ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

FUZZY-PSO CONTROL OF LINEAR AND NONLINEAR SYSTEMS

M.Sc. THESIS

Tolga KAYA

Department of Control and Automation Engineering

Control and Automation Engineering Programme

JANUARY 2014

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

FUZZY-PSO CONTROL OF LINEAR AND NONLINEAR SYSTEMS

M.Sc. THESIS

Tolga KAYA
(504081142)

Department of Control and Automation Engineering

Control and Automation Engineering Programme

Thesis Advisor: Asst. Prof. Dr. Gülay ÖKE

JANUARY 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DOĞRUSAL VE DOĞRUSAL OLMAYAN SİSTEMLERDE BULANIK-SÜRÜ PARÇACIĞI OPTİMİZASYON YAKLAŞIMI İLE KONTROL

YÜKSEK LİSANS TEZİ

Tolga KAYA
(504081142)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Yrd. Doç. Dr. Gülay ÖKE

OCAK 2014

Tolga Kaya, a **M.Sc.** student of ITU **Graduate School of Science Engineering and Technology** 504081142, successfully defended the **thesis** entitled "**FUZZY-PSO CONTROL OF LINEAR AND NONLINEAR SYSTEMS**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst. Prof. Dr. Gülay ÖKE** İstanbul Technical University	.....
Jury Members :	**Prof.Dr. İbrahim EKSİN** İstanbul Technical University	.....
	Asst. Prof. Dr. Uğur YILDIRAN Yeditepe University	.....

Date of Submission : 10 January 2014
Date of Defense : 24 January 2014

To my family,

FOREWORD

Special thanks for my supervisor Asst. Prof. Dr. Gülay ÖKE during thesis preparation in every aspect about leading to reach my goals, giving advises about preperation. In addition to that special thanks to Prof. Dr. İbrahim EKSİN about gaining different apects of the topic and making me the go further every time.

I would expect that this thesis can be very helpful about studying on Fuzzy Control Systems with tools such as Particle Swarm Optimization on nonlinear systems.

January 2014

Tolga KAYA
Systems Control Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

PID	**:** Proportional Integral Derivative Controller
PSO	**:** Particle Swarm Optimization
ZN	**:** Ziegler Nichols
TS	**:** Takagi-Sugeno Rule Table
IMC	**:** Internal Mode Control
FC	**:** Fuzzy Control
GA	**:** Genetic Algorithm
IAE	**:** Integral Absolute Error
ISE	**:** Integral Squared Error
FOPDT	**:** First Order Plus Dead Time System
SOPDT	**:** Second Order Plus Dead Time System
LSFM	**:** Least Square Fitting Method

LIST OF TABLES

Page

LIST OF FIGURES

Page

FUZZY-PSO CONTROL OF LINEAR AND NONLINEAR SYSTEMS

SUMMARY

The goal of the thesis is to introduce a new global optimization method called particle swarm optimization that is implemented via MATLAB to use to find the optimal parameters for PID coefficients and Takagi-Sugeno rule base's crisp values in order to control linear and nonlinear systems within specified operating conditions. The most important advantages of particle swarm optimization algorithm is that it requires less number of iterations and it enables us to deal with a few lines of computer codes in a cheapest manner rather than other optimization methods such as genetic algorithm. It requires only primitive mathematical operators in terms of both necessity of more available memory and speed. Particle swarm optimization method has been successfully applied to the design of coupled tanks system control with meaningful time domain criteria.

Since the coupled tank system to be controlled is nonlinear and time varying charecteristic, it is almost not possible to find one set of parameters that satisfy for all operating conditions. Therefore some predetermined operating points have been chosen and find out the optimal control parameters' values for the operating points while keeping Takagi-Sugeno crisps values constant for all operating points within the different ranges. Different functions are calculated for each controller parameters within different operating points based on the referenced height of tank two as an input value to the coupled tank system by using the predetermined points and least curve-fitting algorithm. It has been observed that these functions, which derive fuzzy controller parameters, have achieved very satisfactorly systems responses.

This thesis is mainly composed of three parts. First part is to introduce the classical tuning methods, fuzzy control structure and particle swarm optimization algorithm. Ziegler Nichols, Set Point Weighting, Cohen Coon and lastly Internal Model Control methods have been reviewed as classical tuning methods. The focus in this thesis is to control linear and nonlinear system within specified operating conditions by fuzzy PID controller with particle swarm optimization technique as an optimization tool. The evaluation of particle swarm optimization algortihm is also reviewed and new proposed method, which is called improved particle swarm optimization, has been tested on different benchmark functions. At the end of testing of the benchmark functions, it is decided to use improved particle swarm optimization method due to its performance on the convergence rate and convergence precision compared to standard particle swarm optimization. The integration of fuzzy system to PID controller has been also studied and complete architecture of fuzzy PID controller has been designed to engage with improved particle swarm optimization as an optimization tool.

Second part of this thesis is a preliminary study for the third part of the study. The aim is to implement improved particle swarm optimization technique as an optimization tool with fuzzy structured PID controller on different type of the

systems such as first order plus dead time system, second order plus dead time system and finally second order plus dead time oscilattory process model. The parameters of PID and the crisp values of the Takagi-Sugeno rule have been tuned offline for minimizing the performance criteria given as integral absolute error. The performance results in terms of maximum overshoot, settling time and rise time of the proposed approach have been depicted. By the guidance of the work on those systems and motivated by the good performances achieved, it is decided to implement the proposed method on nonlinear couple tank system to understand the applicability of the proposed study to control the water level on tank two which is the complete focus on the third part of the study.

In the third part of the study, fuzzy PID controller with particle swarm optimization technique as an optimization tool has been applied to nonlinear and time varying characteristics of the coupled tank water system since nonlinear and time varying systems have been encountered almost all areas especially in process industries. The water levels between different ranges are chosen respectively as a three typical operating regions of second tank and input space is divided into three fuzzy subspaces based on operating regions. Fuzzy PID parameters have been calculated online by proposed method despite of the fact that Takagi Sugeno crisp values have been calculated offline and stored before calculating PID parameters for the three operating regions. We can generalize that Takagi-Sugeno crisp values, which are structural parameters, are determined offline design while the tuning parameters are calculated during online adjustment of fuzzy PID controller to enhance the process performance, as well as to accommodate the adaptive capability to system uncertainty and process disturbances. The proposed architecture is also tested in case of process disturbance and systems faults. Simulation results showed that the couple tank system was successfully controlled with acceptable performance criterions in both cases.

DOĞRUSAL VE DOĞRUSAL OLMAYAN SİSTEMLERDE BULANIK SÜRÜ PARÇACIĞI OPTİMİZASYONU YAKLASIMI İLE KONTROL

ÖZET

Bu tezin amacı, yeni optimizasyon yöntemi olan parçacık sürü optimizasyon algoritmasını MATLAB'e uygulayarak bulanık PID kontrolörü katsayıları ve Takagi-Sugeno kural tabanındaki keskin değerleri cevrimdışı optimize ederek doğrusal ve doğrusal olmayan sistemlerin belirli çalışma koşulları altında kontrolünü sağlamaktır. Parçacık sürü optimizasyonunun diğer optimizasyon yöntemlerinden, örnek olarak verilmesi gerekirse genetik algoritmadan, en önemli avantajı optimizasyon sırasında az sayıda iterasyon içermesi, kolay anlaşılabilir olması ve bize kompleks olmayan az sayıda yazılmış bilgisayar kodları ile kolay ve ucuz bir şekilde uğraşmamızı sağlamasıdır. Genetik algoritma ile olan benzerlikleri ise her ikiside populasyon tabanlı olup, tek set değerden diğer set değerlere geçerken deterministik ve olası kuralları kullanmaları sayılabilir. Son yapılan çalışmalara istinaden parçacık sürü optimizasyon yöntemi en az genetik algoritma kadar büyük oranda doğrusal olmayan yapıların çözülmesinde, yakınsama oranı ve yakınsama hassasiyeti bazında aynı sonuçları vermektedir. Ayrıca basit kodlar içermesinden dolayı hem bilgisayar hafızasından hem de zamandan tasarruf ettirip sonuclara en hızlı ve verimli şekilde ulaşmamıza yardımcı olmaktadır. Parçacık sürü optimizasyon yöntemi doğrusal olmayan ve zamanla değişen karakteristiğe sahip olan ikili tank sisteminde belirli çalışma aralıkları içerisinde bulanık PID kontrolör tasarımında kolayca ve başarılı bir şekilde uygulanabilmiştir.

Yukarıda bahsedildiği gibi ikili tank sisteminin doğrusal olmayan ve zamanla değişen yapısından dolayı, kontrolör tasarımında tek set parametrelerin bulunması ve kontrol sırasında her bölge için aynı parametrelerin kullanılması neredeyse imkansızdır. Bu yüzden daha önceden belirlenmiş çalışma aralıkları içerisinde, Takagi-Sugeno kural tabanındaki parçacık sürü optimizasyon yöntemi ile optimize edilmiş katsayılar her bölge için sabit tutularak, değişik bölgeler için değişik optimal kontol parametreleri bulunup kontrol sırasında çevrimiçi olarak PID katsayılar hesaplanmıştır. Bulanık PID kontrolör parametreleri aynı zamanda ikili tank sisteminin ikinci tankındaki sıvı seviyesini giriş set değeri alarak farklı çalışma aralıklarında doğrusal regresyon yöntemi ile bulunan değişik kontrolör parametre fonksiyonları ile esnek bir yapıya dönüştürülüp farklı giriş değerleri, sistem gürültülerini hatta sistem hatalarını kompanze edecek duruma getirilimiştir. Böylelikle belirlenen çalışma bölgelerinde istenilen kontrol şartlarını sağlayan, değişik senaryolara sahip sistem hataları ve sistem gürültülerini bastıran adaptif yapıya sahip doğrusal olmayan bir sistemin geliştirilmiş parçacık sürü optimizasyonu yöntemi ve bulanık PID kontrolörü ile kontrolü sağlanmıştır.

Bu tez çalışması üç yapıya bölünmüştür. İlk yapıda, kontrolör katsayılarının ayarlanmasında literatüre geçmiş olan klasik yöntemler açıklanmış ek olarak bulanık mantık yapısı ve parçacık sürü optimizasyonu ile integrasyonunun nasıl sağlandığı açıklanmıştır. Klasik yöntemler olarak, Ziegler Nichols, Set Point Weighting, Cohen

Coon ve son olarak Internal Model Kontrol yöntemleri incelenmiştir ve uygulanış prensipleri anlatılmıştır. Bu tezin amacı, doğrusal ve doğrusal olmayan sistemlerin daha önceden belirlenmiş çalışma aralıkları içerisindeki bulanık kontrolör parametrelerinin ve Takagi-Sugeno kural tabanındaki keskin değerlerinin parçacık sürü optimizasyon yötemi ile optimize edilerek sistem kontrolünün sağlanmasıdır. Optimizasyon aracı olarak seçilen parçacık sürü optimizasyon algoritmasının geçmişte ortaya çıkmasındaki sebepleri, diğer optimizasyon araçları ile arasındaki farkları, gelişme süreçleri ve algoritmanın temel prensipleri anlatılıp MATLAB fonksiyon yapısını kullanarak parçacık sürü optimizasyon algoritması yazılmıştır. Yazılan bu algoritma SIMULINK'te kontrol edilecek sisteme erişim sağlaması bakımından esnek ve dışarıdan ulaşılabilir hale getirilmiştir. Standart sürü parçacığı optimizasyon yöntemine kıyasla daha verimli hale getirilen geliştirilmiş parçacık sürü optimizasyon yöntemi, değişik kıyaslama fonksiyonları (Sphere fonksiyonu, Rosenbrock fonksiyonu, Rastrigin fonksiyonu ve Greiwank fonksiyonu) üzerinde test edilmiştir. Geliştirilmiş parçacık sürü optimizayonu ile kıyaslama fonksiyonları üzerindeki test işleminin sonunda, yakınsama oranı ve yakınsama hassasiyeti diğer standart parçacık sürü optimizasyon yönteminden daha iyi sonuçlar vermesi üzerine, geliştirilmiş parçacık sürü optimizasyon yöntemi bundan sonraki çalışmalarda optimizasyon aracı olarak seçilmesine karar verilmiştir. İki farklı optimizasyon algoritmalarının test sonuçlarıda Matlab'de uygulanıp sonuçları tartışılmıştır.

Bulanık mantık yöntemi, değişken koşullara çabuk ve kolay uyum sağlayabilme özelliğinden ve daha önceden belirlenen belirsizlikler altında karmaşık işlerle başa çıkabilme özelliğinden dolayı bu çalışmada kontrol algoritmasında kullanılmıştır. Bulanık önermedeki sonuç ifadesinin yapısına göre bulanık kural tabanı Takagi-Sugeno tipi bulanık kurallardan oluşturulup tekli yapıya dönüştürülmüştür. Takagi-Sugeno bulanık modelinin sonuç kısmında, bir belirgin (kesin) fonksiyon mevcuttur. Dolayısıyla bu model hem matematiksel, hem de dilsel ifadelerle oluşturulan bir model olarak görülebilir. Bu çalışmada tasarlanan bulanık PID, ifade kolaylığı açısından çoklu giriş tekli çıkış biçimindedir. Bulandırıcı olarak tekli bulandırıcı seçilmesinin sebebi ise gerçek sistemler üzerinde yapılan uygulamalarda hesap kolaylığı sağlamasındandır.

Tekli bulandırıcı giriş değerleri olarak hata ve hatanın türevi işlem kolaylığı olmasından dolayı şeçilmiştir. Tasarım üçüncü bir değişken olan hatanın integralini de alacak şekilde esnek bir yapıya sahiptir. PID kontrol döngü yöntemi, basit yapıları ve tasarım kolaylıkları nedeniyle yaygın olarak endüstriyel kontrol sistemlerinde kullanılmasından dolayı bu tezde de doğrusal olmayan ve zamanla karakteristiği değişen ikili tank sisteminin kontrolünde uygulanmıştır. Proses kontrol uygulamalarının çoğu PI ve özellikle PID denetleyiciler ile yapılmaktadır. Zaman içinde çok sayıda denetleyici algoritmaları geliştirilse de, endüstride özellikle yüksek performans gerektirmeyen sistemler için yaygın kullanımı devam etmektedir. Gerçek sistemlerdeki doğrusal olmayan yapı ve oluşan parameter değişiklikleri nedeniyle, teoride uygulanan yöntemlerin uygulanmasında güçlükler yaşanmaktadır.

Tezin ikinci kısmı, üçüncü kısmının başlangıcı niteliğindedir. Bu kısımda değişik kıyaslama fonksiyonları (Sphere fonksiyonu, Rosenbrock fonksiyonu, Rastrigin fonksiyonu ve Greiwank fonksiyonu) üzerinde test edilip, performansının yeterliliğinden dolayı kullanılmasında karar kılınan geliştirilmiş sürü parçacık optimizasyon modeli, bulanık yapıya sahip PID kontrolöründeki optimal katsayı değerlerinin elde edilmesinde kullanılarak, birinci dereceden ölü zamalı sistem, ikinci dereceden ölü zamanlı sistem ve son olarak ikinci dereceden ölü zamanlı

integral etkili sistem üzerinde uygulanıp, benzetimleri MATLAB'de yapılarak sonuçlar irdelenmiştir. Kullanılan bu sistemlerde elde edilen bulanık PID katsayıları ve Takagi-Sugeno kural tabanı keskin değerleri çevrimdışı olarak bulunmuş ve optimizasyon kriteri olarak IAE (integral etkili mutlak hata) seçilmiştir. Ayrıca ikinci dereceden ölü zamanlı sistem değişik optimizasyon kriterlerine göre optimize edilerek benzetim sonuçlarının kıyaslanması sonucu optimizayon kriteri belirlenmiştir. Karar verilen optimizasyon kriteri çalışmanın geri kalanında tüm benzetimlerde kullanılmıştır. Bahsedilen sistemler üzerinde kolayca uygulanışı ve vermiş olduğu sonuçların kabul edilebilirliği, tasarlanmış olan yapıyı, doğrusal olmayan ve zamanla karakteristiği değişen ikili tank sistemi üzerinde uygulanmasına karar verilmiştir. Böylelikle tasarlamış olduğumuz yapının doğrusal olmayan bir sistem olan ikili tank sisteminin kontrolünde uygulanabilirliği test edilmiş olacaktır. Bu çalışma ise tezin üçüncü kısmını oluşturmaktadır.

Tezin üçüncü kısmında, optimizasyon aracı olarak seçilen geliştirilmiş parçacık sürü optimizasyonu, bulanık PID (oransal, integral etkili, türev etkili) kontrolörüne entegre edilerek, bu tür problemlerde iyi bilinen doğrusal olmayan ve zamanla karakteristiği değişen ikili tank sistemine uygulanmıştır. Sistem, aynı boyutlu iki silindirik sıvı tankının bir bağlantı borusuyla birleştirilmesiyle oluşturulur. Sistem, elektrikli pompa ile beslenirken, çıkış olarak ikinci tankın su seviyesi alınmıştır. Tekli tank sistemi ile doğrusal olmayan matematiksel denklemlerle ifade edilen yapı ikili tank sisteminin matematiksel denklemlerinin çıkarımında ve anlaşılmasıdan kolaylık olması bakımdan ele alınmıştır. Kütle-denge ve enerji denklemlerine göre matematiksel denklemleri çıkarılan yapı sistem kontrolünde kullanılmak üzere hazır hale getirilmiştir. Doğrusal olmayan ve zamanla karakteristiği değişen sistemlerin proses endustrisinde yaygın olarak karşılaşılmasından dolayı, tasarlanan yöntemin ikili tank sistemine uygulanmasına karar verilmiştir. Doğrusal olmayan sistemlerinin kontrolörünün zor olması klasik yöntemlerden farklı olarak diğer kontrol yöntemlerini, kullanmaya teşvik etmiştir.

Tasarlanan çalışmada ilk önce, üç değişik çalışma bölgesi içerisinde tanımlanan ikinci tanktaki su seviyesi sırasıyla giriş set değeri olarak seçilmiştir. Seçilmiş olan üç bölge için sırasıyla arasında kalan her nokta için, geliştirilmiş parçacık sürü optimizayon yöntemi kullanılarak, bulanık PID optimal katsayıları ve Takagi-Sugeno keskin değerleri bulunmuştur. Her bölge için değişik girişlere göre kapalı çevrim bulunan katsayılar içerisinde uyguluk fonksiyonlarının en az olan parametreler seçilip bölge başına tanımlanmış ve parametreler o bölgeler için dondurulmuştur. Bu parametreler sistemin yapısal parametreleri olup çevrimiçi uygulamalarda sabit tutulmuştur. Sonuç olarak her bölge için üç set parametre değerleri elde edilmiştir. Üç bölge içerisinde tanımlı olan kapalı çevrim ile bulunan, bulanık PID katsayıları daha sonra sistemin adaptif yapı kazanabilmesi için, sistemin giriş set değerlerine göre doğrusal regresyon yöntemi kullanılarak her bölge için ayrı dördüncü dereceden PID katsayı fonksiyonlarına dönüştürülmüştür. Böylelikle çevrimiçi otomatik olarak ayarlanan bulanık PID katsayıları proses performansı için yararlı hale getirilmiş aynı zamanda sistem bilinmezlikleri ve proses gürültülerini kompanze edecek adaptif özellik kazandırılmıştır.

Tasarlanan yapı, değişik genlikte ve yükseklikteki sistem girişlerine uygulanarak alınan sonuçların tatmin edici olduğu görülmüştür. Önerilen tasarım aynı zamanda değişik hata senaryolarına sahip ikili tank sistemine MATLAB/SIMULINK yardımı ile uygulanıp sonuçlar irdelenmiştir. Olası sistem arıza veya hatalardan bazıları, pompada farklı eyleyici hataları, Tank 1'in tabanında belli yarıçaplı daire biçiminde

bir delik, Tank 2'nin tabanında belli yarıçaplı daire biçiminde delik ya da her iki tankın tabanında belli yarıçaplı daire biçiminde delik olarak alınmıştır. Simülasyon sonuçları, değişik hata senaryolarına sahip ve sensor üzerinde gürültüsü olan ikili tank sisteminde kabul edilebilir sonuçlar verip sistemin tasarlanan yöntemle kontrol edilebilirliğini göstermiştir. Yötenmin başarımı, çift tanklı sıvı seviye kontrol sisteminin hatalı durumlarını içeren benzetim örnekleri ile gösterilmiştir.

1. INTRODUCTION

The PID (Proportional Integral Derivative) controller has been utilized as the workhorse of the process control industry. It is accepted universally amongst researchers and practitioners within the control community. The main advantage is its simplicity that no such simple structure that is more effective, robust and comparable in its dynamics. The parameters of the PID controller is another research area that attaracts researchers. Various methods have been proposed to search the parameters of PID controllers such as Ziegler Nicholas, Set Point Weightining, Cohen Coon and Internal Model Control methods (M.Zuang, 1993). The methods are often not applied in practice due to the necessity of control personnel to learn new techniques that are complicated and often time consuming. Besides that, the performances of the methods are not good enough due to the presence of multiple numbers of local optima in the system. Recently, as an alternative to the classical mathematical approaches, modern heuristic optimization techniques have been given much attention by many researchers because of their ability to find global optimal solutions and getting rid of the necessity of control operator within the process. Particle Swarm Optimization method whose mechanics is inspired by swarming and collaborative behavior of biological populations (Kenndy, J. and Eberhart, R., 1995) has been presented recently as a new evolutionary computational technique in various application fields. There has been much attention in terms of implementation of PSO in control theory. In addition, there are some comparison of effectiveness between PSO and other heurictic algorithm has been discussed thorugh some areas (Hassan, R., Cohanim, B. 2004). It is found out that PSO has the same effectiveness as other heuristic methods but significantly better computational efficiency on implementing some benchmark systems.

1.1 Purpose of Thesis

In this study, implementation of improved particle swarm optimization technique with fuzzy PID controller on different type of systems has been presented. The

parameters of PID and the crisp values of the rule base of fuzzy controller have been tuned offline to minimize some predetermined performance criterias. The performance results of the proposed approach have been depicted and it is seen that particle swarm optimization has been successfully implemented to the systems with better performance in terms of maximum overshoot, settling time and rise time. The proposed algorithm has been applied to nonlinear and time varying characteristics of the coupled tank system. The purpose is to control the height of tank 2 by fuzzy PID controller optimized by particle swarm optimization technique. Different liquid level ranges for tank two are chosen respectively as typical working points and input space is divided into three fuzzy subspaces based on working points. Even if the learning process is offline for the rule base, the fuzzy PID parameters are tuned online which make the parameters adjust in such a way that good performance will be ensured.

In this study, improved particle swarm optimization has been applied with linear changes on inertia weight for velocity and positioning update rather than exponentional changes (Wang, D. 2009). It is observed that improved particle optimization is able to converge to optimal solutions as well by using linear changes approach. There have been different studies on choosing the objective fuction in particle optimization tuning methods (Gao, F. and Tong, H. Q., 2006). It is seen that the performance of the system would be better as well when applying typical fitness function as IAE. There are different aspects for the adjustment of rule base in literature such as modification of shema in fuzzy controllers. Due to the observed information being related to a past instant and this delay information causes unsatisfactory results, rule base shifting method can be applied on different systems (Yesil, E., Guzelkaya, M. 2008). This study has flexibility to implement those approaches as well in terms of shifting the rule base when considering time delays on coupled tanks systems. Another approach has been explained on coupled tank system is a neuro-fuzzy-sliding mode controller using sliding surface. Developing a nonlinear sliding surface and fixed boundary layer in order to compensate chattering that means high frequency oscillations of the controller output. Inside the boundary layer, fuzzy logic can ben applied as well as on the outside of the layer the sliding mode control can be applied (Boubakir, A, 2009).

2. PID CONTROL

The PID controller has been widely used in process industries, energy production, and transportation as well as in manufacturing. It is the most fundamental control strategy in the control area. PID controller is generally preferred for control actions because of its simple algorithm, ability to adapt to wide range of applications where it can ensure excellent control performances. PID controllers have survived from many changes in technology from mechanics and pneumatics to microprocessors. Especially, improvement of microprocessors has given a highlighted importance for the evaluation of the PID controllers. These improvements on the microprocessors have provided additional features on PID controllers such as automatic tuning, gain scheduling and continuous adaptation [1].

PID controllers can also be used in control systems where the precise mathematical model of the systems is not available and hence analytical design methods or conventional design methods cannot be used. Recent research has indicated that even though PID controllers may not provide optimal control, it provides satisfactory control [1].

The design and analysis of PID controller requires three parameters. K_p, proportional gain, T_i, integral time constant, T_i, derivative time constant.

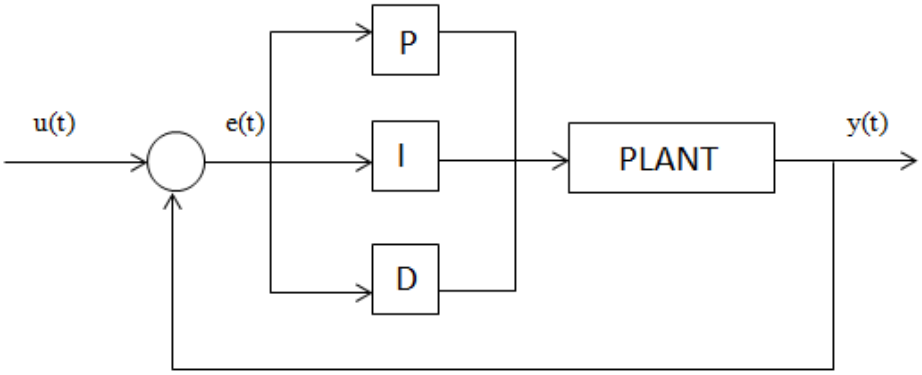

Figure 2.1 : Basic block diagram of PID controller.

PID control law is stated in equation **(2.1).**

$$u(t) = Kp\left[e(t) + Td\frac{de}{dt} + \frac{1}{T_i}\int e(\tau)d\tau\right] \quad \textbf{(2.1)}$$

$$e(t) = y_{sp}(t) - y(t)$$

Where *u(t)* is control input, *e(t)* is error which is difference between system output and set value. Equation **(2.1)** can be rewritten as a combination of the three terms :

$$u(t) = Kp\, e(t) + Kd\, \frac{de}{dt} + Ki\int e(\tau)d\tau \quad \textbf{(2.2)}$$

The P term is proportional to the error, the I term, which is proportional to the integral of the error, and D term refers to the derivative of the error. The controller parameters K_p, T_i and T_d are called proportional gain, integral time and derivative time respectively. These terms can be interpreted as past, present and future in control actions [2].

