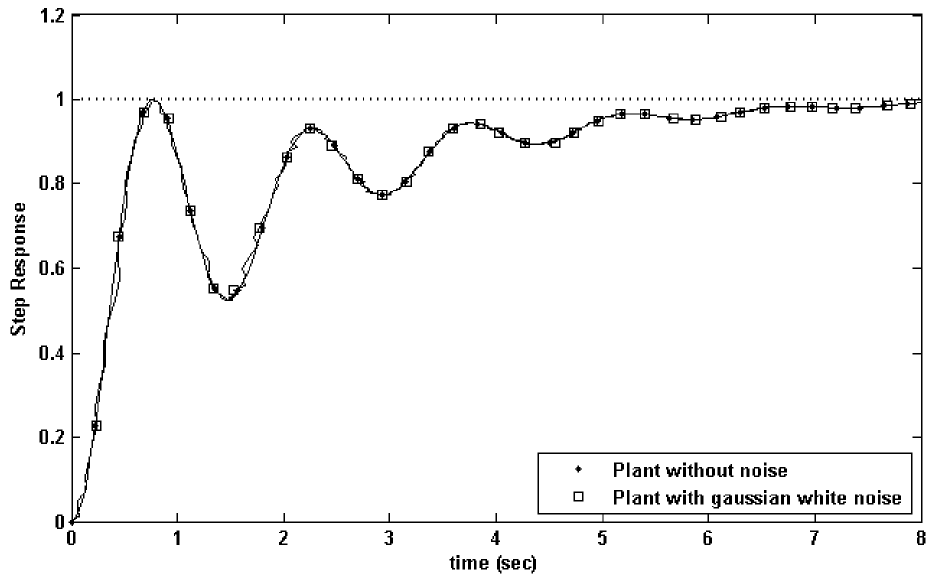

Figure 5.38. Comparison of Gaussian white noise step responses of the plant $G_1(s)$ for the MOABC for σ^2=0.025

The closed loop responses of fourth order linear plant in Example 2 for different variances of gaussian white noise are illustrated in Figure 5.39, 5.40 for the MOBA and in Figure 5.41, 5.42 for the MOABC.

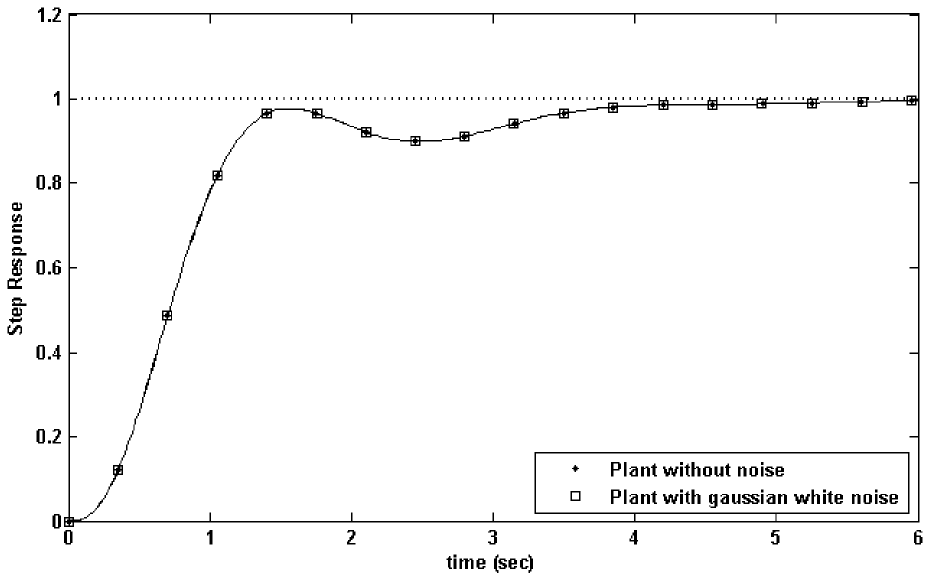

Figure 5.39. Comparison of Gaussian white noise step responses of the plant $G_2(s)$ for the MOBA for σ^2=0.0025

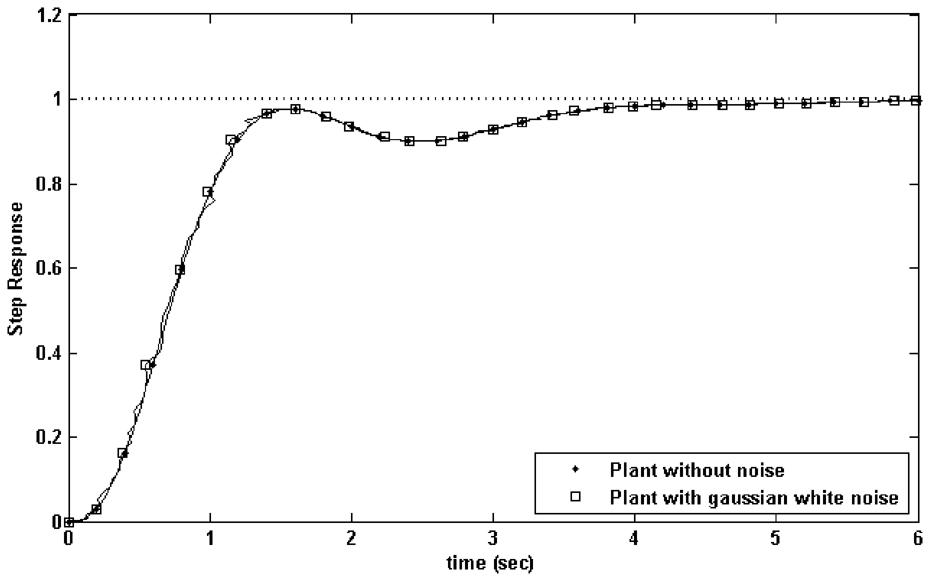

Figure 5.40. Comparison of Gaussian white noise step responses of the plant $G_2(s)$ for the MOBA for σ^2=0.025

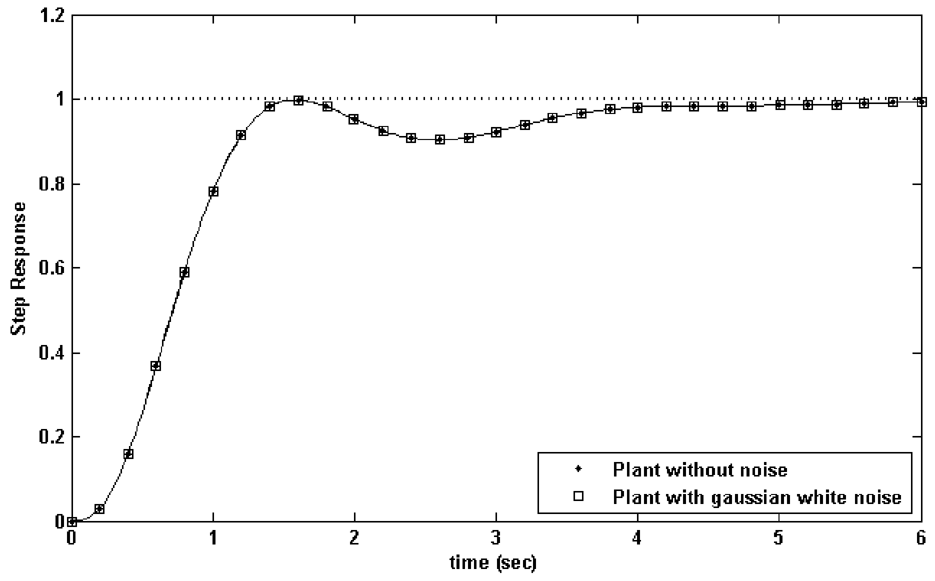

Figure 5.41. Comparison of Gaussian white noise step responses of the plant $G_2(s)$ for the MOABC for σ^2=0.0025