2.1 Tuning Rules for PID Controllers

As shown in Figure 2.2, for a PID controller, the tuning of the parameters indicated in the controller block can be very challenging. If the mathematical model of a plant can be derived, then

we can conclude that it is most likely to implement various design strategies called as a fixed parameter tuning methods, to find out the parameters of the controller that will meet steady state and transient specifications. Nevertheless, if the mathematical model is not known or hard to derive then fixed parameter tuning methods can be applied just for only starting point and necessity of trial and error approach will be required without ensuring good performance. Therefore, we should go through heuristics approaches for tuning the PID parameters, which this study has given a focus on. In this chapter, some of the fixed parameter tuning methods are briefly reviewed. Those are Ziegler – Nichols method, Set –point Weighting method, Cohen – Coon method and finally Internal Model Control method [2].

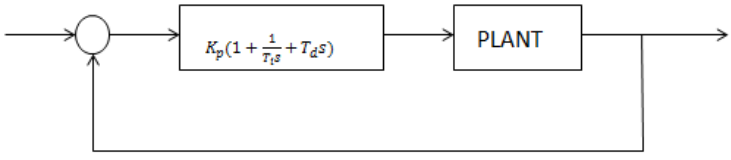

Figure 2.2 : PID control of a plant.

2.1.1 Ziegler – Nichols method

Controller tuning means selecting the controller parameters that will meet given performance spesifications. Ziegler – Nichols (ZN) is a tuning rule that proposes tuning strategy in terms of finding a set value for K_p, T_i and T_d based on experimental responses or based on the value of K_p. This method is implemented when mathematical model of a system cannot be derived. It needs to be noticed that Ziegler Nichols (ZN) method cannot guarantee minimum overshoot in the step response. This method provides only the starting point for obtaining the optimal PID parameters. We need a sequence of fine tunings until an acceptable result is obtained.

Ziegler – Nichols method offers two ways of implementing the tuning rules. In the first method, the step response of the plant is obtained experimentally. If the plant does not involve either integrators or dominant complex conjugates poles, the response will be seen as S shaped curve given in the Figure 2.3. The S shape curve is defined by two constants; delay time L and time constant T, which are derived by drawing a tangent line at inflection point of the curve [1].

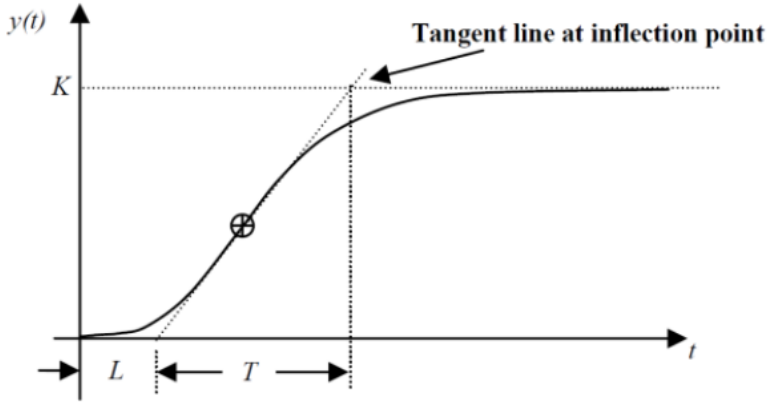

Figure 2.3 : S-shaped response curve.

The intersection of tangent line and coordinate axes give the parameters α, L. ZN method gives PID parameters directly as functions of α and L stated in Table 2.1.

Table 2.1 : PID controller parameter obtained from ZN first method.

Controller	K	Ti	Td
P	T/L	∞	0
PI	$0.9\,T/L$	$L/0.3$	0
PID	$1.2\,T/L$	2L	0.5L

In the second version of the Ziegler Nichols method, the plant controller parameters T_i and T_d are set to ∞ and zero respectively, which make the controller as proportional controller. When settings are done, K_p needs to be increased from zero to K_{cr} (ultimate gain) at which the output first start to oscillate. It needs to be noticed that if the output is not oscillatory for any gain value then this method cannot be implemented to the system. After finding the ultimate gain and ultimate period, the controller parameters can be calculated from Table 2.2 below.

Table 2.2 : PID controller parameter obtained from ZN second method.

Controller	K	Ti	Td
P	0.5Kcr	∞	0
PI	0.45Kcr	$^1/_{1.2}\,Pcr$	0
PID	0.6Kcr	0.5Pcr	0.125Pcr

2.1.2 Set-point Weighting method

Although Ziegler-Nichols method has the ability to reject disturbances, the compensated system response to a step input may result in high overshoot or the computed control signal can be high which may lead to saturation of actuators. In order to compensate for these situations, set point for the proportional action can be weighted as below.

$$e(t) = b.y_{sp}(t) - y(t) \tag{2.3}$$

The advantage of the set point weighting is to reduce overshoots in the closed loop set-point step response. With the above equation, the general controller equation becomes:

$$u(t) = Kp\,(b.y_{sp}(t) - y(t)) + Kd\,\frac{de}{dt} + Ki\int e(\tau)d\tau \tag{2.4}$$

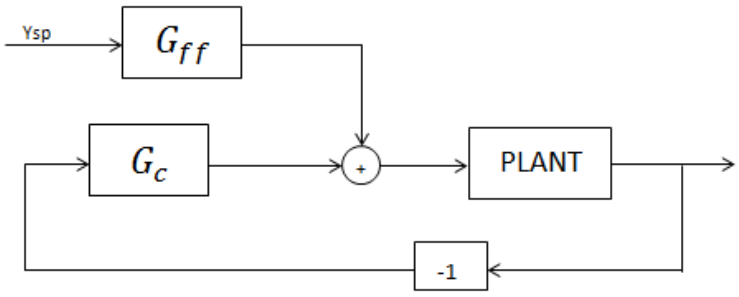

Figure 2.4 : Two degrees of freedom scheme of PID controller.

$$G_{ff} = K_p(b + \frac{1}{sT_i} + sT_d) \quad \textbf{(2.5)}$$

$$G_c = K_p(1 + \frac{1}{sT_i} + sT_d) \quad \textbf{(2.6)}$$

The value b is very important because of the fact that closed loop response is sometimes very sensitive to the weights. A small change in the value of b can result in completely different response of the system. In order to be in accordance with the set point changes, it is necessary to follow a procedure to determine b. Astrom and Hagglund mentioned the dominant pole design method in [3]. In this method, the closed loop system will take two complex conjugate poles and one pole on the real axis as -p_0 with the set point weighting the closed loop system has zero at

$$s = -z_0 = \frac{1}{bT_i} \quad \textbf{(2.7)}$$

By choosing b so that $z_0 = p_0$, we make sure that the set point does not excite the mode corresponding to the pole in$-p_0$. This will work and will give good transient responses for the systems where the dominant poles are well damped ($c = 0.7$). For the systems where the poles are not well damped, the choice $z_0 = 2p_0$ yields a system with less overshoot [3].

The suitable parameter b can be calculated as:

$$b = \begin{cases} \dfrac{0.5}{p_0 T_i}, if\ c < 0.5 \\ \dfrac{0.5 + 2.5(c - 0.5)}{p_0 T_i}, if\ 0.5 \le c \le 0.7 \\ \dfrac{0.5}{p_0 T_i}, if\ c > 0.7 \end{cases} \quad \textbf{(2.8)}$$

2.1.3 Cohen-Coon method

The Cohen-Coon tuning method is based on the first order plus dead time delay process model with main design specification as quarter amplitude decay ratio in response to load disturbance.

$$G_p = \frac{K_p}{1 + sT} e^{-sL} \quad \textbf{(2.9)}$$

The main design objectives are to maximize the gain and minimize the steady-state error for P and PD controller. For PI and PID control, the integral gain is maximized. This corresponds to minimization of integrated error, the integral error due to a unit step load disturbance. For PID controllers three closed loop poles are assigned; two poles are complex and the third pole is located at the same distance from the origin as the other poles.

The parameters $\alpha = {}^{K_p L}/_{T}$ and $\tau = {}^{L}/_{L+T}$ are used in Table 2.3. If the system can be defined by K_p, L and T, then it is possible to give tuning formulas with the help of Table 2.3.

Table 2.3 : Controller parameters for Cohen-Coon method.

$\frac{ke^{-\theta s}}{\tau s+1}$	K	Ti	Td
P	$\frac{1}{k}(\frac{\tau}{\theta}+0.35)$		
PI	$\frac{0.9}{k}(\frac{\tau}{\theta}+0.92)$	$\frac{3.3\tau+0.3\theta}{\tau+2.2\theta}\theta$	
PD	$\frac{124}{k}(\frac{\tau}{\theta}+0.13)$		$\frac{0.27\tau-0.09\theta}{\tau+0.13\theta}\theta$
PID	$\frac{1.35}{k}(\frac{\tau}{\theta}+0.18)$	$\frac{2.5\tau+0.5\theta}{\tau+0.61\theta}\theta$	$\frac{0.37\tau}{\tau+0.19\theta}\theta$

It may be difficult to choose desired closed-loop poles for higher order systems. If τ is small, controller parameters are close to others that are obtained by Ziegler Nichols tuning rules [3].

2.1.4 Internal Model Control method

Internal Model Control (IMC) described by Morari and Zafiriou (1989) is a general design procedure for obtaining controllers that meet requirements for stability, performance and robustness of the control systems. A block diagram of such a system is shown in Figure 2.5.

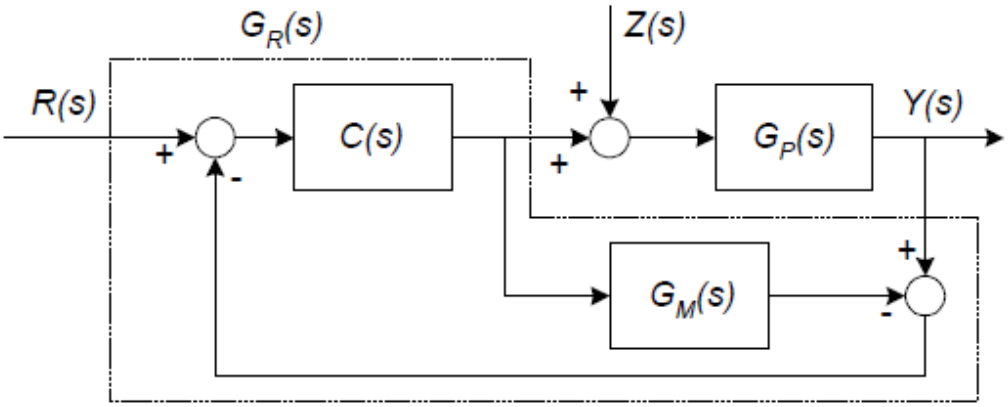

Figure 2.5 : Closed loop system with controller based on the IM principle.

If the model of the system $G_M(s)$ matches $G_p(s)$ and load disturbance is not present, the output of the model cancels the output of the process. In that case, the process turns to control in open loop. If there is a model mismatch and load disturbance is Z(s), then there will be feedback signal and feedback control will be applied. The first step in internal model control is to factor the transfer function modeling the process,

$$G_M(s) = G_M^+(s)G_M^-(s) \tag{2.10}$$

where $G_M^-(s)$ is an inverse of $G_M^+(s)$ which contains only the left half plane poles and zeros and $G_M^+(s)$ contains all the time delays and rigt half plane zeros. The controller *C(s)* is defined as below,

$$C(s) = \left(G_M^-(s)\right)^{-1} G_F(s) \tag{2.11}$$

where $G_F(s)$ is a low pass filter which guarantees that the controller *C(s)* is realizable. The usual form of the filter is below.

$$G_F(s) = \frac{1}{(1+T_F s)^n} \tag{2.12}$$

As indicated in the figure, the relation between a conventional feedback controller $G_R(s)$ and internal model controller C(s) expressed with

$$G_R(s) = \frac{C(s)}{1 - C(s)G_M(s)} \tag{2.13}$$

or inversely:

$$C(s) = \frac{G_R(s)}{1 + G_R(s)G_M(s)} \tag{2.14}$$

The FOPDT model can be used in the internal model control, but the part of the transfer function modeling dead time has to be evaluated with Pade approximations.

First order Pade approximation of the dead time is:

$$e^{-sTt} \approx 1 \tag{2.15}$$

and it leads an IMC PI controller with the following parameters:

$$K_p = \frac{T_1}{K_1 T_F}, T_I = T_1 \tag{2.16}$$

The recommended value for the filter time constant should satisfy $T_F > 1.7T_{t1}$.

The first order Pade approximation:

$$e^{-sTt} \approx \frac{1 - sT_{t1}/2}{1 + sT_{t1}/2} \tag{2.17}$$

The FOPDT model and IMC design lead to a PID controller with parameters:

$$K_p = \frac{2T_1 + T_{t1}}{K_1(2T_F + T_{t1})} \tag{2.18}$$

$$T_I = T_1 + \frac{T_{t1}}{2} \tag{2.19}$$

$$T_D = T_1 + \frac{T_1 T_{t1}}{2T_1 + T_{t1}} \tag{2.20}$$

and the recommended value for the filter time constant is $T_F > 0.8T_{t1}$.

IMC tuning rules are expressed in terms of process model parameters and can ben implemented after the identification of the process model [4].

3. FUZZY CONTROL

Contrary to conventional control approaches where control techniques requires mathematical models of the system and by using the mathematical models of the system to design a controller depicted into differential equations, fuzzy control is based on fuzzy logic mathematical system that processes crisp values in terms of logical variables that take on continuous values between 0 and 1. In a way, differential equations are the languages of conventional control while heuristics and rules about how to control the system are the languages of fuzzy control. Fuzzy control methodology can provide the representation or reflection for manipulating and implementing an operator's heuristic knowledge about how to control a system. Fuzzy control provides an efficient structure to convert linguistic information from human experts into numerical information.

Lotfi Zadeh from University of California, Berkley, introduced the concept of fuzzy logic as a way of processing data by allowing partial set membership rather than crisp set membership. This approach was not implemented in control theories until mid 70's because of lack of sufficient capability of the computers.

In conventional control, even if the design of the system can be possible or the mathematical model of the complicated systems can be achievable, the model may be too complex to use in controller design. Especially, some conventional techniques for construction of controllers require some assumptions while linearizing a nonlinear system. Hence, fuzzy control has been developed to find some alternative control techniques for control theory rather than struggling with failure modes on conventional control techniques [5].

In terms of performance objectives and design constraints, there is no difference between conventional control and fuzzy control since purpose is still to meet same type of closed loop specifications like minimum overshoot, minimum settling time, low steady state error etc… Fuzzy systems have been used in a wide range of applications in science, medicine, engineering, business etc… Fuzzy control has been

successfully used in aircrafts, automobiles, manufacturing systems, process control and robotics. Advantages of the fuzzy control can be summarized below

- An explicit model of the plant or process is not required
- Human experience, expertise and qualitative knowledge can be incorporated
- Incomplete, imprecise, general and approximate knowledge may be incorporated.
- Explicit optimization is not needed
- Suitable for large-scale and complex systems where analytical modeling is difficult.

3.1 Internal Structure of Fuzzy Controllers

The fuzzy controller is mainly composed of four main components. First part is the "fuzzification" which converts controller inputs into information that inference mechanism can easily perceive to activate and implement rules. Second part is the "rule base" where the knowledge is kept in the form of fuzzy logic sets of rules. There are different kinds of rule bases, such as Mamdani type of rule base, Singleton type of rule base, Takagi-Sugeno type of rule base and Tsukamoto type of rule base. In this study, Takagi-Sugeno type of rule base is used with particle swarm optimization technique. Third part is the "inference mechanism" which evaluates the expert's decision making in interpreting and deciding what the control input to the plant should be given. Last part of the fuzzy controller is the "defuzzification interference" which converts fuzzy outputs decided by the inference mechanism into the crisp input to the plant [6].

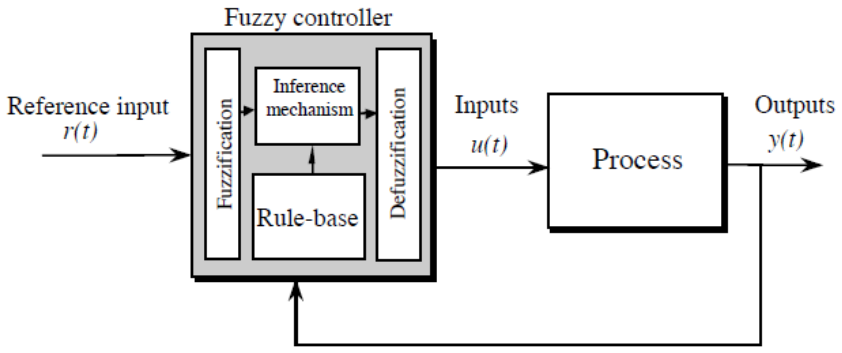

Figure 3.1 : Internal structure of fuzzy controller in closed loop control system.

3.1.1 Fuzzification

Fuzzification is the first step of fuzzy inference process that decomposes crisp inputs measured by sensors to fuzzy sets. The crisp inputs such as height, temperature, pressure or velocities are evaluated by inference engine as fuzzy inputs. Each crisp input has their own group of membership functions or sets that are to be processed by the fuzzy inference unit. Different fuzzy sets can be defined linguistically for different systems. It will be discussed in further chapters that, in this study, in order to decompose crips inputs of e and de/dt for fuzzy inference engine, three triangular membership functions have been defined between [-1, 1] as Negative, Zero and Pozitif as illustrated below.

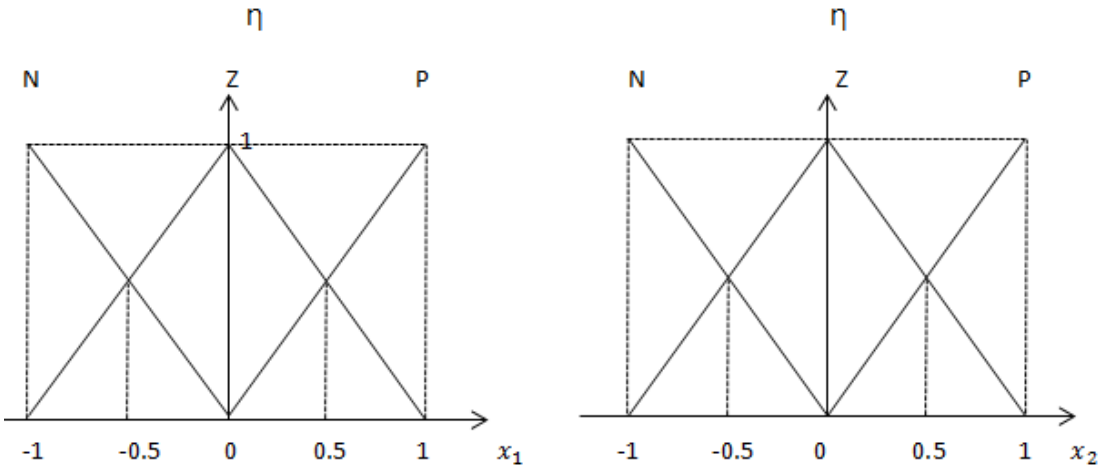

Figure 3.2 : Membership functions for e, x_1 and de, x_2.

These sets cover the other sets partially, hence some crisp inputs are members of different fuzzy sets. However, each input has different degrees of membership in different fuzzy sets. These membership degrees are utilized in controller processes.

3.1.2 Rule Base

Fuzzy rules are linguistic IF-THEN constructions which have the general form of "if A then B" where A and B are condition and conclusion respectively. The controller can be applied to either multi-input-multi-output (MIMO) problems or single-input-single-output problems. The controller needs normally three different crisp inputs that are the error, the change of error and the integrated error. In principle, the third variable, the integral of error is hard to define by the operators and engineers. Therefore, it is generally preferred to use two inputs, the error and the change of the

error. To simplify, the control objective is to regulate some process output around a prescribed set point or reference.

A linguistic controller contains rules in the IF-THEN format such as,

1. If error is negative and change in the error is negative then output is negative big.
2. If error is negative and change in the error is zero then output is negative medium.
3. If error is negative and change in the error is pozitif then output is zero.
4. If error is zero and change in the error is negative then output is negative medium etc…

3.1.3 Inference Mechanism

The inference mechanism has two main tasks; the first task is to determine the firing strength of each rule. Crisp inputs that passed through the fuzzification and became fuzzy inputs are evaluated for each rule in the rule base. Depending on the defined membership functions of the inputs, some of the rules will be fired.

The other task is to combine the outputs of fired rules to obtain a fuzzy set as the overall output of the inference mechanism. This output will be the input of the defuzification stage where it is converted to a crisp value.

3.1.4 Defuzzification

The output of the inference engine is the input of the defuzzification stage. The fuzzy set, which is the output of the inference mechanism, is converted to a crisp value by using defuzzification methods in order to get a scalar value as the control input to the system. There are various methods for defuzzifications. The centroid method is the most popular one in which the centre of the mass of the result gives the crisp value. Another approach is the height approach, which takes the value of the biggest contributor.

3.2 PID Tuning Method Using Fuzzy Logic

Different structures of fuzzy controllers have been studied and developed recently. A simple way of constructing a fuzzy PID controller is to combine a fuzzy PD controller with an integrator and to add a summation unit at the output as depicted below;

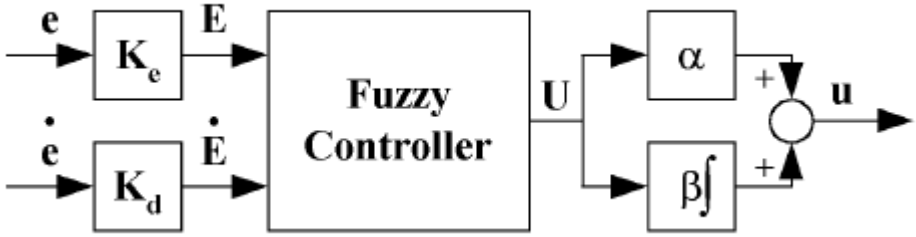

Figure 3.3 : Fuzzy PID controller.

The design of the fuzzy PID controller above has less number of rules and scaling factors compared to other fuzzy PID structures [6].

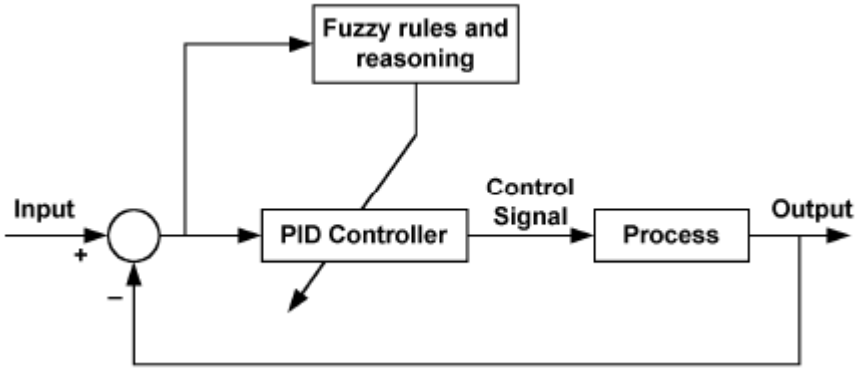

Figure 3.4 : Block diagram of fuzzy tuning PID controlled system.

In this study, fuzzy logic controller parameters are tuned in terms of an optimization of the parameters with particle swarm optimization. Error and change in error are the inputs of defuzzification, as analog inputs, which will then be processed in terms of linguistic variables in order to make inference engine analyze the information. While processing of inference engine, from the standpoint of optimization approach, some predetermined fitness functions can be used to minimize the error by adjusting values of inference engine or parameters of fuzzy controller.

4. PARTICAL SWARM OPTIMIZATION

Particle swarm optimization is a recently proposed heuristic search method for optimization of continuous nonlinear functions, inspired by the swarm methodology. The method was derived through simulation of simplified social models such as bird flocking, fish schooling and swarming theory in particular. Kennedy and Eberhart invented partical swarm optimization in the mid 1990's while trying to simulate the choreographed, graceful motions of swarms of birds. Particle swarm optimization has two roots. One of them is to tie to artificial life. It is also related to evolutionary computation such as genetic algorithms and evolutionary programming. The ability of flocks of birds, schools of fish and herds of animals to adapt to their environment, to avoid predators and to find rich sources of foods by implementing an "information sharing" approach intrigued the inventors of the methodology. Among other heuristic search methods, it can be easily implemented in a few lines of computer code in the cheapest manner. It requires only primitive mathematical operators, which makes it advantageous in terms of both availability of larger memory and higher speed. Particle swarm optimization has successfully been applied to a wide variety of problems such as neural networks, structural optimization, share topology optimization and fuzzy systems.

4.1 The Evolution of Paradigms of Particle Swarm Optimization

Reynold, Heppner and Grenander firstly presented bird flocking with simulations (Reynolds, C., 1987). Reynold was interested in the choreography of bird flocking, nevertheless Heppner and Grenander were interested in the main logic underlying how birds flock synchronously, changing their direction suddenly. Both of these scientists had the idea that local pressures make the graceful motions of swarms of birds. The intent which underlyies bird flocking is the manipulation of inter-individual distances which means being a function of birds' effort to maintain an optimum distance between themselves and their neighbors [7].

4.2 The Etiology of Particle Swarm Optimization

In order to easily understand the concept of particle swarm optimization, it would be better to explain its conceptual development. The algorithm began as a simulation of simplified social milieu. Particles were assumed collision-proof birds and the original intent was to simulate the unpredictable group dynamics of bird flocking behavior. As sociobiologist, E.O Wilson has written in reference to fish schooling, "In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantages of competition for food items, whenever the resource is unpredictably distributed in patches" (Wilson, 1975).

4.2.1 Simulating a social behaviour

A number of scientists have created computer simulations of various interpretations of the movement of organisms in a bird flock or fish schooling. Both model relied heavily on manipulation of inter-individual distances; that is, the synchrony of flocking was thought to be a function of birds' efforts to maintain an optimum distance between themselves and their neigbors. Reynolds proposed a behavior model in which each agent follows three rules [7].

- Seperation: Each agent tries to move away from its neighbors if they are too close.
- Alignment: Each agent steers towards the average heading of its neighbors.
- Collision: Each agent tries to go towards the average position of its neighbors.

The following models are given in the following figure respectively for illustration of the simple concept.

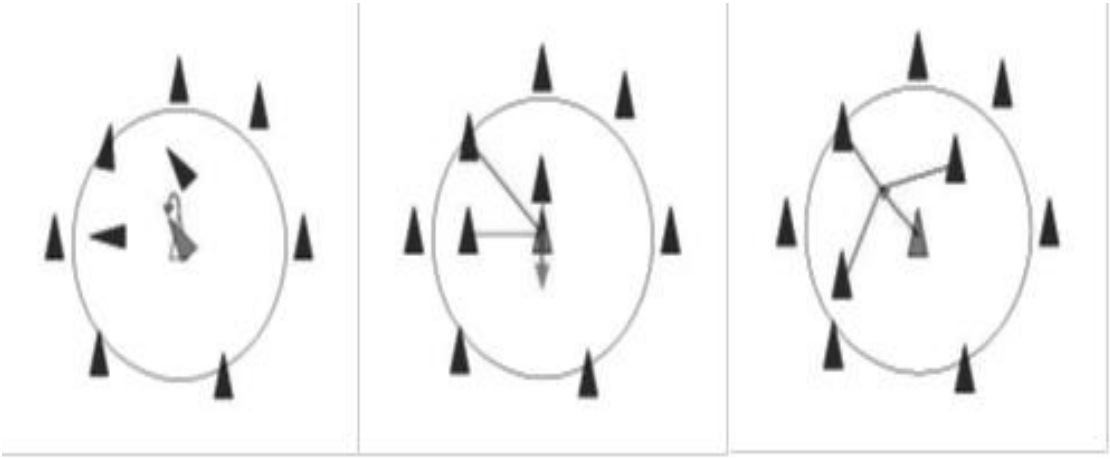

Figure 4.1 : Seperation, alignment and collosion (Reynold, 1987).

4.2.2 Nearest neighbor velocity matching and craziness

The first attempt for simulation was to write a computer code based on nearest neighbor velocity matching and craziness. A population of particles were randomly located on a torus pixel grid and with velocities in x and y directions.

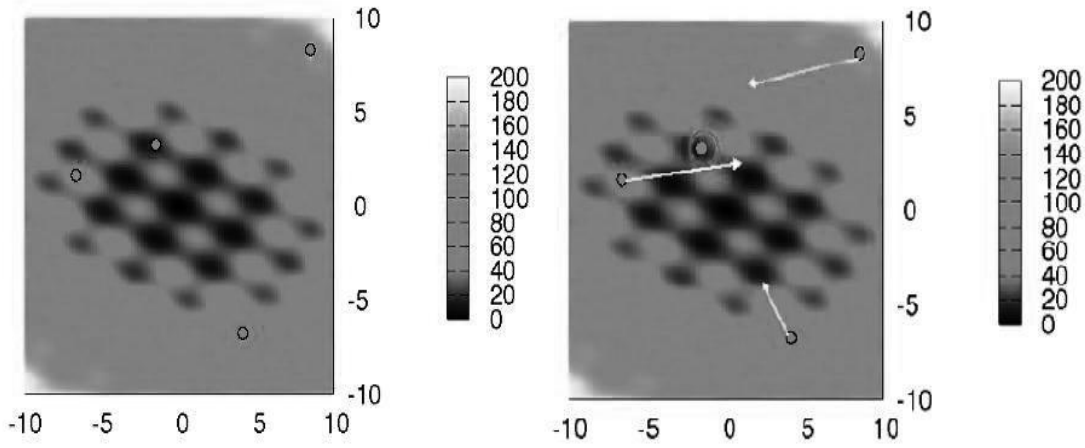

Figure 4.2 : Particles on a torus pixel with velocities (Reynold, 1987).

At each iteration for each particle, a loop in the program decides which other agent is its nearest neighbor and assign that particles' X and Y velocities to the agent in focus. This adjustment of each individual's velocities and positions according to agent in focus makes a synchrony of movement [7].

To give the simulation lifelike appearance, a stochastic variable called "craziness" was added to randomly chosen X and Y velocities. In birds' flocking or fish schooling, a bird or a fish often changes direction suddenly. This is described by using a "craziness" factor.

4.2.3 Roost and the cornfield vector

A new feature called “roost” which is introduced as a dynamic force factor was presented in Heppner’s simulations. The roost attracted them until they finally landed there. This eliminated the need for a variable like craziness. In this simulation, the particles knew the position of the roost but in real life, great number of birds will find a roost even though they had no previous knowledge of its location. So each agent shared information with its neighbors, originally all other agents, about its closest location to the roost [7].

Kennedy and Eberhart, inventors of particle swarm optimization, included a roost in Heppner-like simulation given in the following figure, so that:

- Each particle was attracted towards the location of the roost.
- Each particle remembered where it was closer to the roost.
- Each particle shared information with its closest location to the roost.

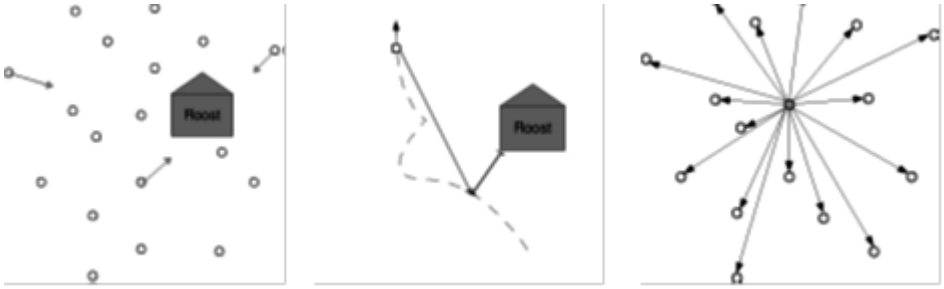

Figure 4.3 : Roost used in Heppner-like simulations to attaract the particles.

Instead of a known position, the authors defined a cornfield vector and each particle was programmed to evaluate its present position. If the point (100,100) represents the cornfield, the function value is zero at that point and the proposed function is expressed as **(4.1).**

$$f(x,y) = \sqrt{(x-100)^2} + \sqrt{(y-100)^2} \qquad \textbf{(4.1)}$$

The proposed velocity and the position update of the particles are given such that each particle remembers the best value p_{best} and the best position p_{bestx} and p_{besty}. Its X and Y velocities are updated in a simple manner as shown below:

$$\text{present}_x > \text{pbest}_x \text{ then } v_{k+1} = v_k - \text{rand} * p_{\text{increment}} \quad \textbf{(4.2)}$$

$$\text{present}_x < \text{pbest}_x \text{ then } v_{k+1} = v_k + \text{rand} * p_{\text{increment}} \quad \textbf{(4.3)}$$

Each particle knows the globally best position of one member of flock has found g_{bestx}, g_{besty} so far and its value is $g_{best} = [g_{bestx}, g_{besty}]$. Again X and Y velocities are updated as expressed below:

$$\text{present}_x > \text{gbest}_x \text{ then } v_{k+1} = v_k - \text{rand} * g_{\text{increment}} \quad \textbf{(4.4)}$$

$$\text{present}_x < \text{gbest}_x \text{ then } v_{k+1} = v_k + \text{rand} * g_{\text{increment}} \quad \textbf{(4.5)}$$

The new position is calculated as below:

$$x_{k+1} = x_k + v_{k+1} \quad \textbf{(4.6)}$$

Deduction of the nearest velocity matching makes the optimization slightly faster and changes the visual effect more likely to swarm from flock. If $g_{increment}$ and $p_{increment}$ are set relatively high, the flock seems to rapidly converge into the cornfield. If $g_{increment}$ and $p_{increment}$ set low, the flock swirls around the goal and approaches it realistically and finally lands onto the target. If $p_{increment}$ is set relatively higher than $g_{increment}$, it results in the excessive wandering of isolated individuals through the problem space while the reverse results in the flock rushing prematurely towards local minima. Approximately equal values of two increments seem to result in the most effective search of problem domain.

4.2.4 Modifications of the proposed method

Some experimentation revealed that instead of adjusting the velocities on a crude inequality test like “if presentx > bestx, make it smaller”, “if present < bestx, make it bigger”, it would be better to revise the algorithm to make it easier to understand and to improve its performance. Therefore, the velocity was adjusted according to difference of current velocity and best velocity of an individual achieved so far. The necessity of removing the increments was soon realized because there is no way to guess which one should be larger in order to yield good performance. Therefore,

these two terms were also removed out of the algorithm. The current simplified particle swarm optimization now adjusts the velocities more swarm like than any other paradigms.

It became more obvious that the behavior of the population of agents is now more like a swarm rather than a flock. Swarm Intelligence systems are typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local interactions between such agents lead to the emergence of complex global behavior. Natural examples of swarm intelligence include ant colonies, bird flocking, animal herding, bacterial growth and fish schooling. Millonas proposed five basic principles of swarm intelligence [7].

- Proximity principle: The population should be able to carry out simple space and time computations.
- Quality principle: The population should be able to respond to quality factors in the environment.
- Principle of diverse response: The population should not commit its activities along accessively narrow channels.
- Principle of stability: The population should not change its mode of behavior every time the enviroment changes.
- Principle of adaptability: The population must be able to change behavior mode when it is worth to computational price.

4.3 General Particle Swarm Optimization Algorithm

As mentioned in previous section, particle swarm optimization belongs to the category of swarm intelligence methods related to evolutionary computation techniques motivated by biological genetics and natural selection. Particle swarm optimization shares many similarities with genetic algorithm as an evolutionary computation technique. One important similarity is that both are initialized with a population of random solutions and searches for optima by updating generations. The dynamics of population in particle swarm optimization is similar to the collective

behavior and self-organization of socially intelligent organisms. All single individuals in the population capitalize information between the other's and benefit from their discoveries while exploring the local minima.

The basic particle swarm optimization algorithm consists of three steps; generating particles' positions and velocities, velocity update and position update. In this algorithm, a particle, which can be represented as a point in design space, tends to change its position from one move to another move based on velocity updates.

At first, the positions x_k^i and velocities v_k^i of the initial swarm of particles are randomly generated using upper and lower bounds on the design variables values x_{min} and x_{max}, as shown in equations **(4.7)** and **(4.8)**.

$$x_0^i = x_{min} + rand(x_{max} - x_{min} \tag{4.7}$$

$$v_0^i = \frac{x_{min} + rand(x_{max} - x_{min})}{\Delta t} = \frac{position}{time} \tag{4.8}$$

The second step is to update the velocities of all particles at time *k+1* by using the particles fitness values which are functions of the particles current positions in the design space at time *k*. The fitness function value is used to decide which particle has the best global value in the current swarm, p_k^g and determines the best position of each particle over time, p^i. The velocity update formulation uses this information for each particle in the swarm. The velocity update formulation also includes some random parameters, which are represented by uniformly distributed variables, to ensure good coverage of the design space and avoid getting captured in local optima. The update formula for velocities is given in equation **(4.9)**.

$$v_{k+1}^i = wv_k^i + c_1 rand \frac{(p^i - x_k^i)}{\Delta t} + c_2 rand \frac{(p_k^g - x_k^i)}{\Delta t} \tag{4.9}$$

where v_{k+1}^{i} is the velocity of particle *i* at time *k+1*, *w* is the inertia weight ranging between 0.4 to 1.4, c_1 is self confidence factor ranging between 1.5 to 2, c_2 is swarm confidence factor chosen between 2 to 2.5. It can be easily said that velocity update formulation is made up of a combination of current motion, particle memory influence and swarm influence. v_k^i is limited by a max velocity V_{max} as below [8].

$$V_{ij} = \begin{cases} V_{ij} \ , \ if \ |V_{ij}| \le V_{max} \\ -V_{max} \ , \ if \ V_{ij} < -V_{max} \\ V_{max} \ , \ if \ V_{ij} > V_{max} \end{cases} \quad \textbf{(4.10)}$$

The last step is to update the positions of all particles at time *k+1*. The position of each particle is updated using its velocity vector as shown in equation **(4.11)** and depicted in Figure 4.4.

$$x_{k+1}^{i} = x_k^i + v_{k+1}^i \Delta t \quad \textbf{(4.11)}$$

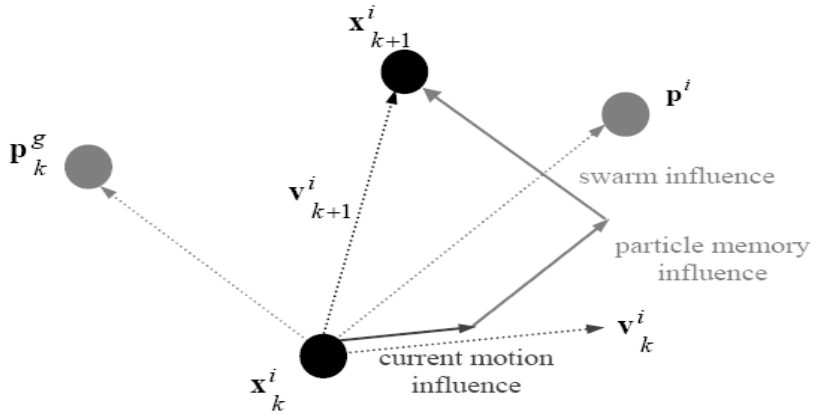

Figure 4.4 : Depiction of the velocity and position updated in PSO.

Sometimes x_{k+1}^{i} can be modified as [9]:

$$x_{k+1}^{i} = x_k^i + \gamma v_{k+1}^i \Delta t \quad \textbf{(4.12)}$$

where γ is constriction factor, normally $\gamma = 1$.

$$\gamma = {2k}\Big/{\left|2 - \emptyset - \sqrt{\emptyset^2 - 4\emptyset}\right|} \quad \textbf{(4.13)}$$

with $\emptyset = c_1 + c_2$ and $k = 1$. A complete theoretical analysis of the derivation of **(4.13)** can be found in [22, 23]. The complete flow diagram of particle swarm optimization algorithm depicted in Figure 4.5.

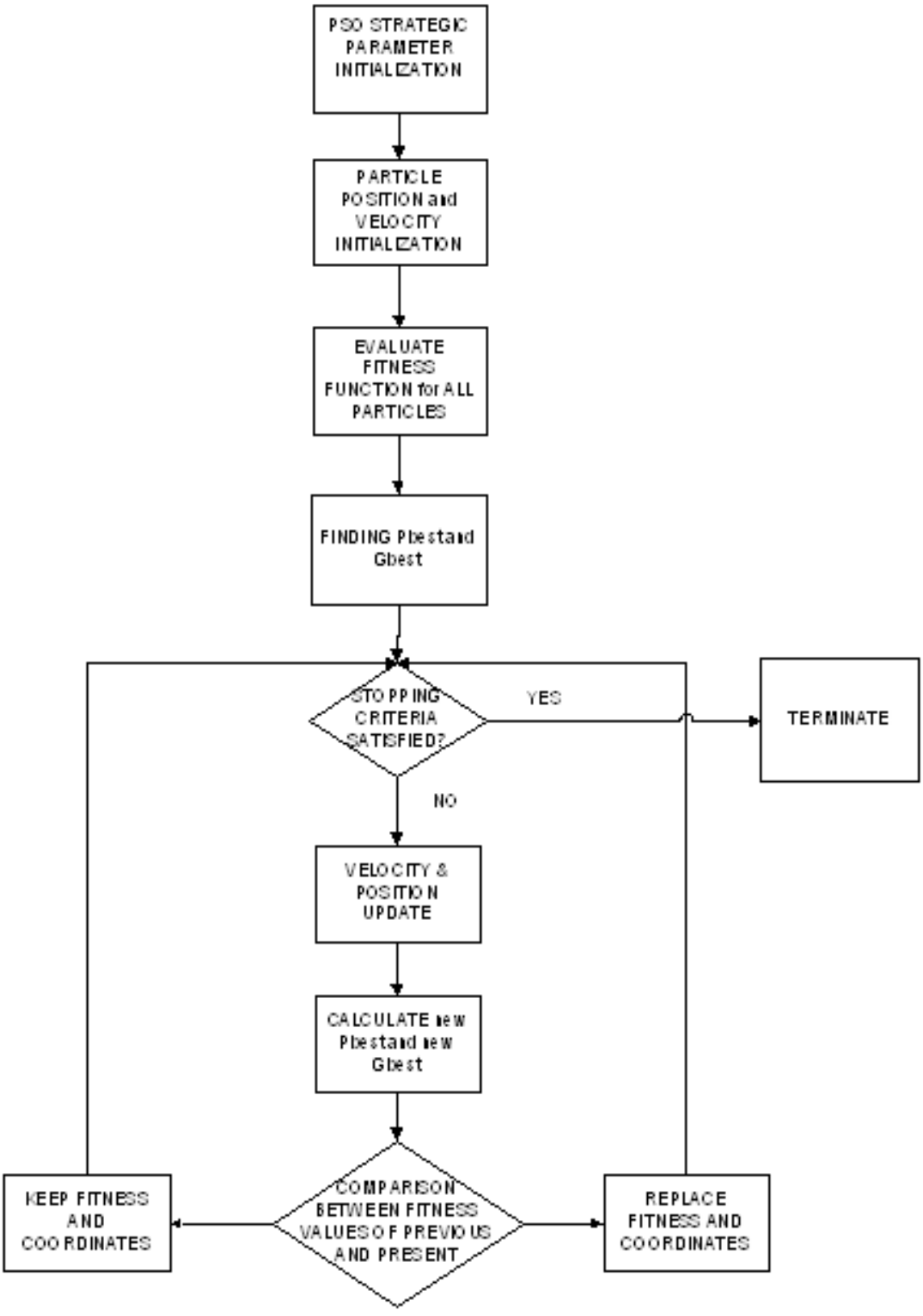

Figure 4.5 : Flow diagram of a particle swarm optimization.

4.4 An Improved Particle Swarm Optimization Algorithm

The particle swarm optimization is a stochastic optimization technique for finding optimal regions of nonlinear continuous functions through the interaction of individuals in the swarm. An improved particle swarm optimization algorithm is used in this study in all applications since it ensures better performance and faster convergence compared to classical particle swarm optimization. The new algorithm is applied to some benchmark problems; the numerical experiments show that improved particle swarm optimization algorithm has better performance than the standart particle swarm optimization and partical swarm optimization with inertia weight.

While PSO runs, the inertia weight *w* decreases either linearly or exponentially depending on the value of constant *k.*

$$w = w - k(w_s - w_e)/T \tag{4.14}$$

In most application of the algorithm w_s takes the value of 0.95 and w_e takes 0.4. As mentioned above particle swarm optimization is able to search more globally at the begining of the algorithm because of the large inertia weight. But, when the number of iterations increase, inertia weight decreases and algorithm looses the ability of searching globally to searching more locally. But it has now high efficiency in convergence due to the small inertia weight. The new proposed velocity update based on improved algorithm can be modified as [9]

$$v_{k+1}^i = wv_k^i + c_1 rand\frac{(p^i - x_k^i)}{\Delta t} + c_2(1 - w/2)rand\frac{(p_k^g - x_k^i)}{\Delta t} \tag{4.15}$$

New social weight is brought to the existing velocity update formula. The intention is to make social weight increased while inertia weight decreasies. In this case small social weight makes the global best position p_k^g have minor impact on the velocity updating. At the end of the run, the large social weight ensures the best particles information make a great influence on the swarm search behavior. According to the above proposed method, the structure of the algorithm isn't changed except adding the small social weight.

4.4.1 Experimental results and discussion

In order to show the performances of the improved algorithm, the numerical experiments are conducted on various benchmark functions such as Sphere function, Rosenbrock function, Rastrigin function and Greiwank function.

- Sphere function

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \tag{4.16}$$

- Rosenbrock function

$$f_2(x) = \sum_{i=1}^{100} (100(x_{i+1} - x_i)^2 + (x_i - 1)^2) \tag{4.17}$$

- Rastrigrin function

$$f_3(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{4.18}$$

- Greiwank function

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{4.19}$$

For the above four functions, the dimension which treated as population size, search range and the optimal value are listed in Table 4.1. All the fuctions will be evaluated within the specific set values in order to ensure consistency. The purpose of the comparison is to check the performance of the improved particle optimization on different benchmark functions to understand if it is worth to use such approach for getting better performance in further studies.

Table 4.1 : The parameters used in benchmark functions.

Function Name	Dimensin n	Range $[x_{min}, x_{max}]$	Optimal Value
Sphere	30	[-100, 100]	0
Rosenbrock	30	[-2.048, 2.048]	0
Rastrigrin	30	[-5.12, 5.12]	0
Griewank	30	[-600, 600]	0

These numerical experiments are used to compare three particle swarm optimization algorithms including SWPSO as improved PSO, WPSO as standart PSO and SWPSO-Random as randomly changing inertia weight improved PSO. The population size is set to 30 and maximum number of iteration is set to 5000 in all simulation. The self confidence factor c_1 and the swarm confidence factor c_2 are set to 2. The inertia weight *w* decreases linearly from 0.95 to 0.4 with constant *k* = 1. The fitness evolutionary curves of four benchmark functions for all three algorithm are depicted below.

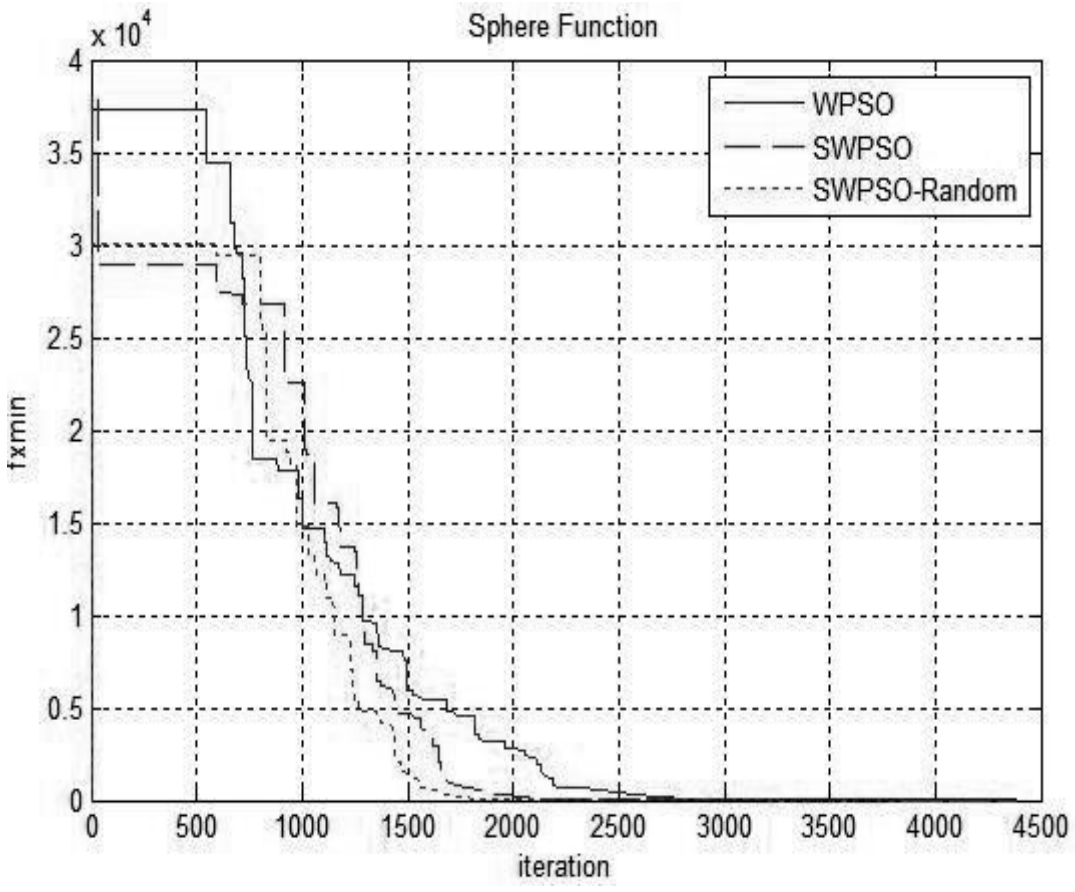

Figure 4.6 : The fitness evolutionary curve of sphere function.

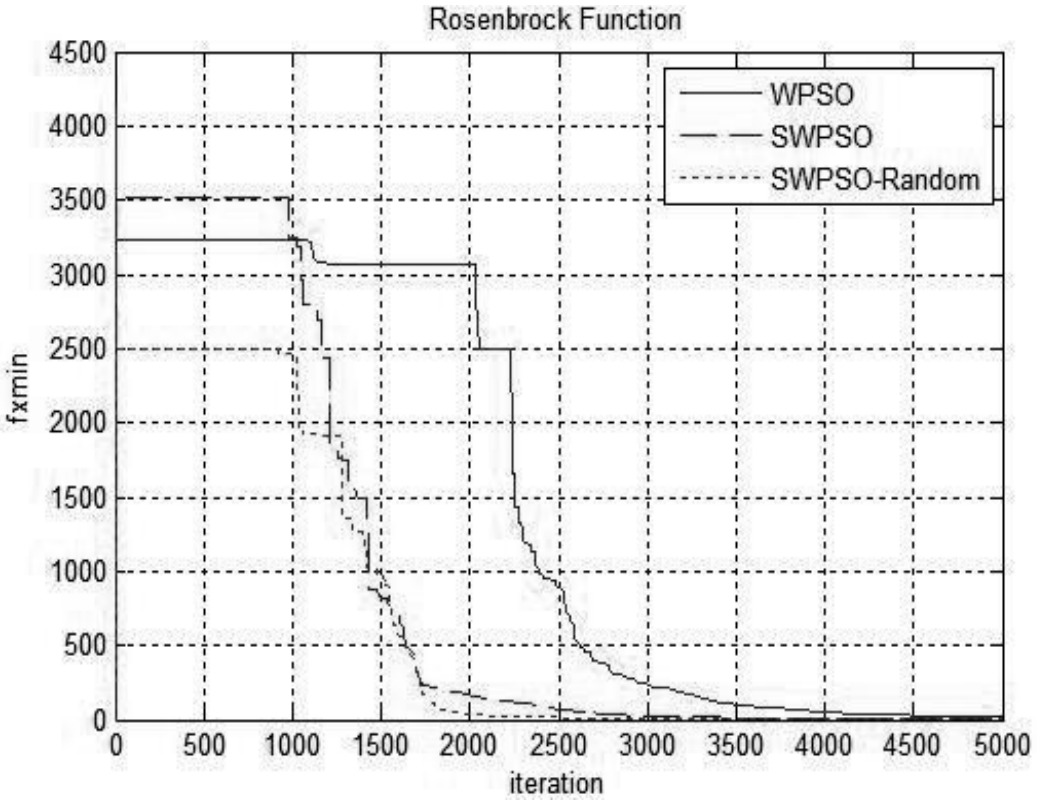

Figure 4.7 : The fitness evolutionary curve of rosenbrock function.