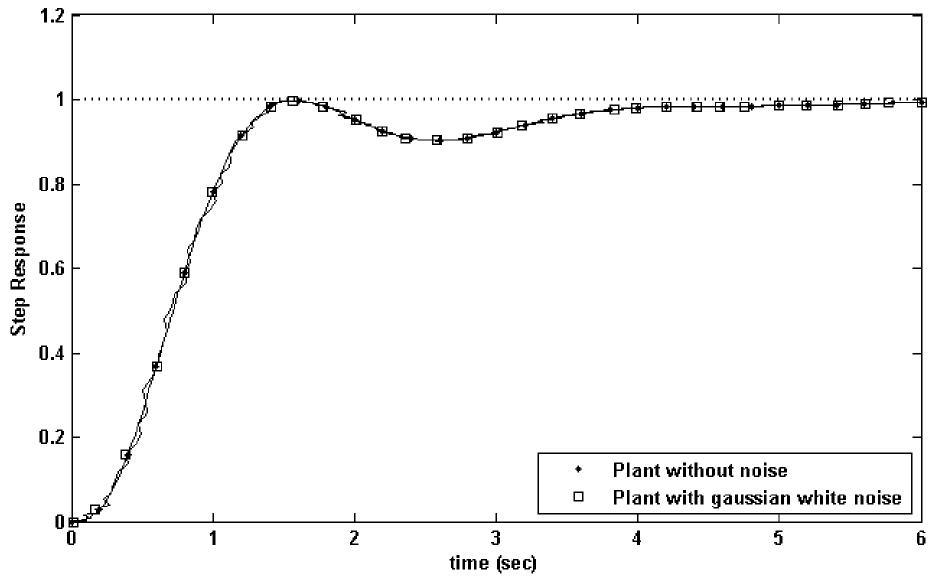

Figure 5.42. Comparison of Gaussian white noise step responses of the plant $G_2(s)$ for the MOABC for σ^2=0.025

The closed loop responses of second order linear DC Motor plant in Example 3 for different variances of gaussian white noise are illustrated in Figure 5.43, 5.44 for the MOBA and in Figure 5.45, 5.46 for the MOABC.

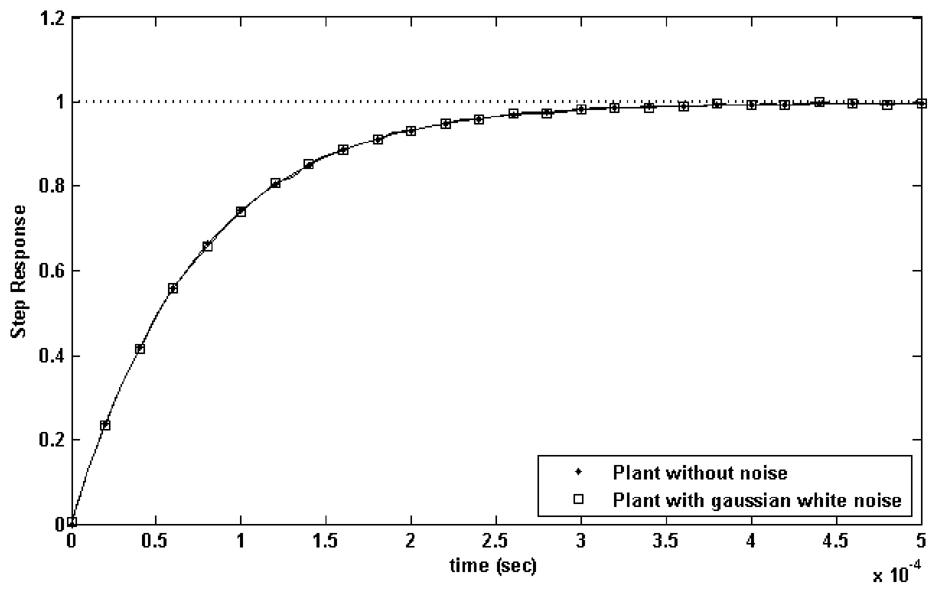

Figure 5.43. Comparison of Gaussian white noise step responses of the plant $G_3(s)$ for the MOBA for σ^2=0.0025