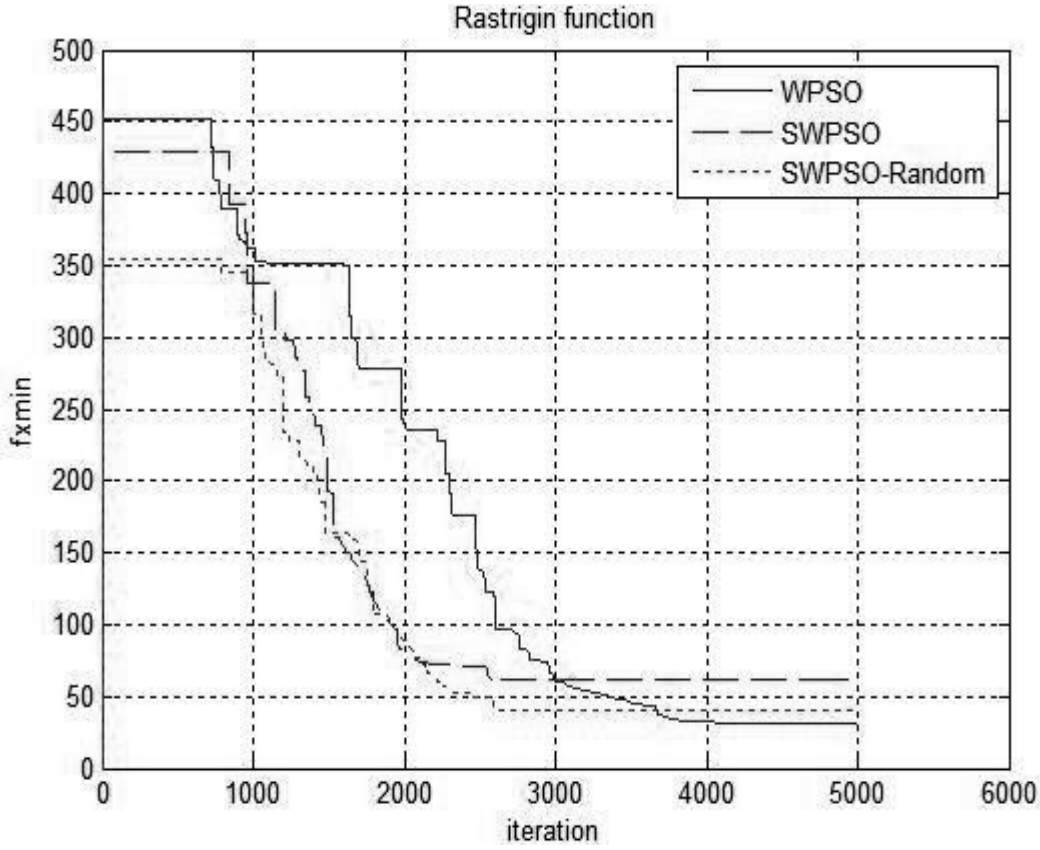

Figure 4.8 : The fitness evolutionary curve of rastrigin function.

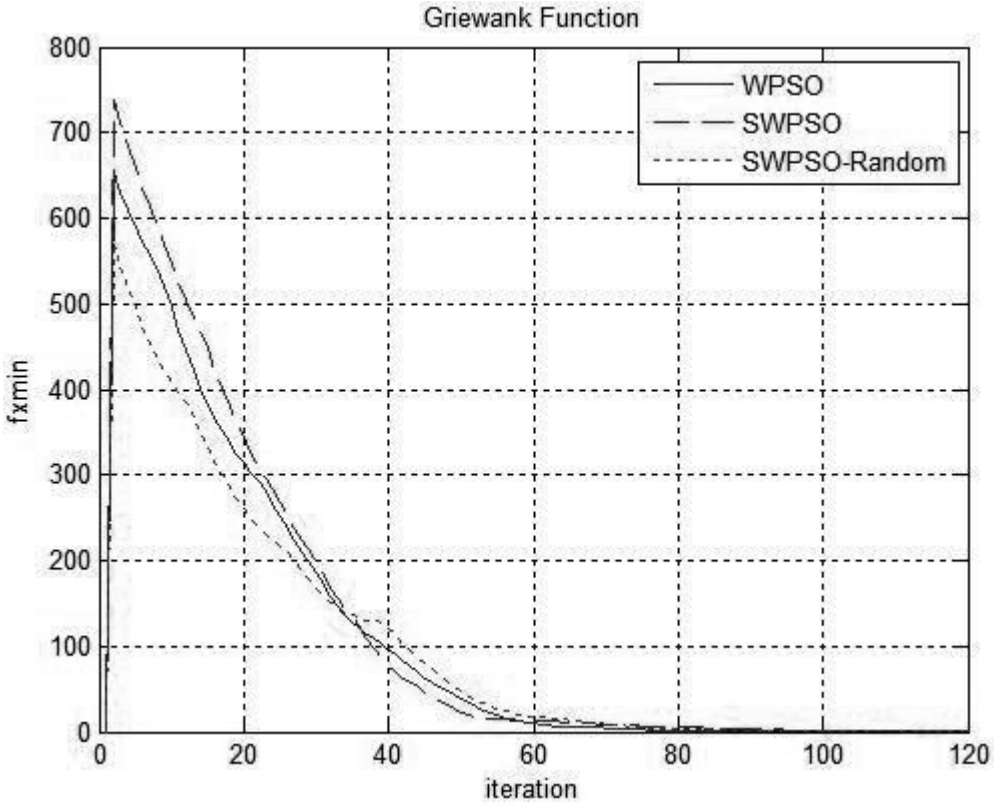

Figure 4.9 : The fitness evolutionary curve of griewank function.

As an interpretation of the figures, we can conclude that SWPSO-Random has the best performance on the convergence rate and convergence precision compared to the standart WPSO and SWPSO. We can also conclude that SWPSO has better performance with respect to standart PSO. Therefore, in the remaing part of the study, SWPSO-Random will be used for evolution for other simulations for further topics. The status of the 30 particles as the simulations run are depicted in Figure 4.10. According to this figure, it can be easily understood that while the algorithm runs, particles tends to attain optimum values to ensure global optimum of the fitness function. For example, when the algorithm runs to approximate the sphere function, all the particles are set to 0 finally which makes the fitness function converge to 0.

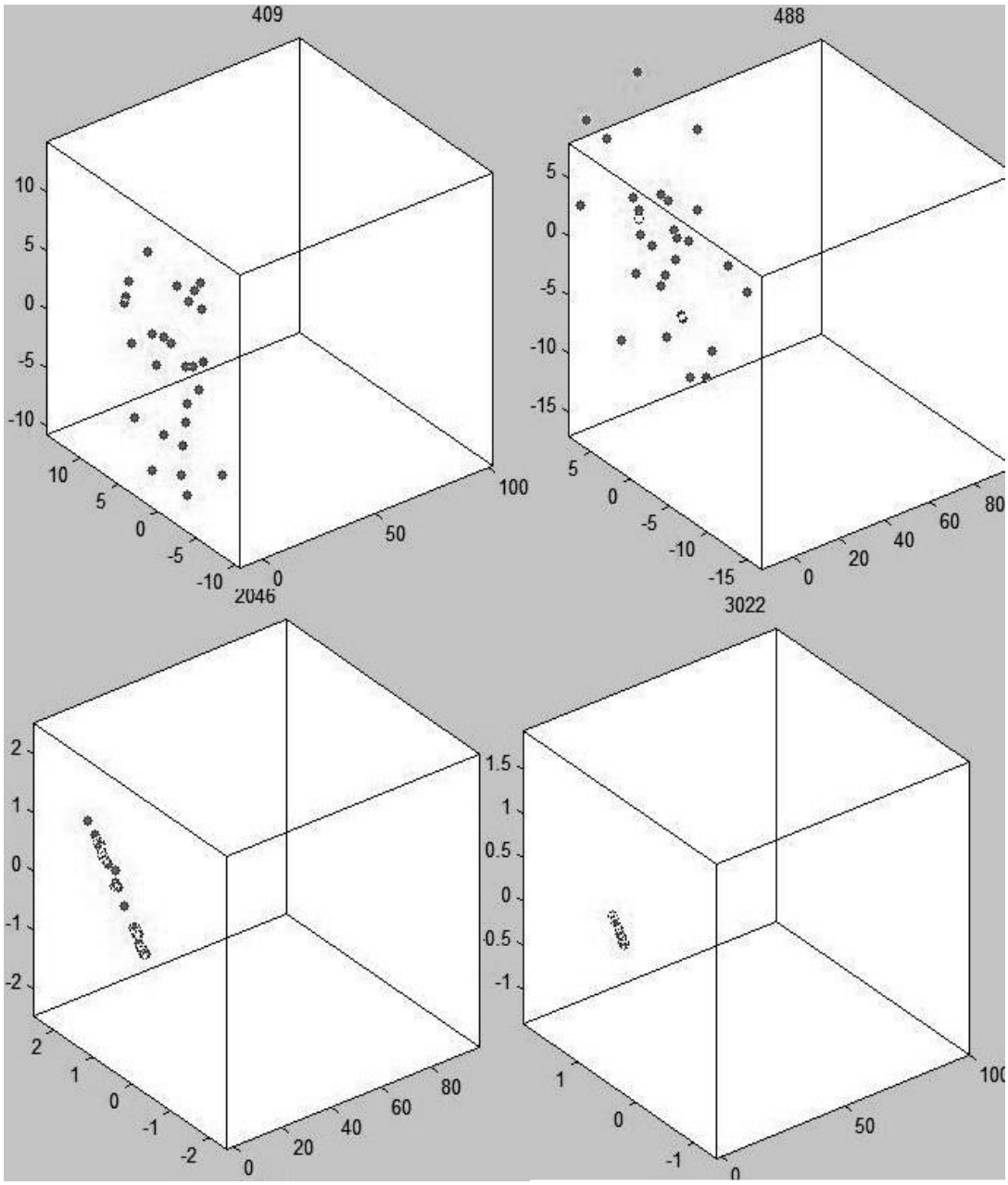

Figure 4.10 : The particle placement while algorithm run.

4.4.2 Partical Swarm Optimization and Genetic Algorithm

Particle swarm optimization is similar to the genetic algorithm in such a way that both of these evolutionary algorithms are based on population search methods. Particle swarm optimization and genetic algorithm move from a set of points to another set by using a combination of deterministic and probabilistic rules. The genetic algorithm has the ability to solve optimization problems for highly nonlinear systems and used in today's academia and industries. In spite of its ability for solving such complex systems, it has a drawback of being expensive in computational cost and it needs more memory for execution. According to recent researches, particle

swarm optimization has the same effectiveness in finding the global optimal solution as genetic algorithm but exhibits significantly better computational efficiency and performance. Particle swarm optimization and genetic algorithm are able to arrive at solutions with the same quality but particle swarm optimization is less computational-expensive and needs small memory than the genetic algorithm in general. Particle swarm optimization offers more computational savings for unconstrained nonlinear problems with continuous design variables as well. In all the applications for the rest of this study, improved particle swarm optimization is used. Parameters of the particle swarm optimization algorithm can be varied with respect to implementation on simulations [8].

5. OPTIMAL PARAMETERS OF THE FUZZY-PID CONTROLLER

As mentioned earlier PSO is able to generate high-quality solutions within shorter computation time and has more stable convergence characteristics than other stochastic optimization methods. In this study, PSO is used for tuning the proportional integral derivative controller gains and the crips values of Takagi-Sugeno rule base for a fuzzy PID controller and the results are presented. While the crips values of the rule base of the Takagi-Sugeno model are optimized using particle swarm optimization technique, fuzzy inference mechanism is used for specifying the consequent values of the controller output. The performance of the fuzzy PID autotuner has been well achieved by the use of the particle swarm optimization for optimizing the crisp values of the rulebase on different systems. In this chapter, first order plus dead time (FOPDT), second order plus dead time (SOPDT) and finally second order oscilattory process model have been in simulations and the results of using particle swarm optimization for parameter tuning process are given.

5.1 Automatic Tuning : A Fuzzy – PSO Approach

PID controller has been the most commonly used controller for the applications of process control. These controllers have solved most of the control problems in process control. PID controller has survived during the course of time even though the control technology developed more sophisticated methods. The most important effects of PID controllers in process controls are elimination of steady state error and generation of adequate corrective signals through the derivative action.

PID controller captured much attention when the concept of automatic control first emerged. Nevertheless, unfortunately for a long time, researchers paid little attention to it because of the difficulty of tuning the three controller parameters by trial and error. Parallel to the increasing usage of the microprocessors, there has been a resurgence of interest in PID control. From then on, autotuning became a feature, which has been extensively used in PID controllers. Autotuners have much more advantages than others have since automatic tuning is faster than manual tuning and

it decreases the commissioning time for installation of new processes. In addition to manual tuning of the three parameters, step response and Ziegler – Nichols can also be used.

The autotuners that have the ability of self-tuning have many techniques for increasing the performance of control process. In this thesis, the fuzzy inference mechanism is based on the Takagi-Sugeno model and the rule base is derived with the help of particle swarm optimization algorithm based on error criterias such as IAE, ISE and ITSE. The advantage of the proposed method is that it is less sensitive to the knowledge of process experts as mentioned earlier. The parameters of PID and crisp values of Takagi Sugeno that are to be optimized have been obtained offline by the help of the implementation of fuzzy controller and particle swarm optimization technique together. This approach has been applied to FOPDT, SOPDT, and SOPDT oscillatory systems based on predetermined error criterias [10].

5.2 The Fuzzy PID Controller

The PID control law is generally of the form

$$u(t) = Kp.e(t) + Ki\int_0^t e(\tau)d\tau + Kd\frac{d}{dt}e(t) \qquad \textbf{(5.1)}$$

where; e is the system error between desired and actual outputs, u is the control force, K_p is the proportional gain, K_i is the integral gain and K_d is the derivative gain. In conventional usage of the PID, K_p will have the effect of reducing the rise time but not eliminating the steady state error, K_i will have the effect of eliminating steady-state error while making transient response worse and finally K_d will have the effect of increasing the stability of the system. The purpose of using the fuzzy-PID controller is to design a set of PID gains with rulebase and inference mechanism such that the system output response satisfies certain spesifications. In this thesis, the parameters of the PID and the crisp values of the rulebase of Takagi-Sugeno have been optimized by using improved particle swam optimization technique. Even though the proposed rule base is valid for all kinds of fuzzy logic controllers, in this study PID type of fuzzy logic controller will be used for various practical real time processes.

The error *e* and the change in error *de* have been used as the inputs of the fuzzy PID controller and the control signal (u) as the output of the PID type of fuzzy logic controller as illustrated in Figure 5.1.

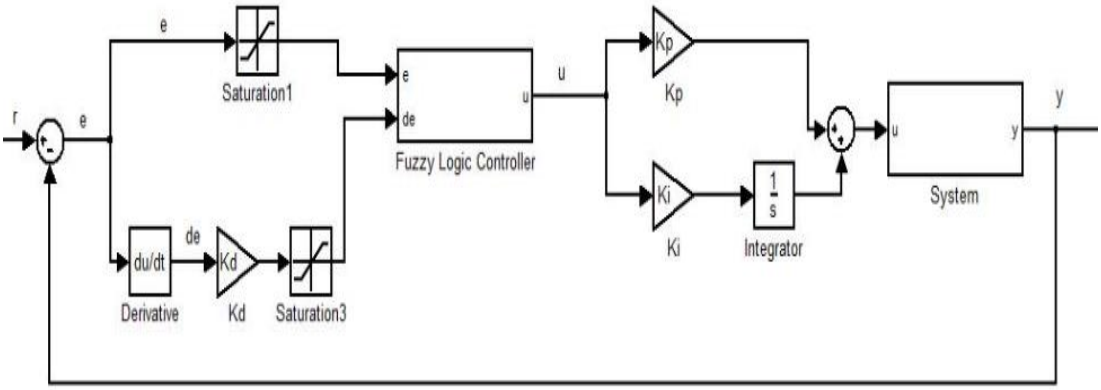

Figure 5.1 : An overall architecture of the fuzzy PID controller.

In order to be able to access the parameters of fuzzy-PID and the crisp values of Takagi-Sugeno rule base by using particle swarm optimization technique, it is a necessity to design the controller in such way that all the tuning parameters must be reachable from anywhere in the system. The number of necessary fuzzy sets and their ranges are designed based upon the study in [10]. The inputs of the fuzzy logic controller are the error *e* and the change in error *de* whereas the output variable is the control signal *u* after defuzzification to the system. The universe of discourse for the input variables is divided into three regions using the following linguistic variables as Negative (N), Zero (Z) and Positif (P). Triangular membership functions are used in the controller design. The universe of discourse of the output variables is assigned crisp values since a Takagi-Sugeno rule base is implemented. Input variables are defined on the normalized domain of [-1, 1] whereas output variables have no any boundaries since they affect the inference mechanism of the controller. The heights of the membership functions are one. The membership functions of inputs are illustrated in Figure 5.2. According to the figure x_1 refers to input of *e* and x_2 refers to input of *de*.

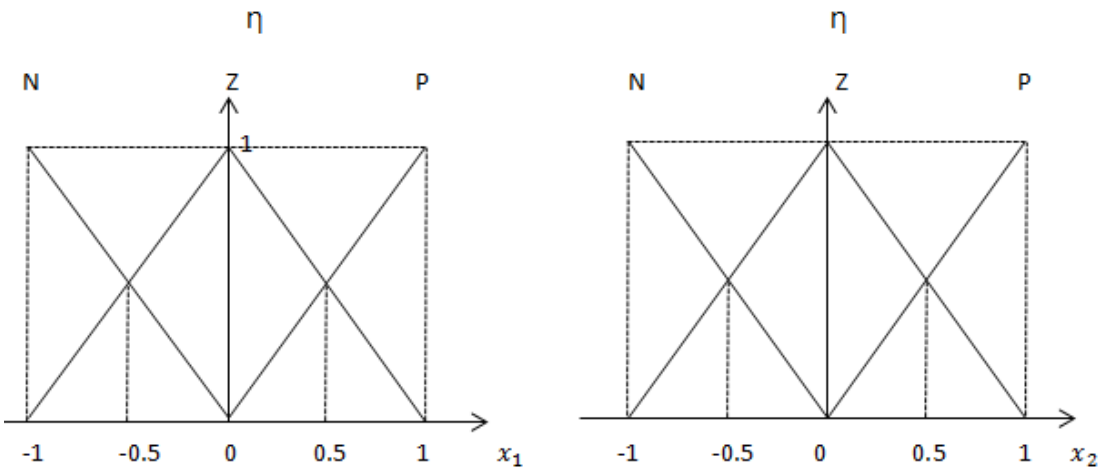

Figure 5.2 : Membership functions for e, x_1 and de, x_2.

During the design of the fuzzy controller, normalized process variables are fuzzified by fuzzification process and sent to fuzzy inference engine. The crisp output value, which is derived from fuzzy inference mechanism by weighted average method, is multiplied with K_i and K_p parameters to compute the control signal to the contolled systems. The fuzzy inference mechanism is illustrated in Figure 5.3 [12].

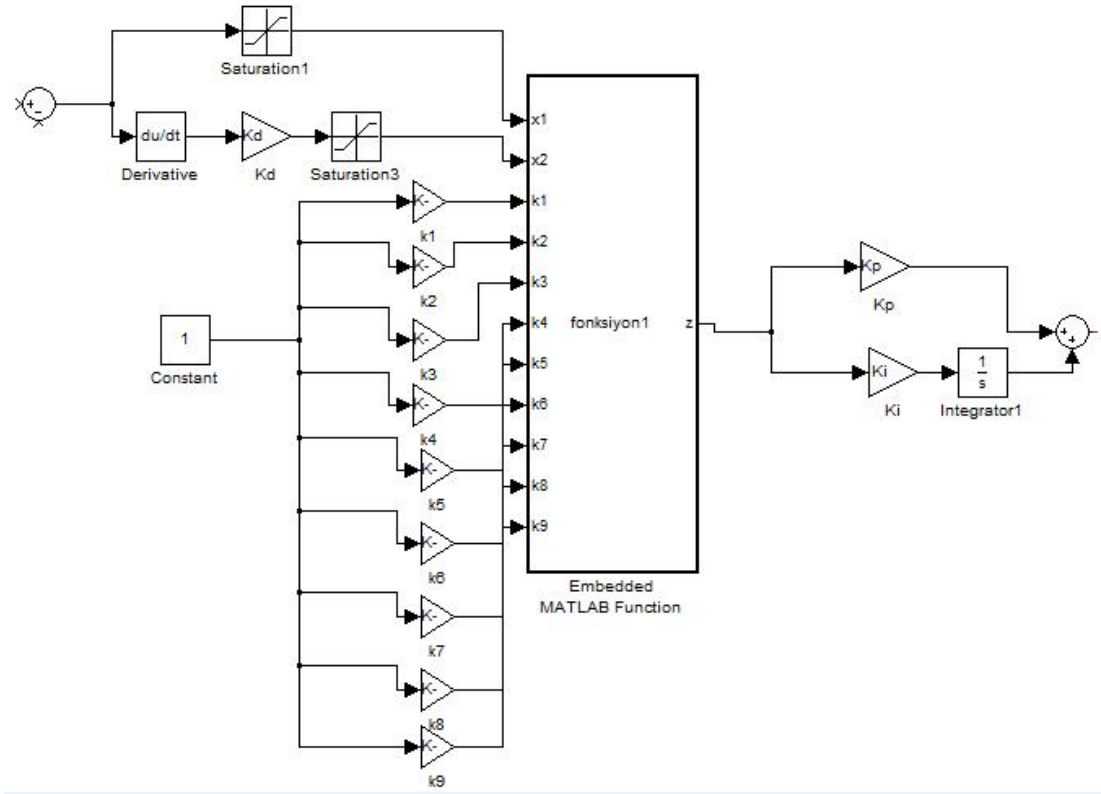

Figure 5.3 : Fuzzy inference mechanism.

The fuzzy rule base is in the form of fuzzy conditional statement as given below;

$$R_i = \text{If "e" } A_i \text{ and "de" } B_i \text{ then "u" } C_i.$$

It is composed of the antecedent (if-clause) and the consequence (then-clause). In this study, Takagi-Sugeno type of fuzzy rule base is used and the lookup table is derived as shown in Table 5.1.

Table 5.1 : Crisp values for rule base.

e,de	N	Z	P
N	k_1	k_2	k_3
Z	k_4	k_5	k_6
P	k_7	k_8	k_9

During the control process, normalized process variables *e* and *de* are fuzzified into fuzzy variables in order to compile through the fuzzy inference mechanism. The rule base is based on Takagi-Sugeno type model. According to Takagi-Sugeno modeling, even if A and B ancedent values are fuzzy values, the consequents should be crisp values. The control signal before multiplying the parameter of K_i and K_p will be crisps values as well. The proper rule base structure is very important from controller design point of view. In this study, particle swarm optimization tool is used for optimizing the 3x3 crisp values and the parameter of the PID controller K_p, K_i and K_d. Cumulative number of optimized parameters will be 12 [10].

5.2.1 Structure of the Takagi Sugeno rule base model

Fuzzification and defuzzification of the fuzzy controller are carried out differently for different values of normalized *e* and *de*. Four possible cases for the derivation of the fuzzy inference mechanism and the rule base are summarized below.

First Case: $e, de \in [-1, 0]$,

The singleton values in the lookup table when process variables are between [-1, 0] are N-N (k_1), N-Z (k_2), Z-N (k_4) and Z-Z (k_5).

Table 5.2 : Fired crisp values according to first scenario.

e,de	N	Z	P
N	k_1	k_2	k_3
Z	k_4	k_5	k_6
P	k_7	k_8	k_9

Fuzzy rule base is a combination of optimized control rules. Takagi Sugeno's minimum operation rule is used as fuzzy implication function. Weighted average method is used to defuzzify the inferred output as shown in **(5.2)**. Inferred output will be multiplied with K_i andK_p to compute the control signal.

$$z = \frac{k_1(-x_1)(-x_2) + k_2(-x_1)(x_2+1) + k_4(-x_2)(x_1+1) + k_5(x_1+1)(x_2+1)}{(-x_1)(-x_2) + (-x_1)(x_2+1) + (-x_2)(x_1+1) + (x_1+1)(x_2+1)} \quad \textbf{(5.2)}$$

Second Case: $e \in [-1\,, 0\,]$ and $de \in [0\,, 1\,]$,

The singleton values in the lookup table when *e* is between [-1, 0] and *de* is between [0, 1] are N-Z (k_2), N-P (k_3), Z-Z (k_5) and Z-P (k_6).

Table 5.3 : Fired crisp values according to second scenario.

e,de	N	Z	P
N	k_1	k_2	k_3
Z	k_4	k_5	k_6
P	k_7	k_8	k_9

Usign the same minimum operation rule and weighted average defuzzification method, the output of the inference mechanism is computed as:

$$z = \frac{k_3(-x_1)(x_2) + k_6(x_2)(x_1+1) + k_2(-x_1)(1-x_2) + k_5(1-x_2)(x_1+1)}{(-x_1)(x_2) + (x_2)(x_1+1) + (-x_1)(1-x_2) + (1-x_2)(x_1+1)} \quad \textbf{(5.3)}$$

Third Case: $e \in [0\,, 1\,]$ and $de \in [-1\,,0\,]$,

The singleton values in the lookup table when *e* is between [0, 1] and *de* is between [-1, 0] are Z-N (k_4), Z-Z (k_5), P-N (k_7) and P-Z (k_8).

Table 5.4 : Fired crisp values according to third scenario.

e,de	N	Z	P
N	k_1	k_2	k_3
Z	k_4	k_5	k_6
P	k_7	k_8	k_9

The output of the inference mechanism is computed as:

$$z = \frac{k_7(x_1)(-x_2) + k_8(x_1)(x_2+1) + k_4(-x_2)(1-x_1) + k_5(1-x_1)(x_2+1)}{(x_1)(-x_2) + (x_1)(x_2+1) + (-x_2)(1-x_1) + (1-x_1)(x_2+1)} \quad \textbf{(5.4)}$$

Fourth Case: $e \in [0\ ,1\ ]$ and $de \in [0\ ,1\ ]$,

The singleton values in the lookup table when process variables are between [0, 1] are Z-Z (k_5), Z-P (k_6), P-Z (k_8) and P-P (k_9).

Table 5.5 : Fired crisp values according to fourth scenario.

e,de	N	Z	P
N	k_1	k_2	k_3
Z	k_4	k_5	k_6
P	k_7	k_8	k_9

The output of the inference mechasim is computed as:

$$z = \frac{k_9(x_1)(x_2) + k_8(x_1)(1-x_2) + k_6(1-x_1)(x_2) + k_5(1-x_2)(1-x_1)}{(x_1)(x_2) + (x_1)(1-x_2) + (1-x_1)(x_2) + (1-x_2)(1-x_1)} \quad \textbf{(5.5)}$$

5.2.2 Implementation of Particle Swarm Optimization

In Chapter 4, the efficiency of the improved particle optimization technique was proved on some benchmark problems. In this study, improved particle swarm optimization will be used for all the simulations due to its effectiveness in optimization. Particle swarm optimization starts with the initialization of individuals similar to other artificial intelligence based heuristic optimization techniques. In a physical *n*-dimensional search space, the velocity and the position of individual *i* are represented as the vectors $X_i = (x_{i1}, \dots, x_{in})$ and $V_i = (v_{i1}, \dots, v_{in})$ in the PSO algorithm. $Pbest_i = \left(x_{i1}^{Pbest}, \dots, x_{in}^{Pbest}\right)$ and $Gbest_i = \left(x_{i1}^{Gbest}, \dots, x_{in}^{Gbest}\right)$ are the best position of individuals and their neighbours' best position, respectively. The velocity of individual *i* is derived under the following equation in PSO algorithm **(5.6)**.

$$V_i^{k+1} = w.V_i^k + c_1.rand_1 x\left(Pbest_i^k - X_i^k\right) + c_2.(1-w/2).rand_2 x(Gbest_i^k - X_i^k) \quad \textbf{(5.6)}$$

where; V_i^k is the velocity of individual *i* at iteration *k*, X_i^k is the position of individual *i* at iteration *k*, *w* is the inertia weight, c_1, c_2 is the weight factors, $rand_1, rand_2$ denotes random numbers between 0 and 1, $Pbest_i^k$ is the best position of individual *i* until iteration *k*, and best position of the group until iteration *k* is denoted as $Gbest_i^k$.

Each individual moves from the current position to the next one by modified velocity using the following position update equation **(5.7).**

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad \textbf{(5.7)}$$

During updating of velocity and position, the parameters of w, c_1, c_2 should be determined from the following equation.

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}}\ x\ iter \quad \textbf{(5.8)}$$

where, w_{max}, w_{min} are initial and final inertia weights, *iter* denotes current iteration number while running the simulation, $iter_{max}$ is the maximum iteration number. c_1, c_2 are the confidence factors and are always set to the same number to give particle memory influence and swarm influence equal weight. V_{max} and -V_{max}will be used to limit the velocities of the particles. The PSO algorithm is summarized as follows:

Step 1: Initialization of each individual.

Step 2: Setting of weights.

Step 3: Update of velocity, position and iterator.

Step 4: Update of *Pbest* and *Gbest*,

Step 5: Go to step two until stopping criteria as max *iter* or error goal has been satisfied

5.3 Simulation Results

The proposed algorithm for tuning the paramenters of a fuzzy - PID controllers have been applied to a number of process models. Simulations are performed for the first-order dead-time process model (FOPDT), the second-order dead-time process model (SOPDT) and at the end, second-order oscillatory process model to test the proposed algorithm. In order to evaluate the models presented here on the same basis or criterias regarding particle swarm optimization, parameters of the algorithm are taken as listed in Table 5.6 for all models [10].

Table 5.6 : Particle swarm optimization algorithm parameters.

Swarm Size	Number of particles	20
Iterations	Maximum iterations	250
Error Goal	Termination point	1e-10
Dimension	Coordinates of particles	12
c1, c2	Self , swarm confidence	2
w_start	Velocity weight at the begining	0.95
w_end	Velocity weight at the end	0.40
Vmax	Maximum velocity	10
Chi	Constriction factor	1
Objective Function - IAE	Integral Absolute Error	$\int \lvert e(t) \rvert \, dt$
Objective Function - ISE	Integral Square Error	$\int e^2(t) \, dt$
Objective Function - ITSE	Integral of time multiply Square Error	$\int te^2(t) \, dt$

5.3.1 First order plus dead time model

A fuzzy PID controller is designed for a linear first order plus dead time (FOPDT) system. The parameters of the fuzzy PID controller and the crisp values of the Takagi–Sugeno rule base are tuned by improved particle swarm optimization technique. The transfer function for the FOPDT system is given by:

$$T(s) = \frac{e^{-\theta s}}{\tau s + 1} \tag{5.9}$$

For this FOPDT model, simulation results are shown for θ = 0.2 s and τ =1.0 s. The sampling time is taken as 0.01 s. The nine crisp values of the rule base are calculated by the proposed particle swarm optimization technique implemented using matlab, the values are given below:

Table 5.7 : Crisp values for FOPDT – IAE.

e,de	N	Z	P
N	9.2954	4.0313	4.8648
Z	-0.0062	0	-1.4033
P	0.0561	-6.7924	20.0207

Overall structure of the FOPDT model is illustrated at Figure 5.4. According to the system, the parameters of PID and the crisp values of the rule base have been tuned offline to minimize the performance criteria given as integral absolute error (IAE). The fitness evaluation curve for integral absolute error while searching the optimum values is illustrated in Figure 5.5. According to the figure, the final value of the fitness function derived via particle swarm optimization is 0.3001 at the end of the 250 iterations. Total funtions evaluated is 5020 since algorithm runs 20 particles' fitness functions per one iteration. A change in the fitness value during search is shown in Table 5.8 with iteration number. Ensured tuned parameters are given to the simulink as parameters after optimization since offline tuning has been applied to find the optimum values. Observed system response, error and control signal can be seen in Figures 5.6, Figure 5.7 and Figure 5.8 respectively. The saturation block in the system is used for limiting the process error *e* and change of error *de* between [-1, 1]. The parameters of the PID which are derived from the optimization are K_d = 0.2667, K_p = -0.73804 and K_i = -0.73821. A linearly decreasing inertia weight is used for particle swarm optimization algorithm since this is the most efficient way to get rid of the local minimas according to the researches. For further studies, this model would be taken as reference for examining the efficiency of the program with changing the necessary values in terms of PSO parameters. Another version of the improved particle optimization algorithm can be applied to the program since the program has the flexibility for further studies. The simulation results are shown in the graphs of system response for FOPDT. The proposed parameter tuning method was successfully achived in terms of rise time, and settling time.It is to be stressed that reductions in rise time and peak deviation or overshoot are the primarly concerns for improved performance. However, actual reduction in IAE would also be based on the values of θ, τ and T as well.

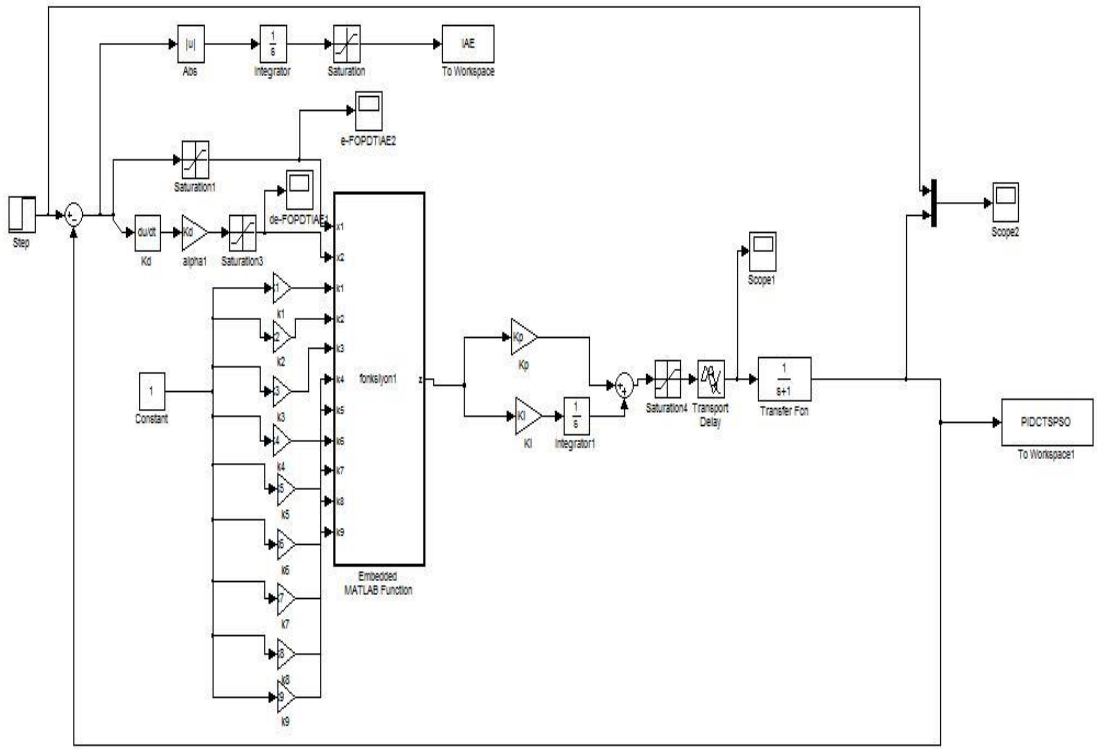

Figure 5.4 : The structure of FOPDT-IAE.

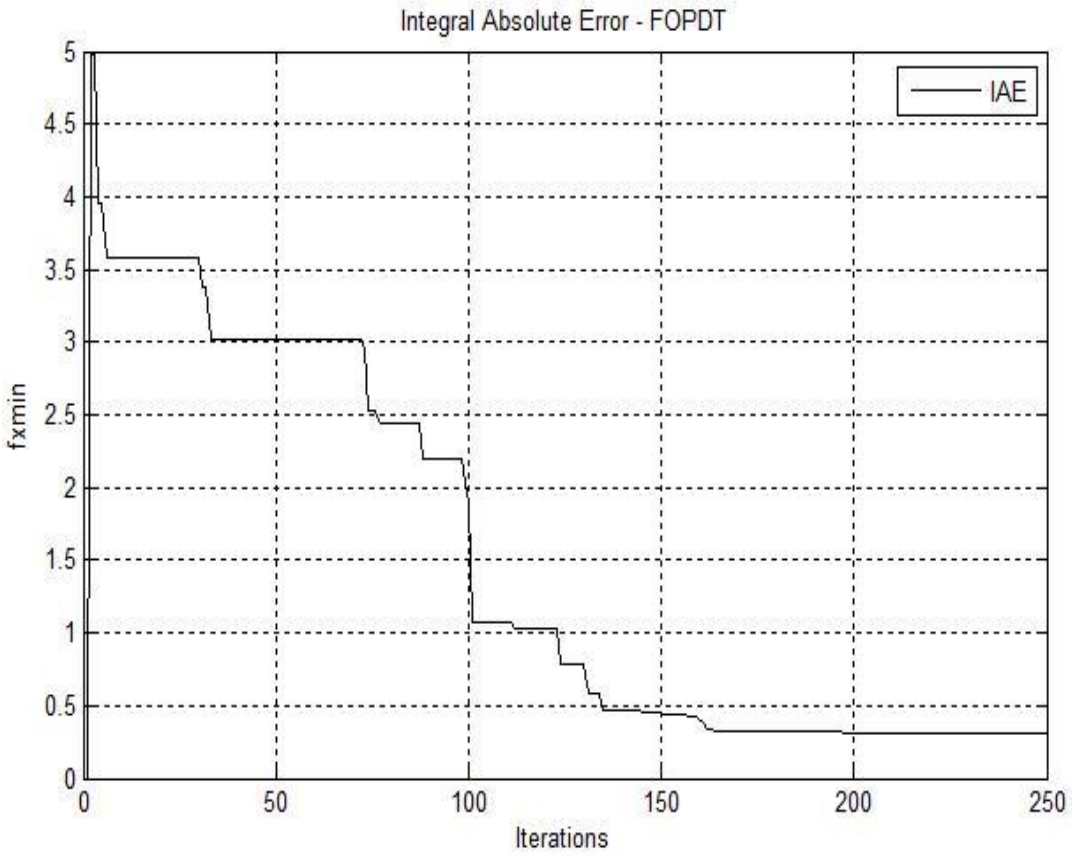

Figure 5.5 : The fitness evaluation curve of FOPDT-IAE.

Table 5.8 : Fitness functions of IAE during execution.

Iterations	fGBest	fevals	Iterations	fGBest	fevals
10	3,2801	220	130	0,4	2620
20	3,2801	420	140	0,383	2820
30	2,938	620	150	0,354	3020
40	2,938	820	160	0,346	3220
50	1,99	1020	170	0,331	3420
60	1,99	1220	180	0,324	3620
70	1,99	1420	190	0,303	3820
80	1,7633	1620	200	0,303	4020
90	0,81003	1820	210	0,302	4220
100	0,40878	2020	220	0,301	4420
110	0,40878	2220	230	0,301	4620
120	0,40584	2420	240	0,301	4820
130	0,39982	2620	250	0,300	5020

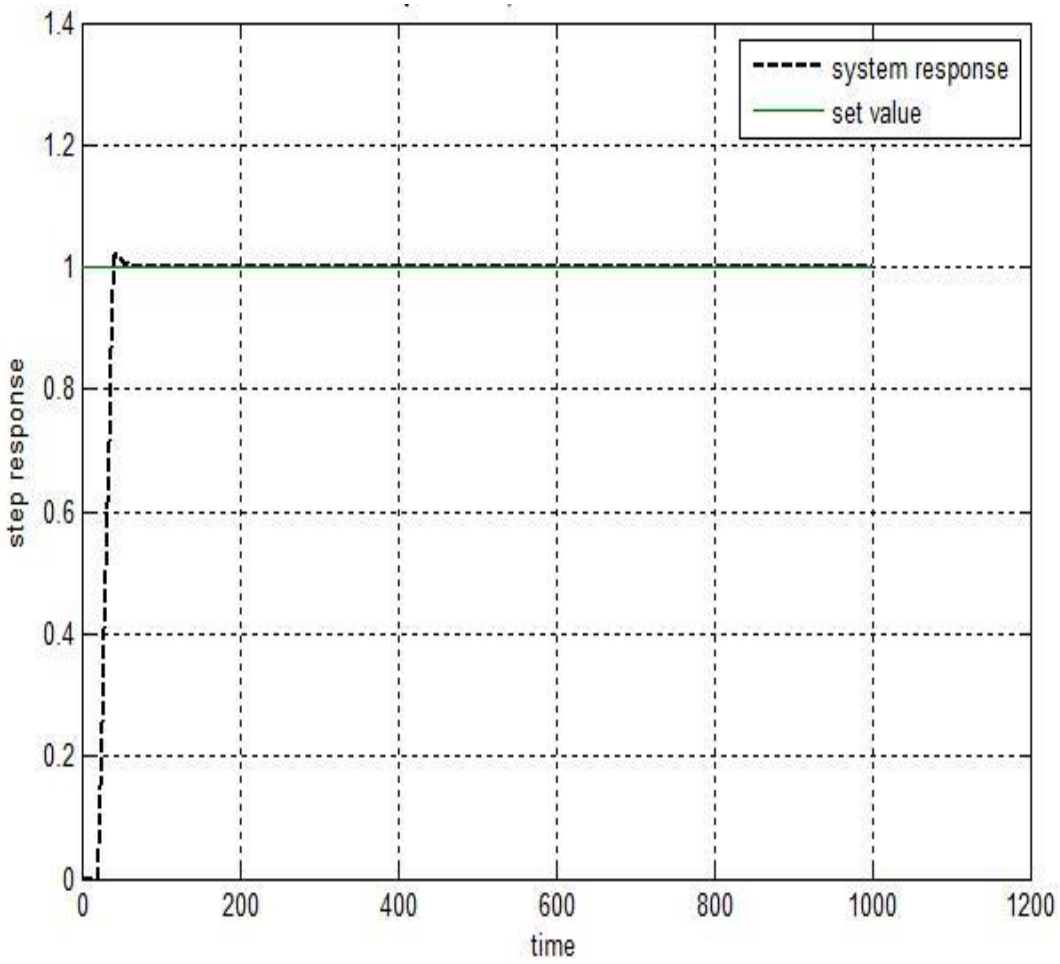

Figure 5.6 : System response of FOPDT-IAE.

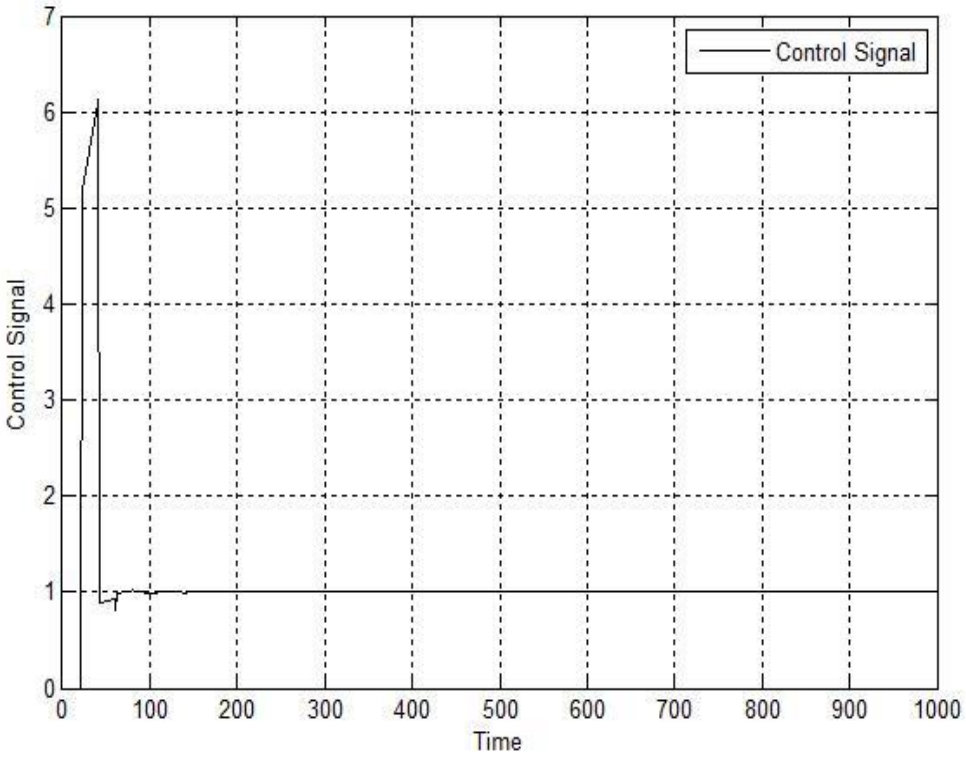

Figure 5.7 : Control signal for FOPDT.

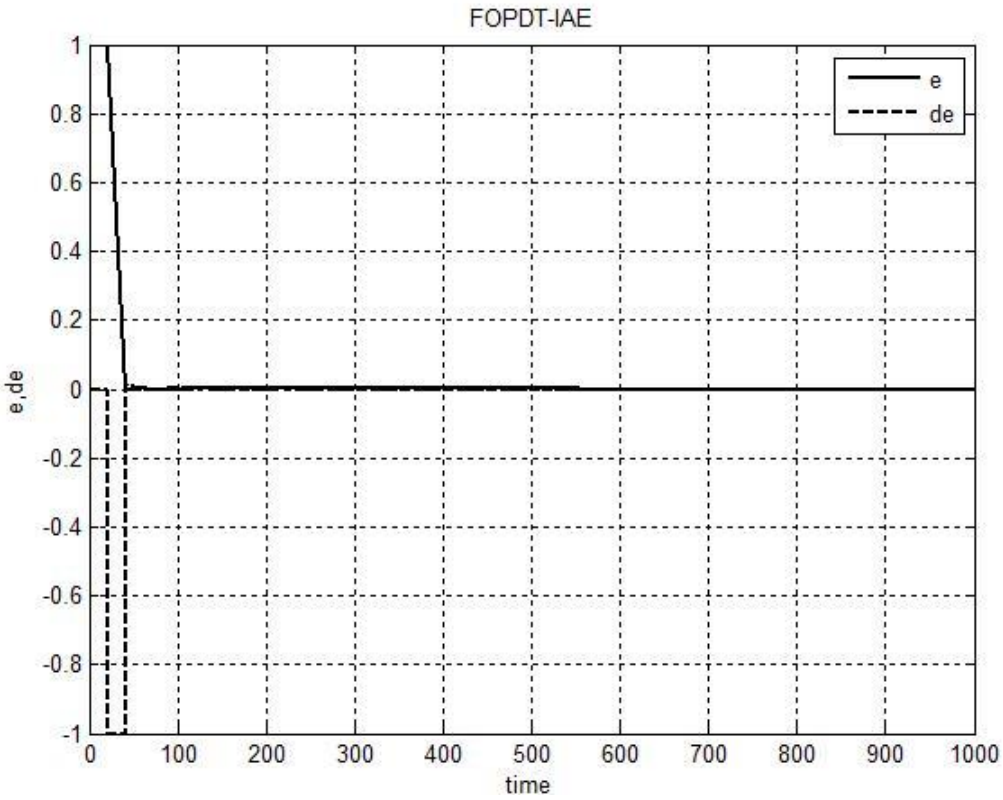

Figure 5.8 : Error and derivative of error for FOPDT – IAE.

5.3.2 Second order plus dead time model

A fuzzy PID controller is designed for a linear second order plus dead time (SOPDT) system specified by equation **(5.10)**. The parameters of the fuzzy PID controller and the crisp values of the Takagi–Sugeno rule base are tuned by improved particle swarm optimization technique. The proposed algorithm is implemented using integral absolute error (IAE), integral square error (ISE) and finally integral of time multiply square error (ITSE) in this simulation study. At the end, assessments for the performances with different fitness functions evaluation and the impact on system response, error and control signal has been discussed [10,13,14].

$$T(s) = \frac{e^{-\theta s}}{(\tau_1 s + 1)(\tau_2 s + 1)} \qquad \textbf{(5.10)}$$

where, $\theta = 0.3\ s\ ve\ \tau_1 = 0.4\ s\ and\ \tau_2 = 0.5\ s$. Sampling time is 0.01 sec. The 3x3, 9 crisp values of the rule base have been calculated based on three different objective functions with the proposed methods indicated below by using Matlab and particle swarm optimization technique.

Table 5.9 : Crisp values for SOPDT – IAE.

e,de	N	Z	P
N	6.3178	-0.0471	3.3345
Z	0.0885	-0.0002	1.2814
P	-2.7073	6.7767	5.4759

Table 5.10 : Crisp values for SOPDT – ISE.

e,de	N	Z	P
N	-6.31924	-0.22271	-12.8071
Z	-0.89102	0.065926	-0.26551
P	1.938938	-9.43175	8.051075

Table 5.11 : Crisp values for SOPDT – ITSE.

e,de	N	Z	P
N	-9.64815	-0.55859	-6.9098
Z	-0.60387	0.00236	-1.68627
P	8.824716	-13.2785	7.550065

Since the structure of the algorithm is heuristic, it is understandable to see the different values in the Takagi-Sugeno rule base with respect to different performance criteria. Overall structure of the model is illustrated at Figure 5.10 with difference based on performance indexes. The structures of the three different objective functions are depicted below at Figure 5.9. According to the sytem, the parameters of PID and the crisp values of the rule base have been tuned offline for minimizing the performance criteria given as integral absolute error, integral square error and integral time multiplied by square error. The fitness evaluation curves for integral absolute error, integral square error and integral time multiplied by square error while searching the proper values are illustrated in Figure 5.11. It is observed that, among the performance criterias, ITSE is the most successful criteria to find the minimum fitness function by offline tuning. Computed tuned values are given to the system after optimization. The saturation block in the system is used for limiting the process variables between [-1, 1].

Table 5.12 : Kp, Ki, Kd and Min values.

Objective Function	Kp	Ki	Kd	Min Fitness Funct
SOPDT-IAE	0.523805	0.570019	2.212371	0.5859
SOPDT-ISE	-0.59097	-0.38646	-5.11098	0.4705
SOPDT-ITSE	-0.38068	-0.23244	3.557233	0.11954

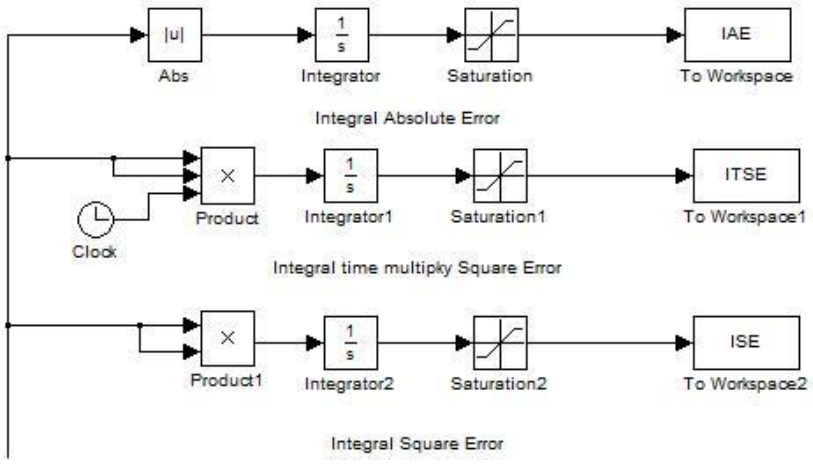

Figure 5.9 : Performance indexes.