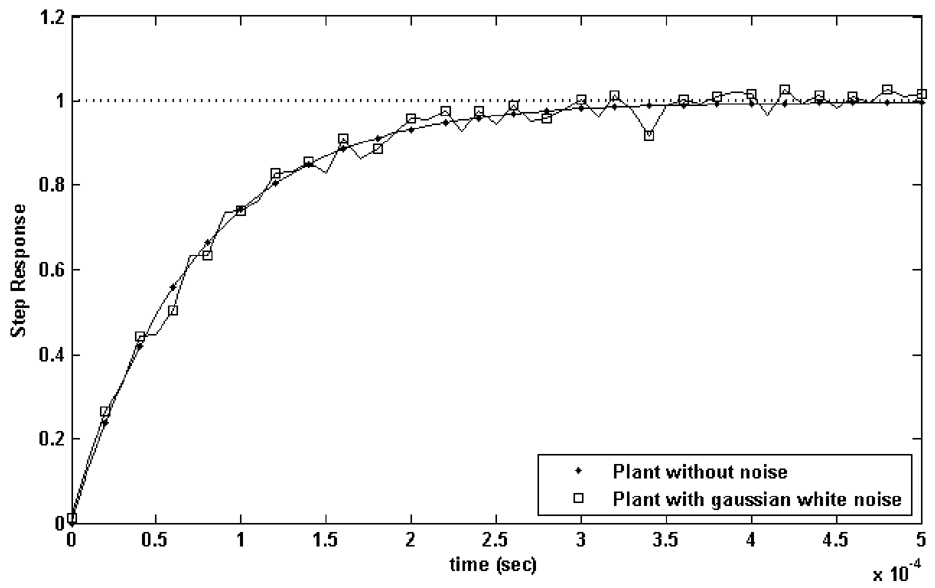

Figure 5.44. Comparison of Gaussian white noise step responses of the plant $G_3(s)$ for the MOBA for σ^2=0.025

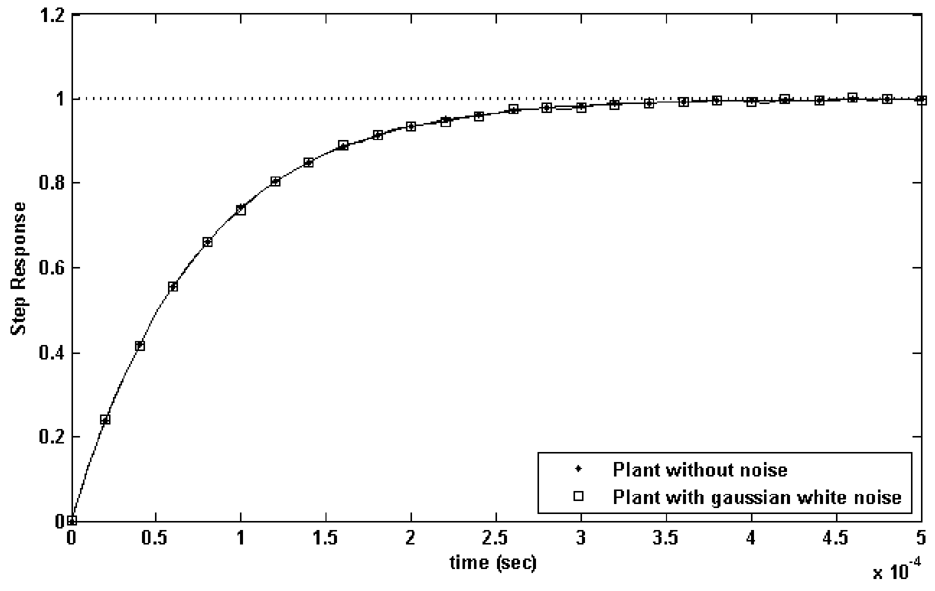

Figure 5.45. Comparison of Gaussian white noise step responses of the plant $G_3(s)$ for the MOABC for σ^2=0.0025

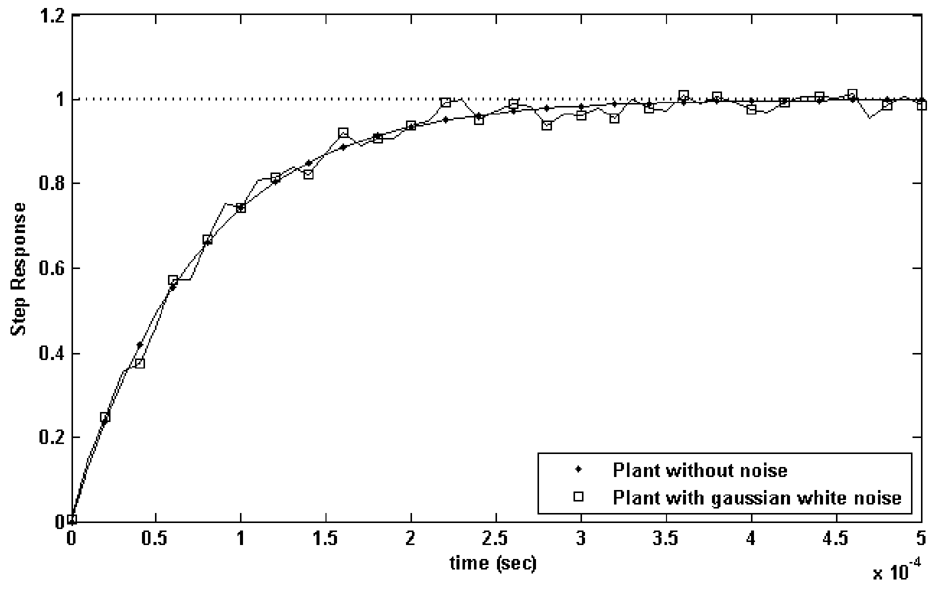

Figure 5.46. Comparison of Gaussian white noise step responses of the plant $G_3(s)$ for the MOABC for σ^2=0.025

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this paper, a novel intelligent tuning design method for determining the PID controller parameters based on Multi-Objective Bees Algorithm (MOBA) and Multi-Objective Artificial Bee Colony (MOABC) optimization is developed for getting good performances and tuning the Pareto-optimal PID parameters. The step response performance of the MOBA and the MOABC were tested with different order linear plants. It is well known that the MOBA and the MOABC have good results in solving numerical optimization problems. Thus, the effectiveness of the PID controller design using the MOBA and the MOABC were researched and was obtained a satisfactory performance. Although the ABC consists of less control parameters, it has a better tuning performance than the BA which consists of many control parameters. Also, the ABC is faster than the other. This study was also applied to tune PID controller parameters of the permanent magnet DC motors commonly used in industry and compared with some existing methods. The simulation results show that the new PID control tuning method using the MOBA and the MOABC achieve minimum overshoot and optimal or near optimal system performance. Due to the fact that some stability criteria are taken into account in the control system design, the proposed method thus can be regarded as a general controller design method that can be applied to a wide class of linear plants.

The ABC algorithm has shown to be versatile when applied to parameter estimation without requiring a detailed mathematical representation of the identification problem. The unit step and sinusoidal response performance of the ABC algorithm is tested with several order linear plants for system identification. The proposed method is also applied to estimate parameters of the permanent magnet DC motors commonly used in industry. The proposed method is flexible and applicable in a wide range of optimization and identification problems. The simulation results show that the proposed method achieve minimum tracking error and estimate the parameters values with a high accuracy.