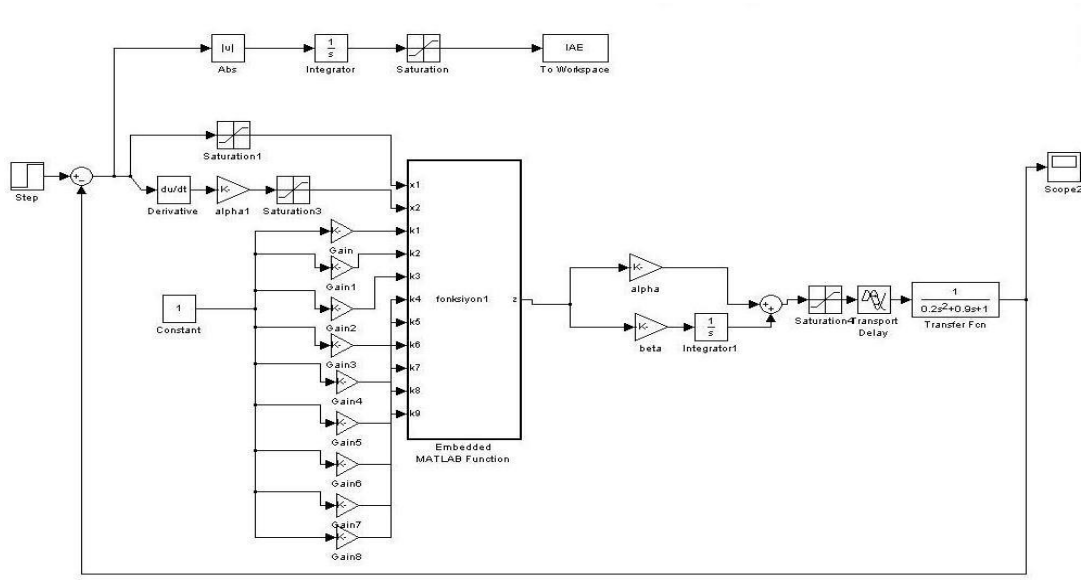

Figure 5.10 : The structure of SOPDT-IAE,ISE,ITSE.

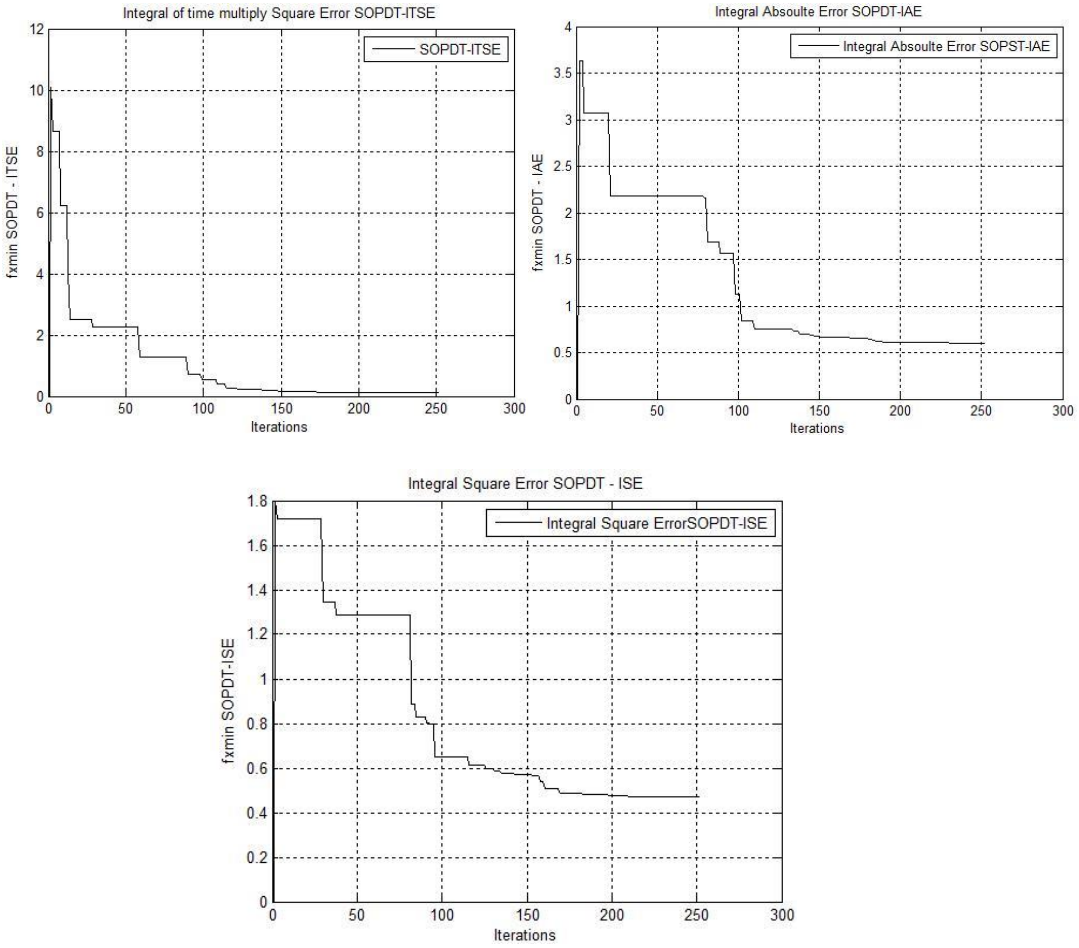

Figure 5.11 : The fitness evolutionary curves of IAE, ISE, ITSE.

Optimized parameters and the related fitness functions for three different performance criteria are listed in Table 5.12. Linearly decreasing inertia weight is used for particle swarm optimization algorithm since this is the powerfull method among other improved models. Particle placements as the algorithm runs are shown below with respect to different performance criterion.

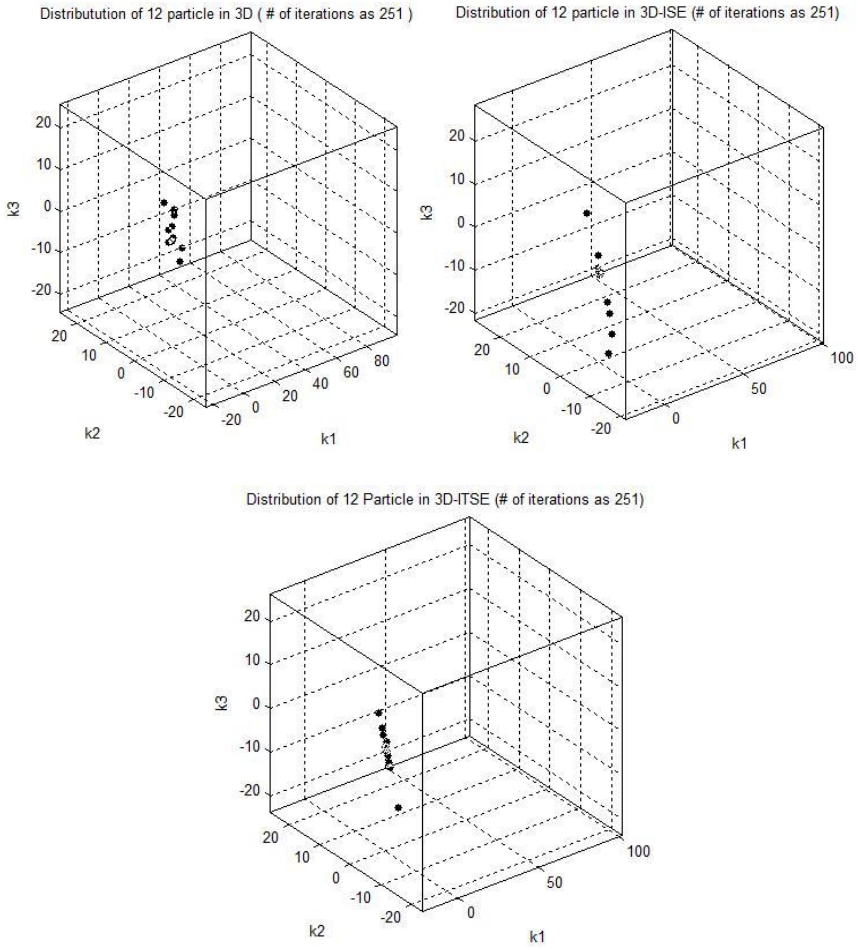

Figure 5.12 : Particle placement for SOPDT-IAE,ISE,ITSE.

As it can bee seen above from particle placement for second order plus dead time system with different fitness criterion, all 20 particles are getting grouping partially. The 3 dimensions of the cubic are k_1, k_2 and k_3 of the first 3 crisp values in Takagi – Sugeno rule base. In very few iterations, the entire 20 individuals are seen to be clustered within the tiny circle surrounding the goal and finally landing on the target. The performance of ITSE fitness criteria can also be seen better in particle placement. The step responses of the three different performance indexes are depicted below.

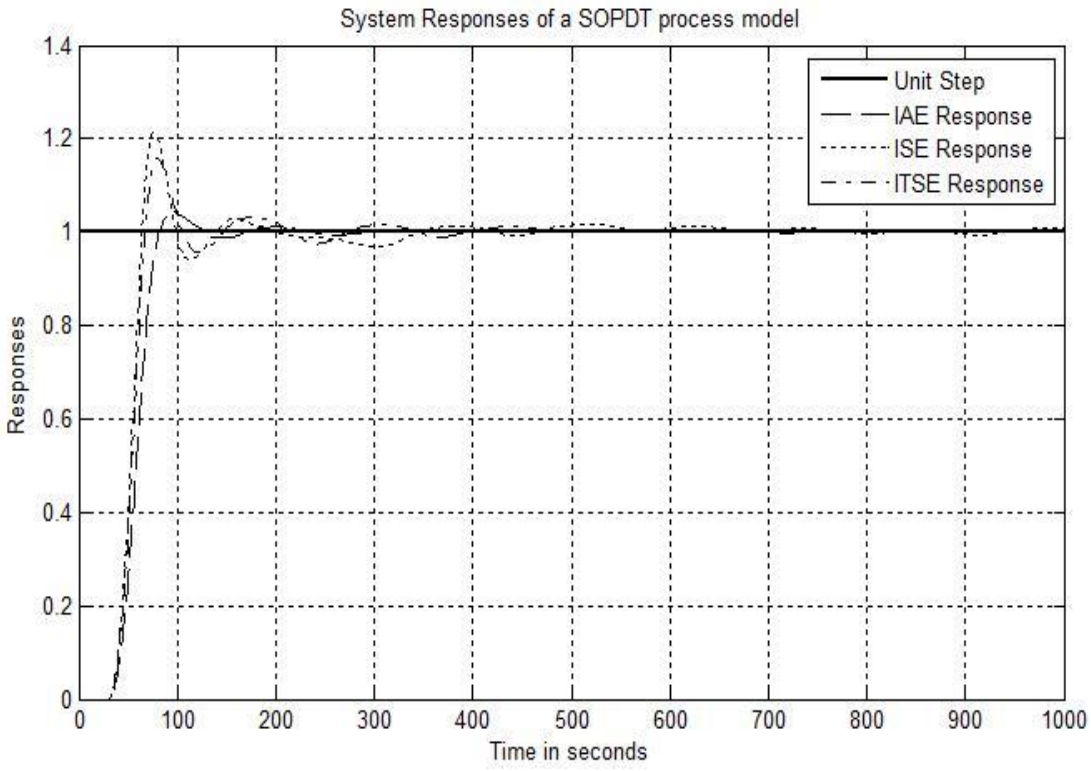

Figure 5.13 : Response of a SOPDT process model.

In Table 4.13, the comparison of performance with the SOPDT monotone process model is shown. Different error criterias are chosen and it has been observed that in all cases the proposed technique produces better results than conventional control methods. Among the different performance indexes, IAE model produces better result than the others since maximum overshoot $M_p(\%) = 5.5$, settling time $t_s = 6\ s, t_r = 0.8\ s$.

Table 5.13 : Performance analysis for the monotone SOPDT.

Objective Function	Mp (%)	t_s(s)	t_r(s)	Min Fitness Funct
SOPDT-IAE	4.25	6	0.8	0.5859
SOPDT-ISE	22	10	0.6	0.4705
SOPDT-ITSE	15	8.7	0.6	0.11954

The control signals computed for second order monotone dead time system based on different performance indexes are shown below.

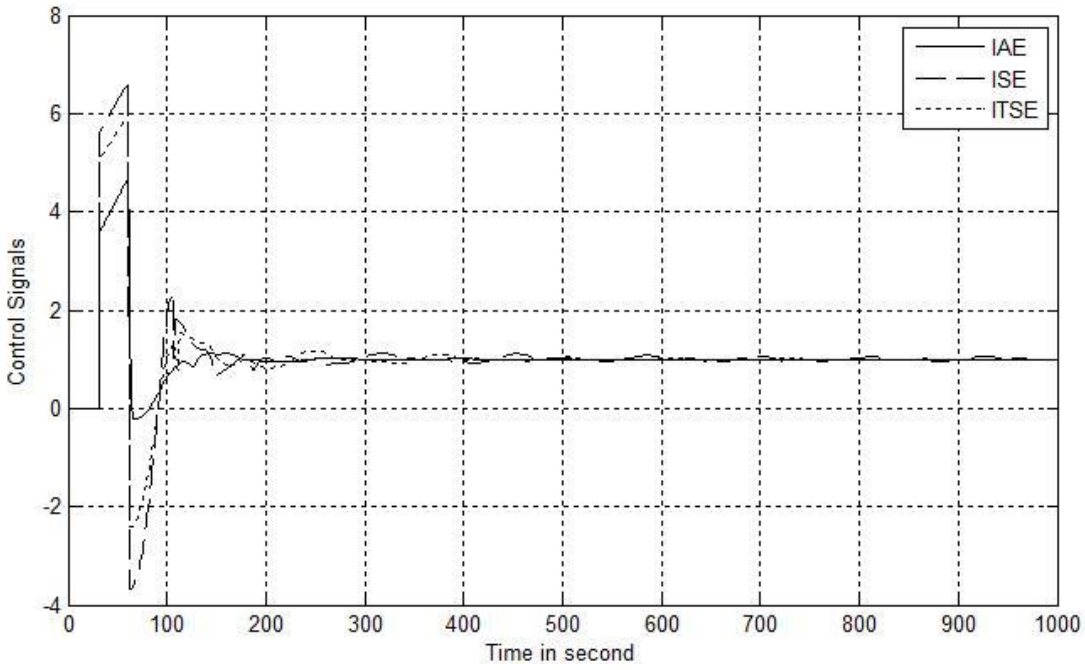

Figure 5.14 : Control signal for a SOPDT process model.

5.3.3 The second order oscillatory process model

A fuzzy PID controller is designed for a second order oscillatory process model with considerable dead time. This process model has been chosen becasuse this type of the process is hardly to control due to the inherent oscillatory nature. A linear second order plus dead time system (SOPDT2) is considered and the parameters of the fuzzy PID controller and the crisp values of the Takagi–Sugeno rule base are tuned by improved particle swarm optimization technique. The proposed algorithm is applied to the following system **(5.11)** [10,13,14].

$$T(s) = \frac{e^{-\theta s}}{s(s+1)} \qquad \textbf{(5.11)}$$

where, $\theta = 0.4$.. Sampling time is 0.01 sec. The 3x3 crisp values of the rule base have been derived from the proposed methods indicated below by using Matlab and particle swarm optimization toolbox. Overall structure of the model is illustrated at Figure 5.15. According to the sytem, the parameters of PID and the crisp values of the rule base have been tuned offline to minimize the performance criteria given as integral absolute error (IAE). Computed values have been entered to the system after optimization and system response, error and change in error are observed. The saturation block in the system is used for limiting the process variables between [-1,

1]. The parameters of the PID which derived from the optimization are K_d = -1.6032, K_p = -0.865 and K_i = -0.0343. Linearly decreasing inertia weight is used for particle swarm optimization algorithm since this is the most efficient way to get rid of the local minimas according to the researches. The observed variables can be seen from Figure 5.16, Figure 5.17 and Figure 5.18 respectively.

Table 5.14 : Crisp values for SOPDT2 - IAE

e,de	N	Z	P
N	2.0041	4.5428	-5.0768
Z	0.5766	0.047	-2.7698
P	2.5991	-5.3052	-7.4069

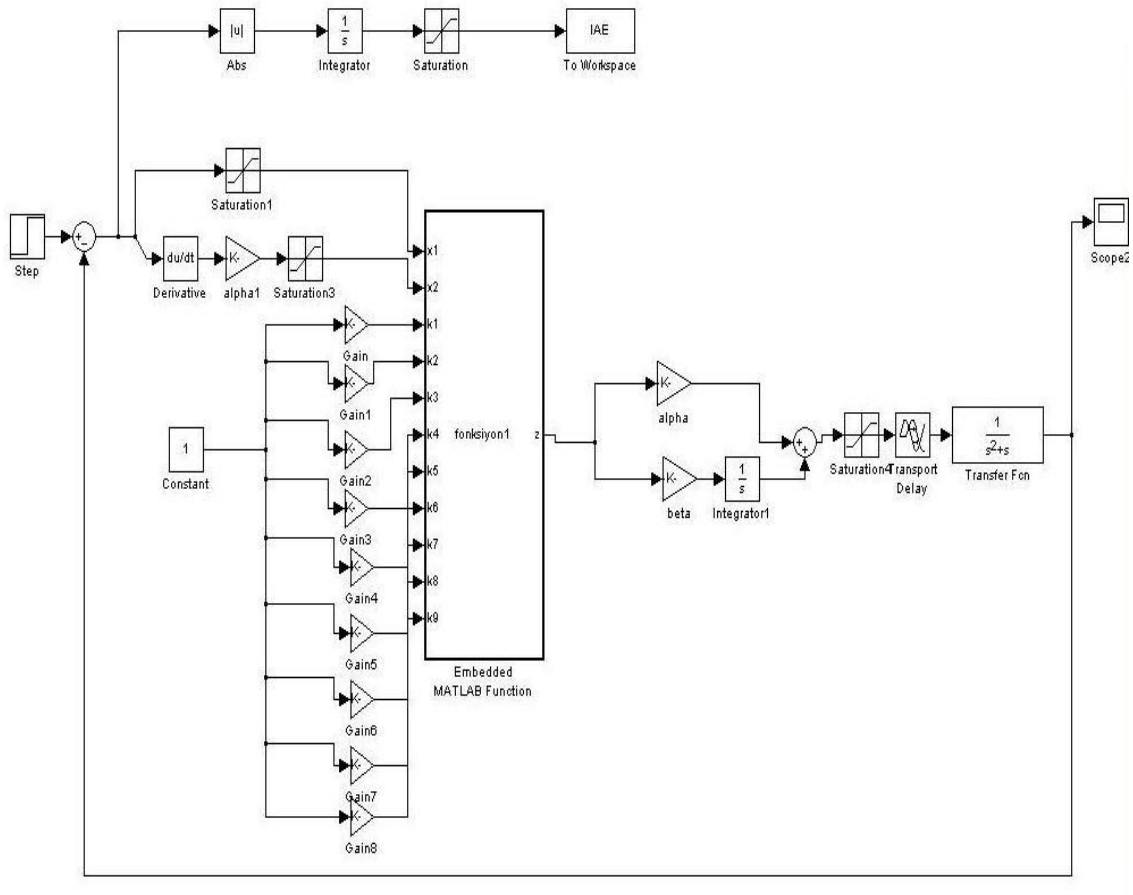

Figure 5.15 : The structure of SOPDT2-IAE.

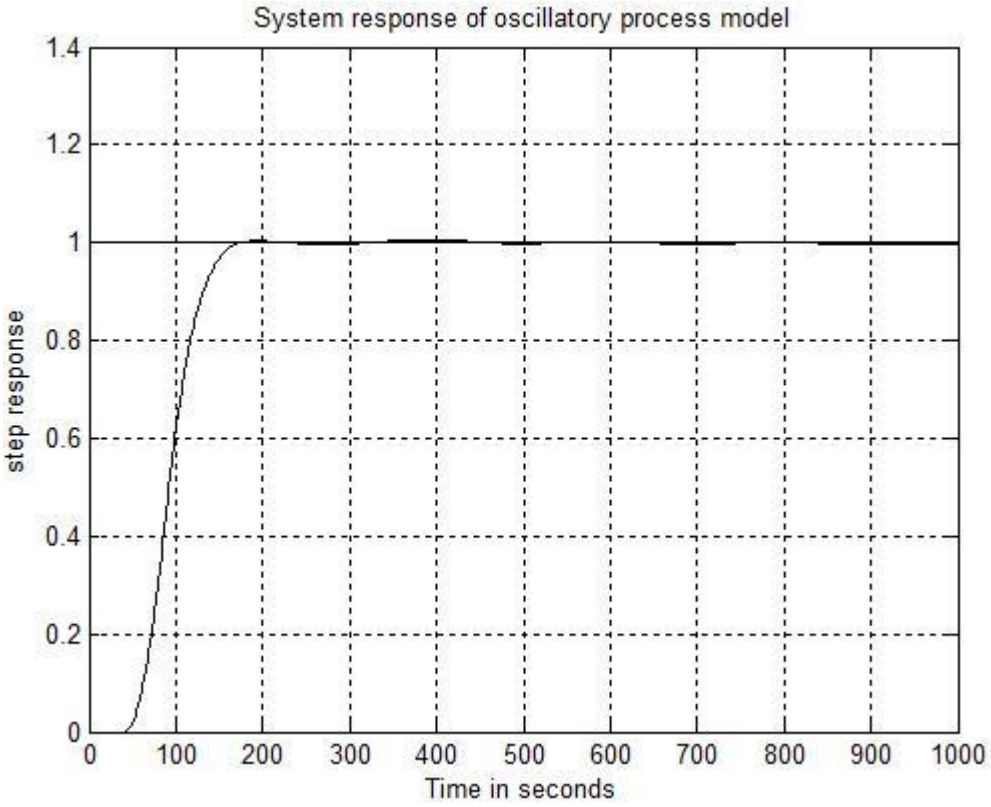

Figure 5.16 : System response of SOPDT2-IAE.

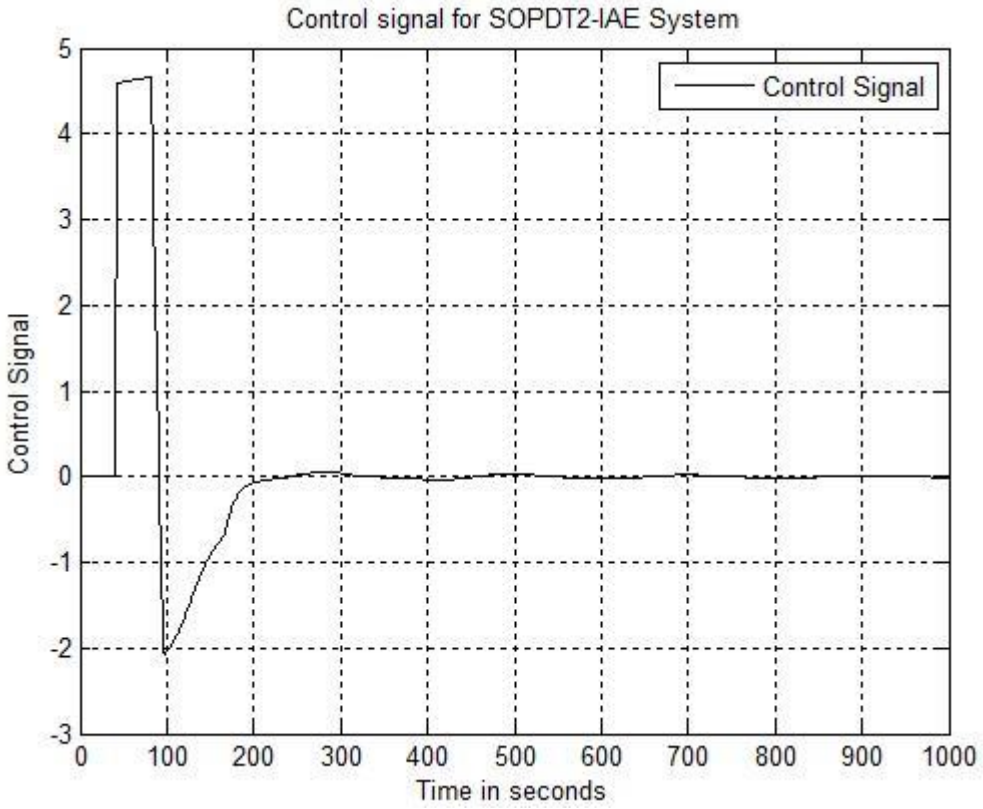

Figure 5.17 : Control signal for SOPDT2-IAE.

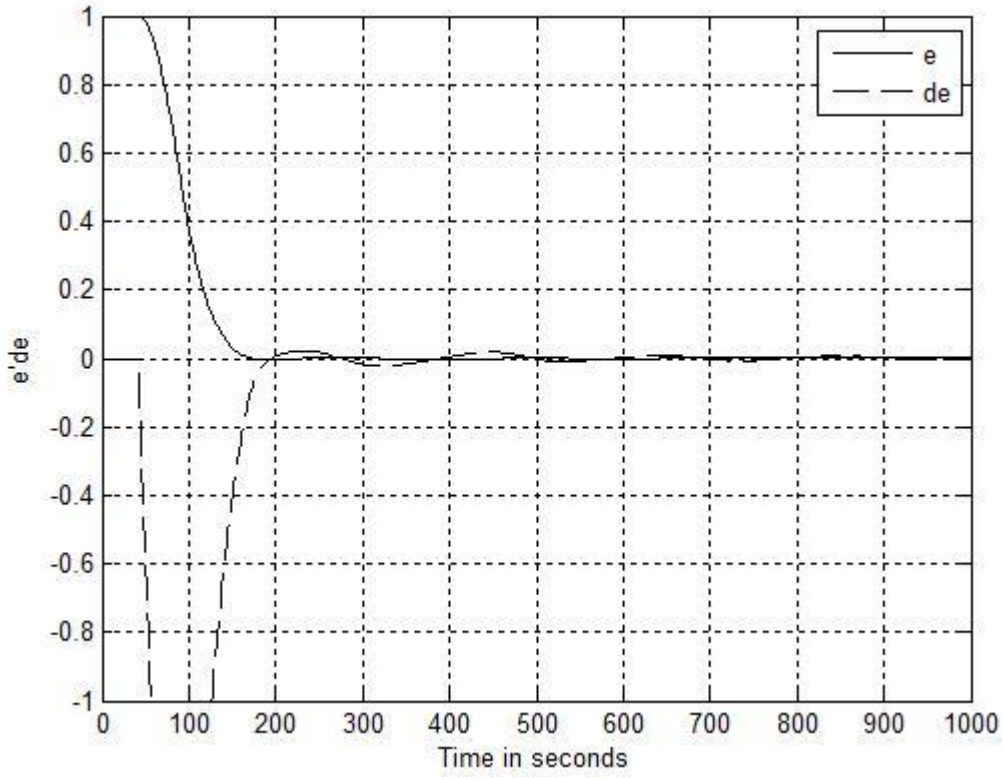

Figure 5.18 : Error and derivative of error for SOPDT2-IAE.