Research was conducted to study the effects of using the honey bee algorithm as a tool for PID tuning and system identification. From the results presented in the study it was shown that the honey bee tuning yielded improved responses and can be applied to different process models encountered in the process control industry.

6.2 Future Work

Proposed method would be beneficial to discuss some alternative ways that could further improve the work with reference to the methodologies applied in this research. The work done in this thesis was based on having PID control structure. Since the real system is nonlinear in industrial control system applications, it would be interesting to apply nonlinear controllers such as sliding mode control algorithms. In the proposed design, off-line simulation was completely used. Our proposed method can be extended to an on-line controlling for any industrial control system applications.

Proposed method would be useful to use a more complex plant to prove the effectiveness of the methods under more general conditions. Testing the application for high order linear plants, plants that can be unstable for certain values of parameters, intrinsic nonlinear unstable plants would be certainly interesting for future work.

REFERENCES

AFSHAR, A., HADDAD, O.B., MARINO, M.A., ADAMS, B.J., 2007. Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. J. Franklin Institute, (344):452-462.

ANG, K.H., CHONG, G.C.Y., and LI, Y., 2005. PID Control System Analysis, Design, and Technology. IEEE Transactions on Control Systems Technology, 13(4):559–576.

ASTROM, K.J., HAGGLUND, T., 1984. Automatic tuning of simple regulators with specification on phase and amplitude margins. Automatica, 20:645-651.

ASTROM, K.J., HAGGLUND, T., 2004. Revisiting the Ziegler-Nichols Step Response method for PID control. Journal of Process Control, 14:635-650.

ASTROM, K.J., WITTENMARK, B., 1997. Computer controlled systems - theory and design (3rd Edition). Prentice Hall, 555p.

BAGIS, A., 2007. Determination of the PID controller parameters by modified genetic algorithm for improved performance. Journal of Information Science and Engineering, 23:1469-1480.

BASTURK, B., KARABOGA, D., 2006. An artificial bee colony (ABC) algorithm for numerical function optimization. In: IEEE Swarm Intelligence Symposium 2006, May 12-14, Indianapolis, IN, USA.

BASTURK, B., KARABOGA, D., 2009a. Solving integer programming problems by using Artificial Bee Colony Algorithm. In: SERRA, R., CUCCHIARA, R., (Eds.), Springer-Verlag, Berlin Heidelberg, 355-364.

BASTURK, B., KARABOGA, D., 2009b. Parameter tuning for the artificial Bee Colony Algorithm. In: NGUYEN, N.T., KOWALCZYK, R., and CHEN, S., Springer-Verlag, Berlin Heidelberg, 608-619.

BASTURK, B., KARABOGA, D., 2010. A modified Artificial Bee Colony algorithm for real-parameter optimization. Information science, doi:10.1016 / j.ins.2010.07.015.

CAMAZINE, S., SNEYD, J., 1991. A model of collective nectar source selection by honey bees: self-organization through simple rules. Journal of Theoretical Biology, 149:547–571.

COHEN, G.H., COON, G.A., 1953. Theoretical consideration of related control. Trans ASME. 75: 827-834.

EYKHOFF, P., 1974. System Identification: Parameter and State Estimation. Wiley, Chichester, UK, 555p.

HANG, C.C., ASTROM, K.J., and HO, W.K., 1991, Refinements of the Ziegler–Nichols Tuning Formula," IEEE Proceedings D Control Theory and Applications, 138(2)111–118.

HAUNG, H.P., CHEN, C.C., 1996. Control-system synthesis for open-loop unstable process with time delay. IEEE Proceedings Control Theory and Applications, 144(4):334-345.

HSIA, T.C., 1977. System Identification, Lexington Books, Lexington.

HSIAO, Y.T., CHUANG, C.L., and CHIEN, C.C., 2004. Ant colony optimization for designing of PID controllers. In: Proceedings of the 2004 IEEE international symposium on computer aided control systems design, Taipei, Taiwan, 321-326.

JURY, E.I., BLANCHARD, J., 1961. A stability test for linear discrete-time systems in table forms. Proc. IRE 49(1961):1947-1948.

KANG, F., LI, J., MA, Z., LI, H., 2011. Artificial Bee Colony Algorithm with Local Search for Numerical Optimization. Journal of Software, 3(6):490-497.

KARABOGA, D., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 39:459-471.

KARABOGA, D., BASTURK, B., 2007. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. Springer-Verlag, IFSA 2007, LNAI 4529, 789–798.

KARABOGA, D., BASTURK, B., 2009. A comparative study of Artificial Bee Colony algorithm. Applied Mathematics and Computation, 214:108-132.

KRUSIENSKI, D.J., JENKINS, W.K., 2005. Design and Performance of Adaptive Systems Based on Structured Stochastic Optimization Strategies. IEEE Circuits and Systems Magazine, 5:8-20.

KWOK, D.P., LEUNG, T.P., and SHENG, F., 1993. Genetic algorithms for optimal dynamic control of robot arms. Proceedings of the International Conference on Industrial Electronics, Control and Instrumentation. November 15-19, San Francisco, CA, 380-385.

LIPTÁK, B.G., 1995. Process Control Instrument Engineers Handbook (3rd Edition). CRC Press, New York, 1550p.

LIN, M.G., LAKSHMINARAYANAN S., RANGAIAH, G.P., 2008. A comparative study of recent/popular PID tuning rules for stable, first-order plus dead time, single-input single-output processes. Industrial and Engineering Chemistry Research, 47(2):344–368.

LIU, G.P., YANG, J.B., and WHIDBORNE, J.F., 2003. Multiobjective Optimisation and Control. Research Studies Press Ltd, Baldock, England.

LJUNG, L., 1999. System Identification: theory for the user (2nd Edition). Prentice Hall, Sweden, 609p.

LU, H.C., CHANG, J.C., and YEH, M.F., 2007. Design and analysis of direct-action CMAC PID controller. Neurocomputing, 70:2615–2625.

MOORE, P.W., VENAYAGAMOORTHY, G.K., 2006. Evolving Digital Circuits using Hybrid Particle Swarm Optimization and Differential Evolution. International Journal of Neural Systems, 16:1-15.

NAKRANI, S., TOVEY, C., 2004. On Honey Bees and Dynamic Allocation in Internet Hosting Centers. Adaptive Behavior, 12(3-4):223-240.

NASRI, M., NEZANABADI-POUR H., and MAGHFOORI, M., 2007. A PSO-Based optimum design of PID controller for a linear brushless DC motor. Proceedings of world academy of science, engineering and technology 26:211-215.

NGATCHOU, P., ZAREI, A., EL-SHARKAWI M.A. 2005. Pareto multi objective optimization. In: Proceedings of the 13th international conference on intelligent systems applications to power system, US, 84-91.

NISE, N.S., 2004. Control System Engineering (4th Edition). Wiley, 983p.

OGATA, K., 1995. Discrete-time control systems (2nd Edition). Prentice Hall, 745p.

ONG, C.M., 1998. Dynamic simulation of electric machinery. Prentice-Hall Inc, New Jersey.

PAOR, D.E., O'MALLEY, M., 1989. Controllers of Ziegler-Nichols type for unstable process with time delay. International Journal of Control, 49:1273-1284.

PHAM D.T., CASTELLANI M., and GHANBARZADEH A., 2007. Preliminary design using the bees algorithm. In: Proceedings of the eighth international conference on laser metrology, CMM and machine tool performance, LAMDAMAP, Cardiff, UK, 420-429.

PHAM D.T., GHANBARZADEH A., 2007. Multi-objective optimization using the bees algorithm. In: proceedings of the 3rd international virtual conference on intelligent production machines and systems (IPROMS 2007), Whittles, Dunbeath, Scotland.