5.4 Conclusion

In this study, tuning the paramaters of a fuzzy controller, which was investigated using particle swarm optimization technique. Standart tuning techniques such as Zeigler - Nichols, Cohen-Coon, Set point weighting method used in optimization toolbox of MATLAB failed to provide good solution because of the presence of a number of local optima. Presented approach has been followed which utilized evolutionary computation for optimization of 12 floating-point elements. By using different performance indexes, proposed method has been applied to different systems to see the performance. It can be observed that same different performance indexes on identical systems yield different system responses and it can be concluded that in optimization, performance criterions are important to specify the parameters of fuzzy PID and crips values of the rule base. The algorithm is flexible to generalize in the sense that all of the parameters that have been tuned offline can be optimized online but it will take too time to finalize the simulation. It is expected that it would perform better in all types of applications especially in non-linear process systems where disturbances are likely to occur frequently.

6. COUPLED TANK PROCESS CONTROL BY FUZZY PID

The control of liquid level and flow rate between tanks has frequently been encountered in the process industries. The process industries often require the liquids to be pumped, stored or pumped to another tank. The purpose of the control in the process industries especially in tank systems is to control the level of fluid in the tanks as well as to regulate the flow between tanks. Level and flow control in tank systems are the most common practice in all chemical engineering systems. Petro-chemicals industries, paper making industries and water treatment industries are some vital industries where liquid level and flow control are implemented. The proposed approach in this study is to design a fuzzy PID controller for a nonlinear-coupled tank process system. In this approach, the design is based on minimizing the performance index by tuning the Takagi-Sugeno rule base values as well as PID parameters with particle swarm optimization algorithm to ensure robust control with minimum settling time and overshoot within specified operation conditions. The efficiency of this method is demosttrated on a coupled tank system where the connection between the tanks and flow in each tank are taken into account.

Industial processes are generally nonlinear systems. Classical control methods are being used to control the nonlinear systems by linearizing the model around a steady state point. The PID paramters are then determined for a chosen set point using frequency response method or using pole placement. Because of inheritance of the nonlinearty of the process systems, it is an inevitable need to monitor and adjust the loops due to parameter variations and operating conditions changes. If the process system is updated online, the controller should be effective over a wider range of operating conditions. This control strategy provides the basis for the self-tuning approach to adaptive control of nonlinear systems. Another control strategy for nonlinear system control is using an empirical model that is developed from experimental data. The most attracted strategy for developing nonlinear dynamic models from input/output date is artificial neural networks. It has been also widely used in process control applications.

6.1 Modeling The Nonlinear Coupled Tank System

It is crucial to understand the background of how the coupled tanks system works. This is the system modeling and it is very important part of control system analysis. To begin with look at a single tanks system in Figure 6.1 to understand how to derive the coupled tank system [15,19].

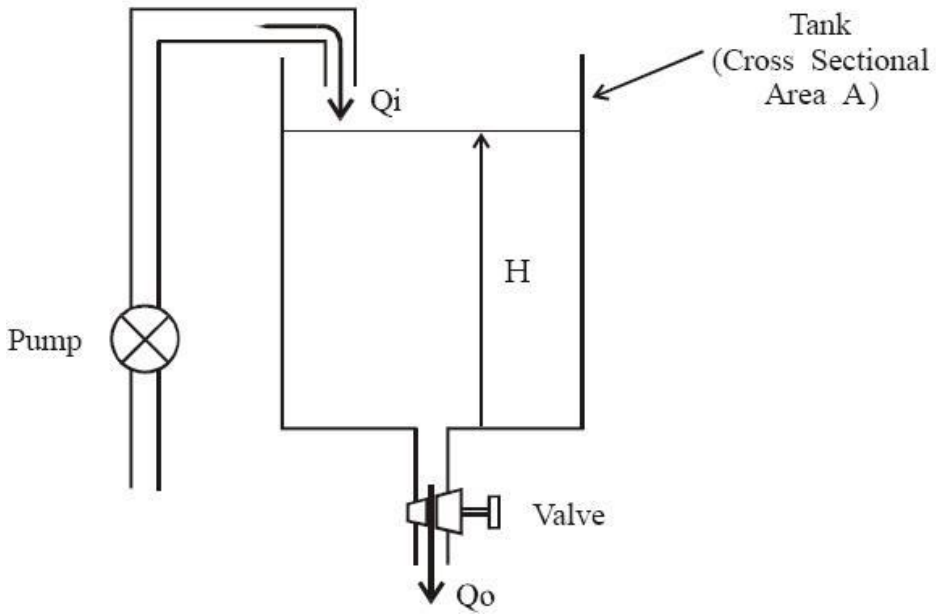

Figure 6.1 : A single tank fluid level system.

The system model is determined by the flow Q_i into the tank and the flow Q_0 leaving through the valve at the tank bottom. Using a balance of flows, the equation of the tank can be expressed as:

$$Q_i - Q_0 = A\frac{dH}{dt} \tag{6.1}$$

where, *A* is the cross-sectional area of the tank and *H* is the height of the fluid in the tank. If the valve behaves like an ideal sharp edged orifice, then the flow through the valve is related to the fluid level in the tank, *H* by the expression

$$Q_0 = C_d a\sqrt{2gH} \tag{6.2}$$

where, a is the cross sectional area of the orifice. C_d is called the discharge coefficient of the valve. This coefficient stands for fluid charecteristics, irregularities and losses in the system. g stands for gravitational constant, which is equal to

$980 cm/sec^2$. When the above equations are combined, we obtain the mathematical model that describes the system behavior of nonlinear single tanks system.

$$A\frac{dH}{dt} + C_d a\sqrt{2gH} = Q_i \qquad \textbf{(6.3)}$$

In the tank level problem the nonlinearty is important to solve in order to ensure robust control. The nonlinearity is smooth and can be made linear at a particular operating level *H* by using the slope of the nonlinearity at *H*. The system dynamics will change as the normal operating level varies. The tank level controller does not perceive the parameter changes in the model.When the two tanks are joined the coupled tank system shown in Figure 6.2 is obtained [15].

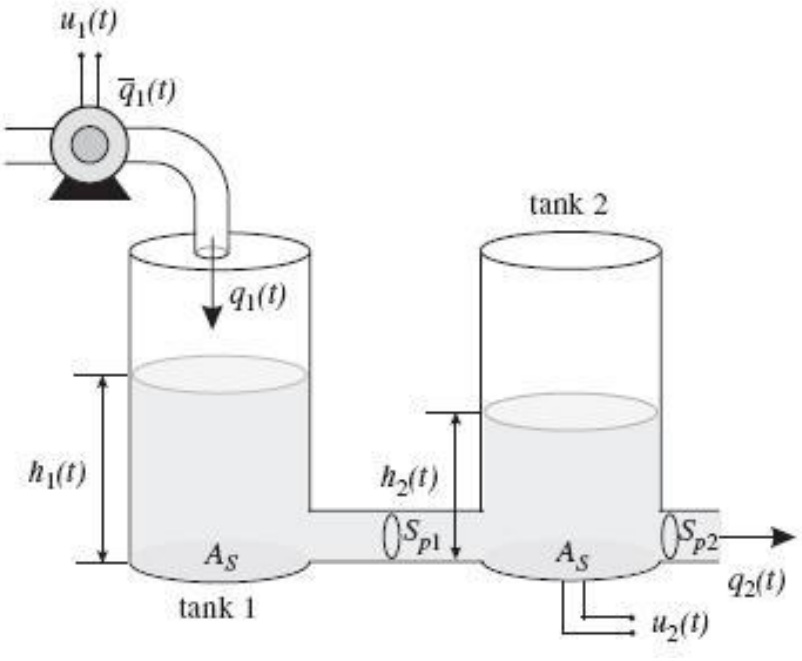

Figure 6.2 : A coupled tank fluid level system.

In particular, a coupled tank fluid level system is a well know benchmark problem for nonlinear control. It deals with a labarotory process using two tanks with fluid flow. In the simulation of fuzzy control with particle swarm optimization, two tanks are identical and cylindrical in shape, with a cross section of $A_s = 0.0154\ m^2$. The cross section of the connection pipes is $S_p = 3{,}6.10^{-5} m^2$ and the liquid levels in the two tanks are denoted by $h_1(t) and\ h_2(t)$ respectively. The supplying flow rates coming from pump to tank 1 are denoted by $u_1(t)$. There is an outflow from tank 2. By using balance equation and Toricelli's rule, following rule can be obtained.

$$\dot{h}_1(t) = (-K_{p1} sign(h_1(t) - h_2(t))\sqrt{2g|h_1(t) - h_2(t)|} + q_1(t))/A_s \quad \textbf{(6.4)}$$

$$\dot{h}_2(t) = (K_{p1} sign(h_1(t) - h_2(t))\ \sqrt{2g|h_1(t) - h_2(t)|} - K_{p2}\sqrt{2gh_2(t)})/A_s \quad \textbf{(6.5)}$$

where, $K_{p1} = a_1 S_{p1}$ and $K_{p2} = a_2 S_{p2}$ for outflow coefficients and g considered as gravity acceleration. For simplicity, a_1 *and* a_2 taken are 1. The overall parameters of the nonlinear-coupled tank system are listed in Table 6.1 [15].

Table 6.1 : System parameters of coupled tank.

Cross section areas of the tanks $[m^2]$	$A_s = 0.0154$
The cross sections of connection pipes$[m^2]$	$S_{p1}, S_{p2} = 3{,}6.\ 10^{-5}$
Gravity acceleration $[m^2/s]$	9.81
Height of the water [m]	h_1, h_2
Flow rate from pump to tank1 $[m^3/s]$	$q_1(t)$
Flow rate from tank2 $[m^3/s]$	$q_2(t)$
Outflow coefficients	K_{p1}, K_{p2}
Height of the tanks [m]	H=0.6

Voltage of the pump is also limited between 0 and 10 V. System input is the voltage of the electrical pump, which is, $u_1(t)$ and it produces the entrance flow as below.

$$\overline{q}_1(t) = K_u(1 + v_1(t))u_1(t) \quad \textbf{(6.6)}$$

where $K_u = 8.8x10^{-6} m^3/Vs$ coefficient of transition and $v_1(t)$ is transition error. The true entrance flow rate is written below.

$$q_1(t) = \overline{q}_1(t) + (1 - K_f)\overline{q}_1(t) \quad \textbf{(6.7)}$$

where, K_f is coefficient of error. The only measuring device for the output signal is the pressure sensor under the tank 2 that transfers the water height of the second tank $h_2(t)$ to voltage of output $u_2(t)$.

$$u_2(t) = K_h(1 + v_2(t))h_2(t) \tag{6.8}$$

where, $K_h = 16.667\, V/m$ coefficient of height to voltage. $v_2(t)$ is the transition error. The upper limits of the errors are, $\overline{v}_1 = \overline{v}_2 = 0.03$. When the dynamical equations are derived with the defined values, the nonlinear-coupled tank system mathematical model is obtained as given below [14].

$$\dot{h}_1(t) = -1.035454x10^{-2} sign\big(h_1(t) - h_2(t)\big)\sqrt{|h_1(t) - h_2(t)|} + 1.143x10^{-3}u_1(t) \tag{6.9}$$

$$\dot{h}_2(t) = 1.035454x10^{-2} sign\big(h_1(t) - h_2(t)\big)\sqrt{|h_1(t) - h_2(t)|} - 1.035454x10^{-2}\sqrt{h_2(t)} \tag{6.10}$$

The transition of the mathematical model into simulink is in Figure 6.3.

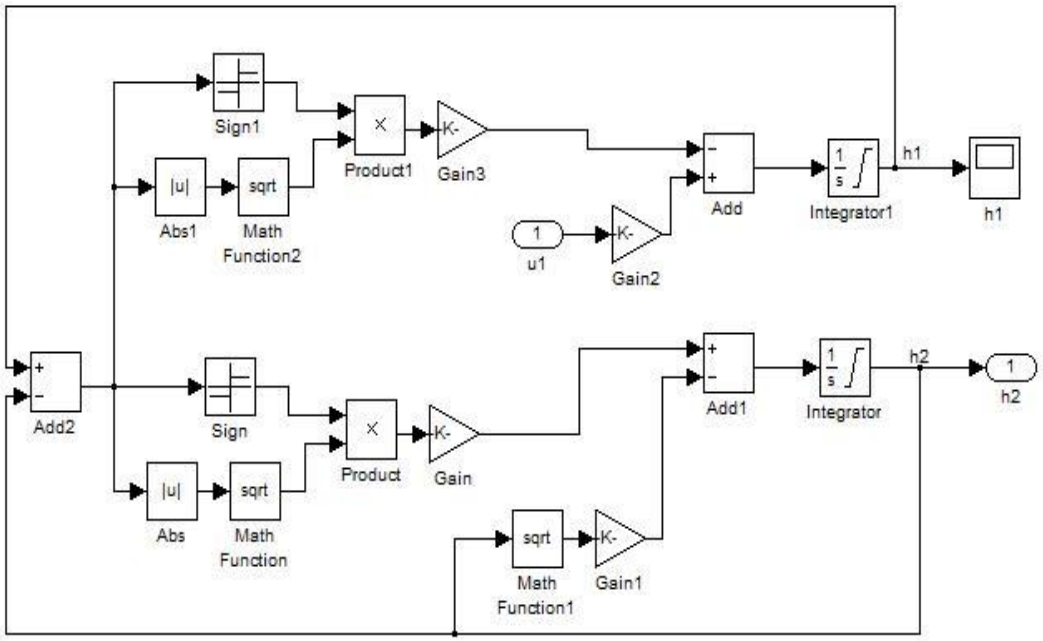

Figure 6.3 : Simulink presentation of mathematical model.

6.2 Implemeting Particle Swarm Tuning Methodology

The derived nonlinear model of coupled tank sytem has been used as a benchmark model for implementing fuzzy PID control with particle swarm optimization to tune the crisp values of Takagi-Sugeno values and PID coefficients for getting robust nonlinear control. In this study, the same particle swarm optimization technique that has been used so far was used to find the crisp values and coefficient of the fuzzy PID controller. Due to the nonlinearity of the process, set value has been limited between 0 and 0.3 and divided into 3 regions. The purpose of limiting the control trajectory is due to the limitation that the heights of the tanks are between 0 and 0.6m in the mathematical model. PSO optimization technique has been implemented in three regions separately with respect to desired input as shown below in order to capture all the transition values. The different regions are depicted in Figure 6.4.

I. h_2: 0.00 - 0.15 (first region)

II. h_2: 0.15 - 0.20 (second region)

III. h_2: 0.20 - 0.30 (third region)

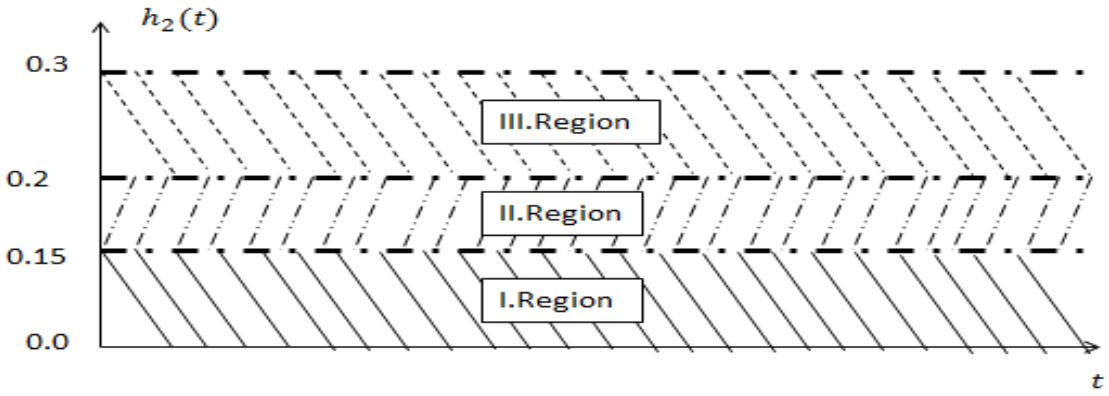

Figure 6.4 : Different control regions for transitions.

When we run the PSO algorithm for different regions with different inputs between specific intervals, all the crisp values and fuzzy PID parameters are computed offline to control the system in specific regions. It can enable us to switch smoothly between different controls trajectories. The regions have been evaluated with the control signals depicted Figure 6.5, 6.6 and 6.7 and the ensured crisp values are shown in Table 6.2 for different regions [16,17].

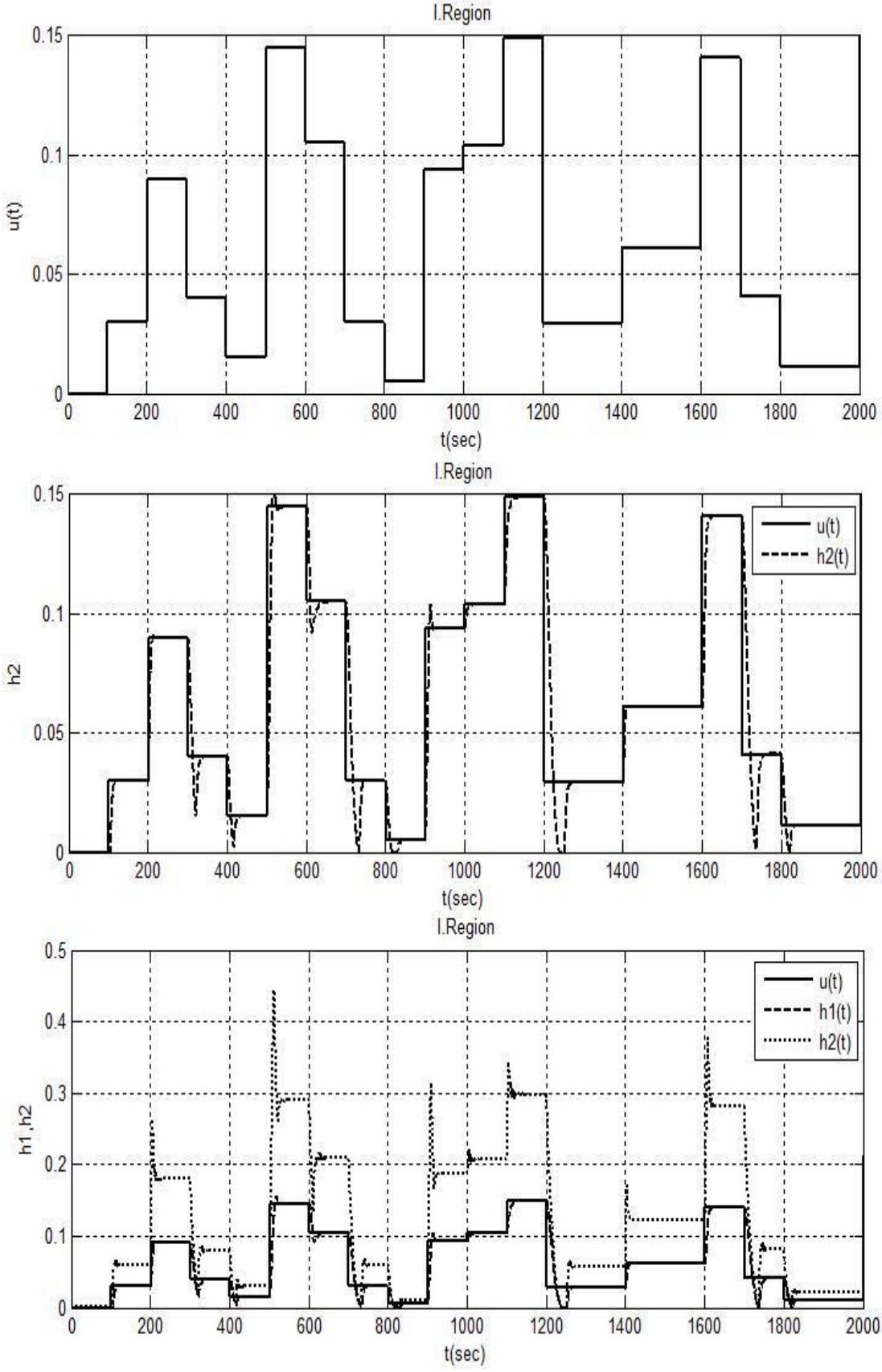

Figure 6.5 : 0.00-0.15 u(t) and h1(t), h2(t) for first region.

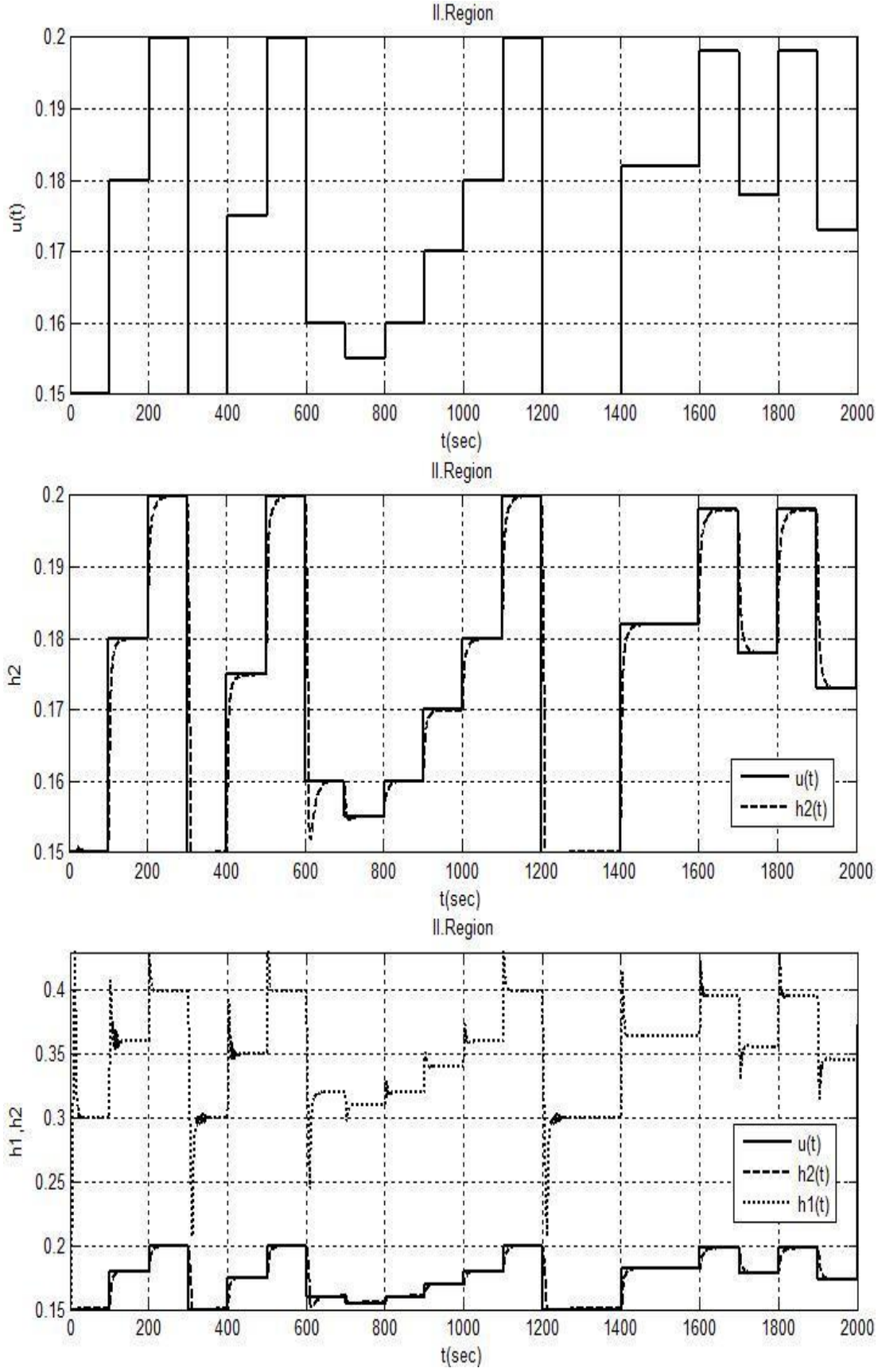

Figure 6.6 : 0.15-0.20 u(t) and h1(t), h2(t) for second region.