PHAM D.T., GHANBARZADEH A., KOC E., and OTRI S., 2006a. Application of the bees algorithm to the training of radial basis function networks for control chart pattern recognition. In: Proceedings of the 5th CIRP international seminar on intelligent computation in manufacturing engineering, Ischia, Italy, 711-716.

PHAM D.T., GHANBARZADEH A., KOC E., OTRI S., RAHIM S., and ZAIDI M., 2006b. The bees algorithm, a novel tool for complex optimization problems. In: Proceedings of the 2nd international virtual conference on intelligent production machines and systems, Cardiff, UK, 454-459.

PHAM D.T., OTRI S., and KOC E., 2006c. Application of the bees algorithm to the training of learning vector quantisation networks for control chart pattern recognition. In: Proceedings of information and communication technologies, Syria, 1624-1629.

POULIN, E., POMERLEAU, A., 1996. PID tuning for integrating and unstable processes. IEEE Proceedings Control Theory and Applications, 143:429-435.

QUIJANO, N., PASSINO K.M., 2007a. Honey Bee Social Foraging Algorithms for Resource Allocation, Part I: Algorithm and Theory. Submitted, American Control Conference.

QUIJANO, N., PASSINO K.M., 2007b. Honey Bee Social Foraging Algorithms for Resource Allocation, Part II: Application. Submitted, American Control Conference.

SEBORG, D.E., EDGAR, T.F., and MELLICHAMP, D.A., 2004. Process dynamics and control. Wiley, New York, 713p.

SEELEY, T.D., 1995. The Wisdom of the Hive. Harvard University Press, Cambridge, MA.

SHAFIEI, M., BOZORG HADDAD, O., and AFSHAR, A., 2007. Evaluation of Performance of Improved HBMO Algorithm in Optimization of Reservoir Operation. 7th River Engineering International Conference, Ahvaz, Iran.

SHAMSUZZOHA, M., SKOGESTAD, S., 2010. The Setpoint Overshoot Method: A Simple and Fast Closed-Loop Approach for PID Tuning. Journal of Process Control. (20):1220-1234.

TEODOROVIC, D., DELL'ORCO, M., 2005. Bee colony optimization - A cooperative learning approach to complex transportation problems. In Advanced OR and AI Methods in Transportation. Proceedings of the 10th Meeting of the EURO Working Group on Transportation, Poznan, Poland, 51–60.

TERESHKO, V., 2000. Reaction-diffusion model of a honeybee colony’s foraging behaviour. SCHOENAUER, M., et al, Eds., Parallel Problem Solving from Nature VI (Lecture Notes in Computer Science, Vol. 1917) Springer-Verlag: Berlin, 807-816.

VAN DEN BOSCH, P.P.J., VAN DER KLAUW, AC., 1994. Stochastische system theorie: modeling, Identification and simulation of dynamical systems. Eindhoven University of Technology, CRC press, 195p.

VENKATASHANKAR, V., CHIDAMBARAM, M., 1994. Design of P and PI controllers for unstable process with time delay. International Journal of Control, 60:1367-144.

YANG, X.S., 2005. Engineering Optimization via Nature-Inspired Virtual Bee Algorithms. IWINAC 2005, LNCS 3562, 17-323.

ZHUANG, M., ATHERTON, D.P., 1993, Automatic tuning of optimum PID controllers. IEE Proceedings-D, Vol. 140(3):216-224.

ZIEGLER, J.G., NICHOLS, N.B., 1942. Optimum Settings for Automatic Controllers. Transaction American Society of Mechanical Engineering, 64:759–768.

CIRRICULUM VITAE

Özden ERÇİN was born in 1982. He received his BSc degrees in Electrical and Electronics Engineering from Mersin University in 2004. His research interests are optimal control systems, industrial control systems and programmable logic controller. He have amassed over 5 years of significant, progressive experience in Control & Instrumentation Engineering within the oil and gas industries. Currently he is working for Botas International Limited as Instrument and Control Engineer. He is married and has a daughter.