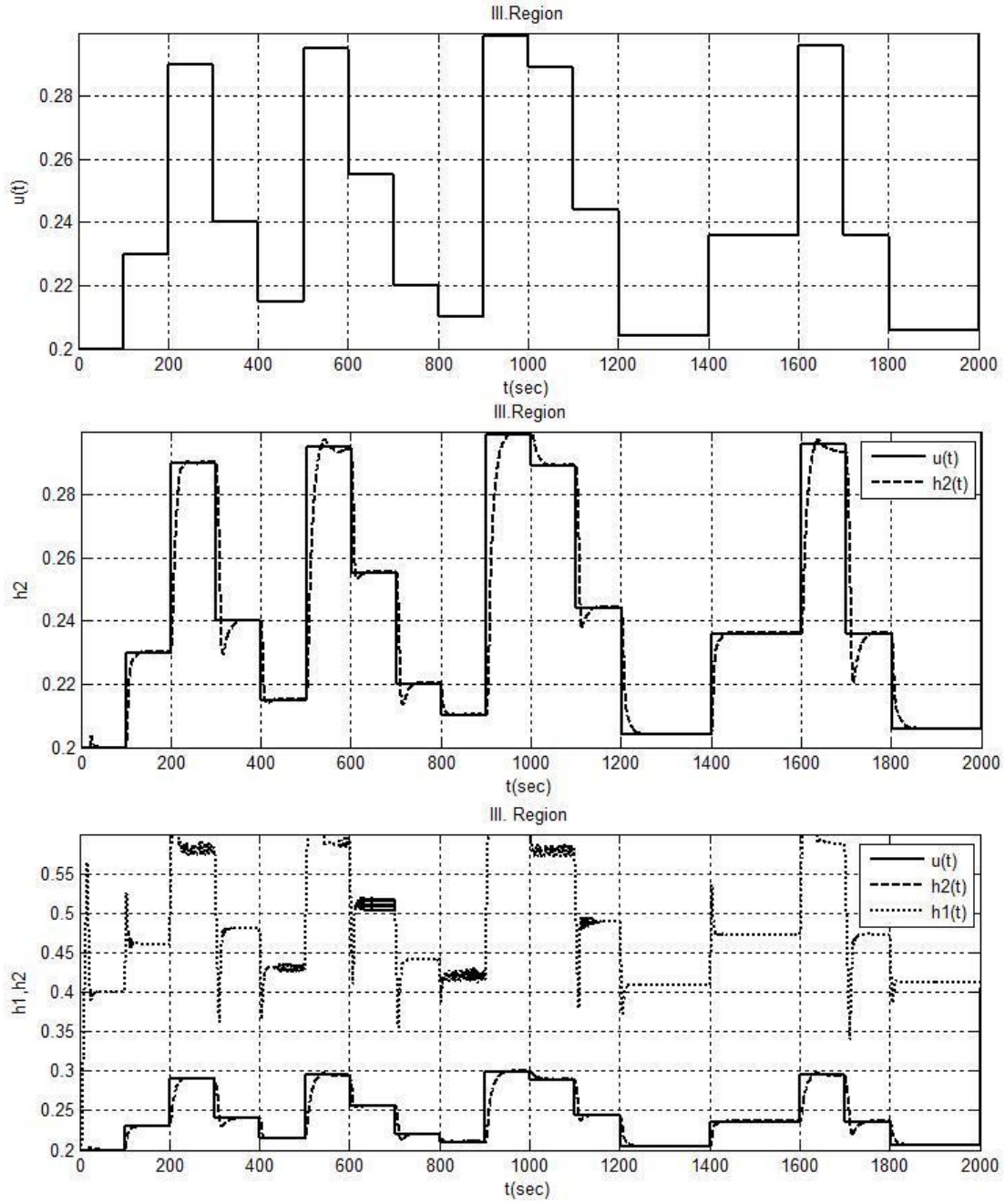

Figure 6.7 : 0.20-0.30 u(t) and h1(t), h2(t) for third region.

Table 6.2 : Takagi-Sugeno crisp values on nonlinear-coupled tank system.

Crisp Values for Fuzzy IPD	k1	k2	k3	k4	k5	k6	k7	k8	k9	fxmin
0.00-0.15 (I. Region)	-19,4	13,07	8,51	21,80	0,00	-8,43	-15,6	-26,7	-32,6	8,87
0.15-0.20 (II. Region)	44,66	-10,5	24,48	-10,4	0,00	11,53	5,83	9,35	-11,9	16,71
0.20-0.30 (III. Region)	-0,53	23,64	-0,57	27,73	0,00	-31,8	27,82	-11,3	11,01	24,07

It is well known that the mathematical model of a coupled tank system is nonlinear. This nonlinearity characteristic feature of the tank model makes it impossible to derive only one set of PID parameters between regions that would be valid for all operating points. In other words, optimal controller parameter values obtained for a particular operating point will not be optimal for another point. Therefore, optimal PID controller parameters are figured out by implementing least square fitting for some predetermined height conditions of the coupled tank. For this purpose, a set of functions for various operating conditions are calculated using least square fitting method for the controller parameters depending on the ensured constant Takagi-Sugeno values. The optimum PID parameters for specific regions can be calculated as below:

For 1st region;

$$K_p = (-7.1907\,u^3 + 4.7879\,u^2 - 0.5777\,u - 0.0547)x\ 10^3$$

$$K_i = (\ 1.2661\,u^3 - 0.3805\,u^2 + 0.0371\,u - 0.0016)x\ 10^4 \qquad \textbf{(6.11)}$$

$$K_d = (-753.8554\,u^3 + 266.7015\,u^2 + 268.3418\,u + 6.3814)$$

For 2nd region;

$$K_p = (-4.6655\,u^3 + 2.4571\,u^2 - 0.4064\,u + 0.0251)x\ 10^4$$

$$K_i = (-2.7268\,u^3 - 1.4185\,u^2 - 0.2449\,u - 0.0141)x\ 10^5 \qquad \textbf{(6.12)}$$

$$K_d = (-4.3872\,u^3 + 2.2802\,u^2 - 0.3932\,u + 0.0226)\ x\ 10^6$$

For 3rd region;

$$K_p = (5.1245\,u^3 - 3.3422\,u^2 + 0.8109\,u - 0.0493)x\ 10^3$$

$$K_i = (3.6542\,u^3 - 2.7966\,u^2 + 0.7105\,u - 0.0607)x\ 10^4 \qquad \textbf{(6.13)}$$

$$K_d = (-6.0174\,u^3 + 4.1898\,u^2 - 0.9584\,u + 0.0710)\ x\ 10^5$$

where, u is the set values of the coupled tanks sytems. Complete structure of the simulated coupled tanks system can be seen in Figure 6.8.

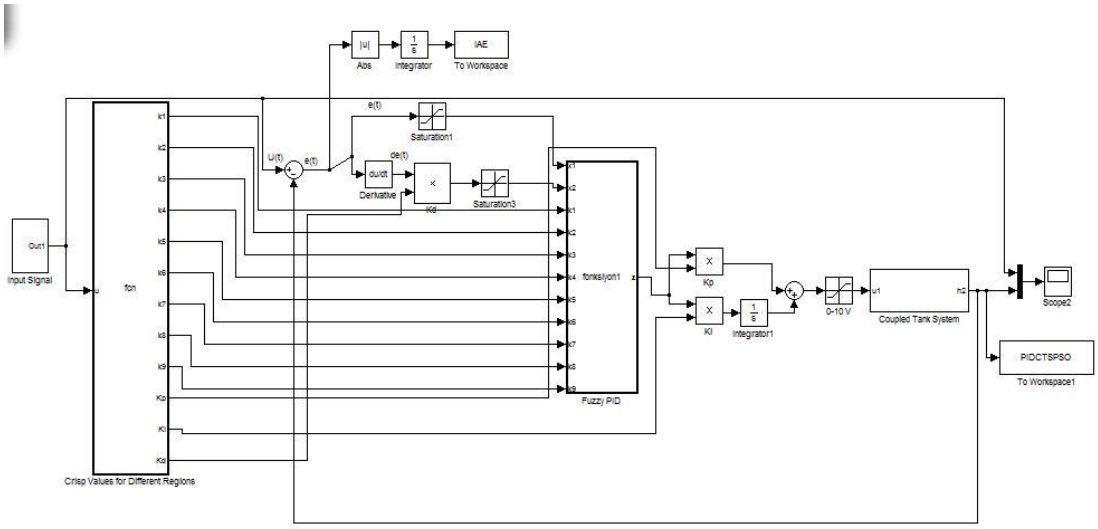

Figure 6.8 : Simulink representation of coupled tank system.

6.3 Simulation Results

Coupled tank system with fuzzy PID was simulated using MATLAB/SIMULINK. The parameters of fuzzy PID are given in the figures for each input. The control parameters were calculated online during the simulations and simulation results are shown on the following pages.

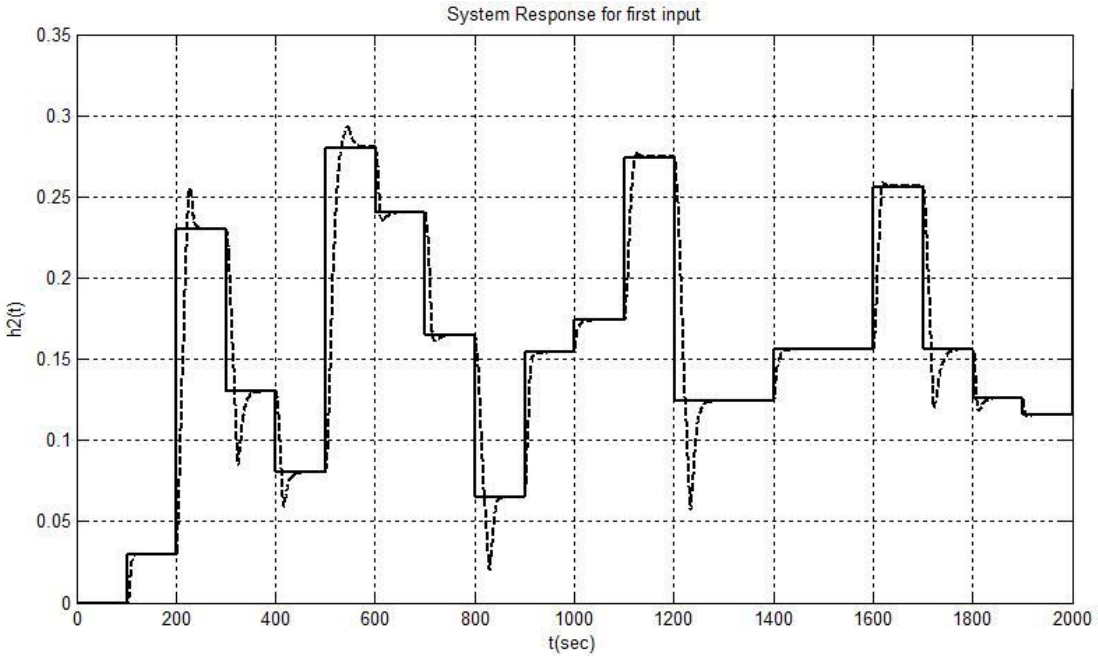

Figure 6.9 : System response for first input signal.

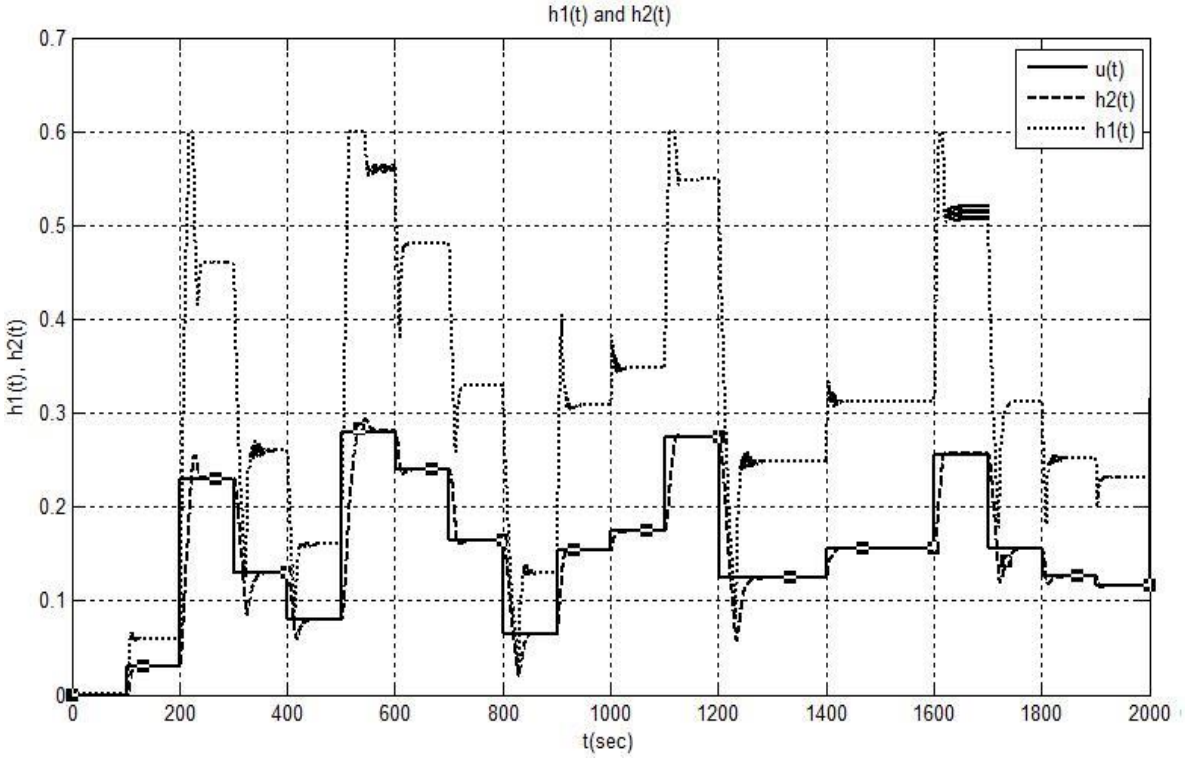

Figure 6.10 : Changes on $h_1(t)$ and $h_2(t)$.

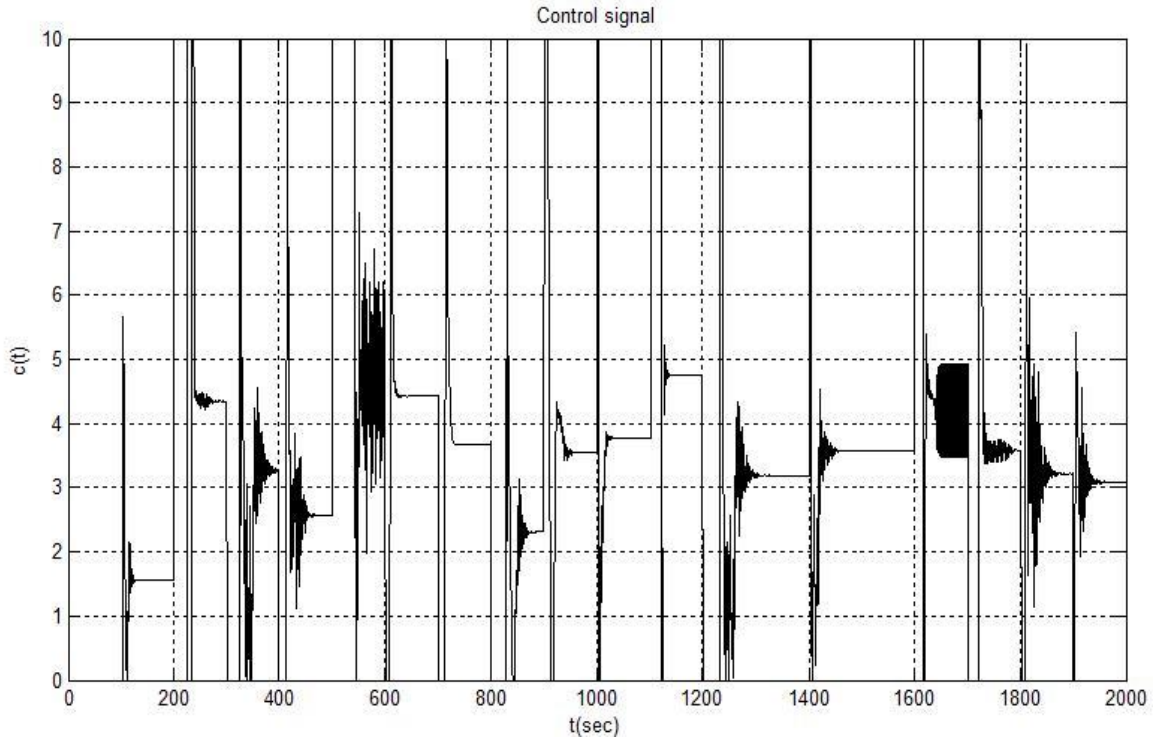

Figure 6.11 : Control signal c(t).

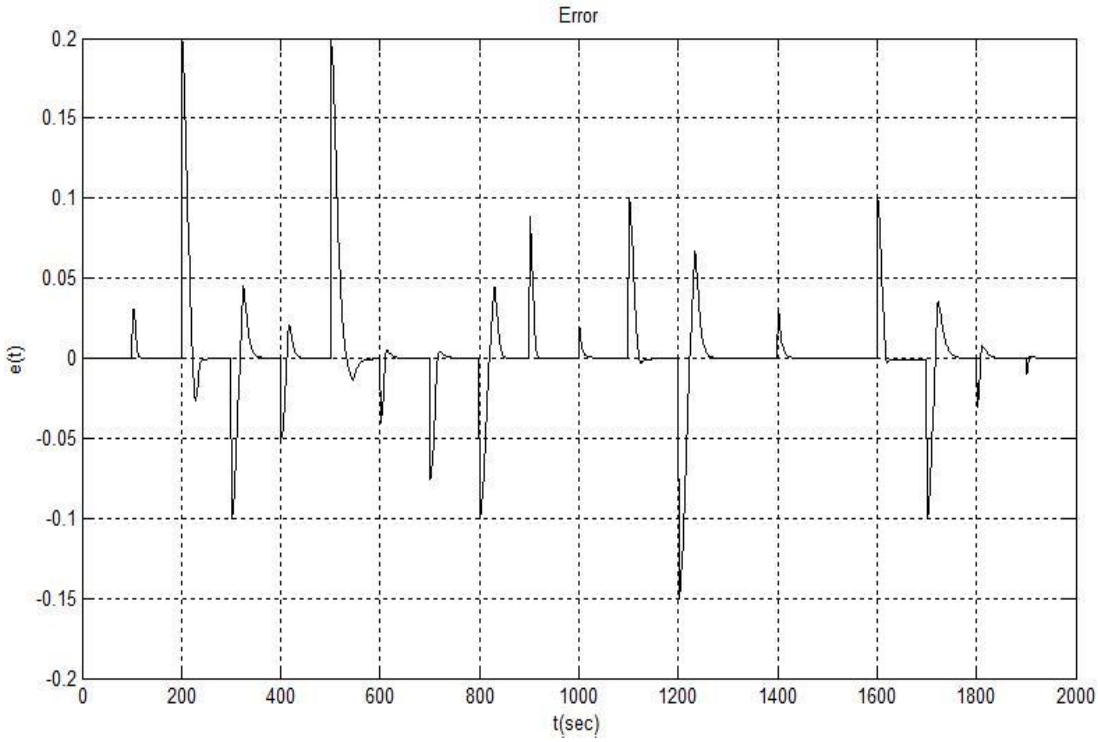

Figure 6.12 : Error e(t).

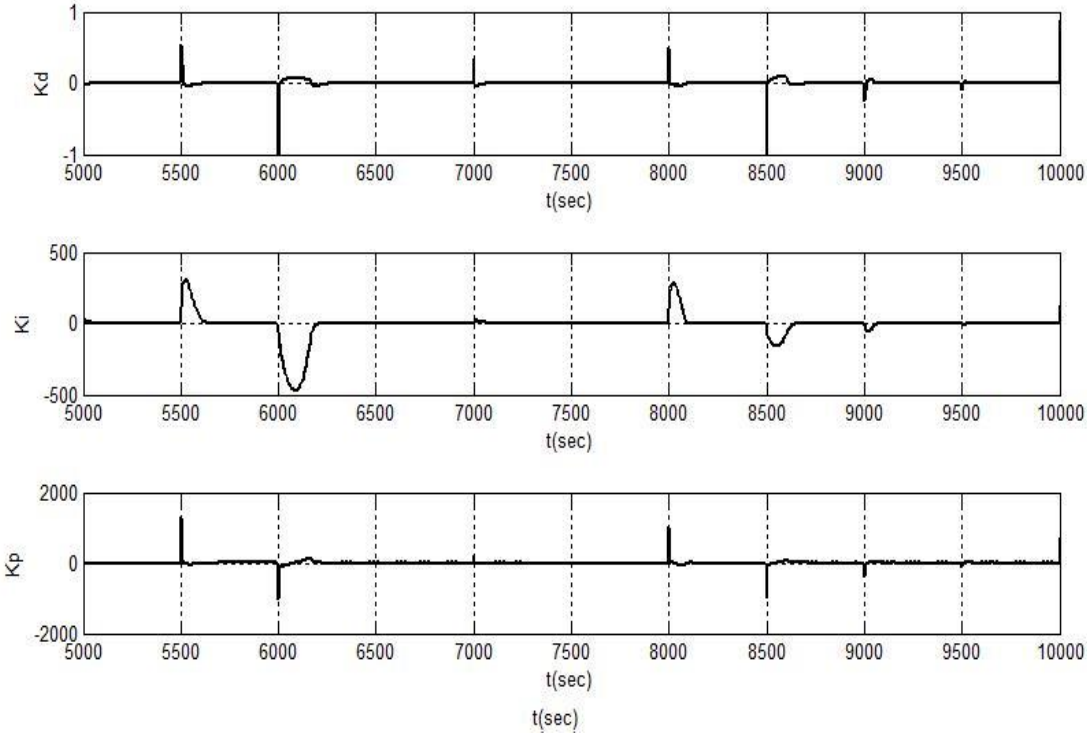

Figure 6.13 : Changes on K_p, K_i and K_d.

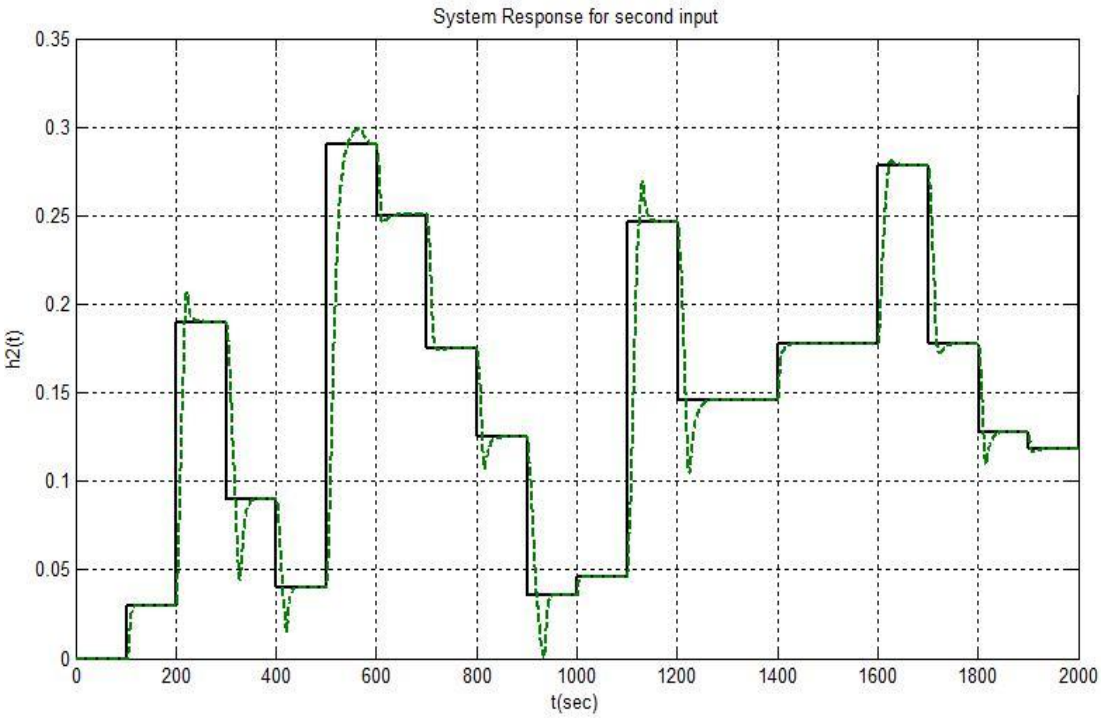

Figure 6.14 : System response for second input signal.

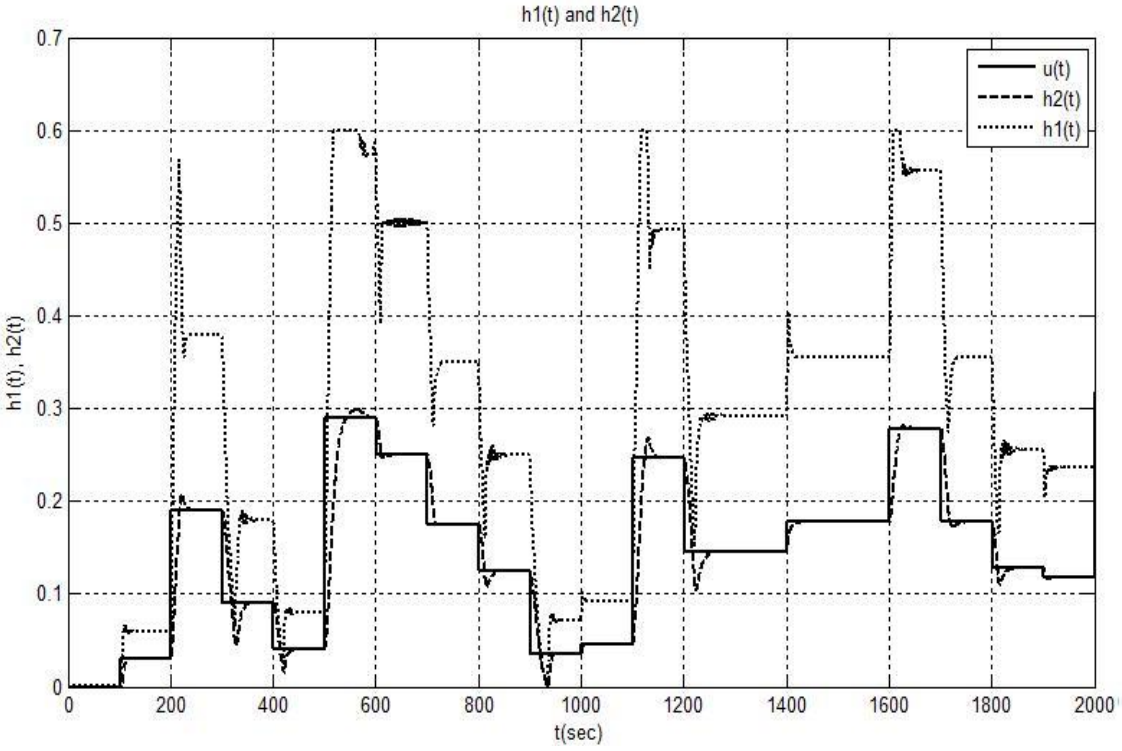

Figure 6.15 : Changes on $h_1(t)$ and $h_2(t)$.

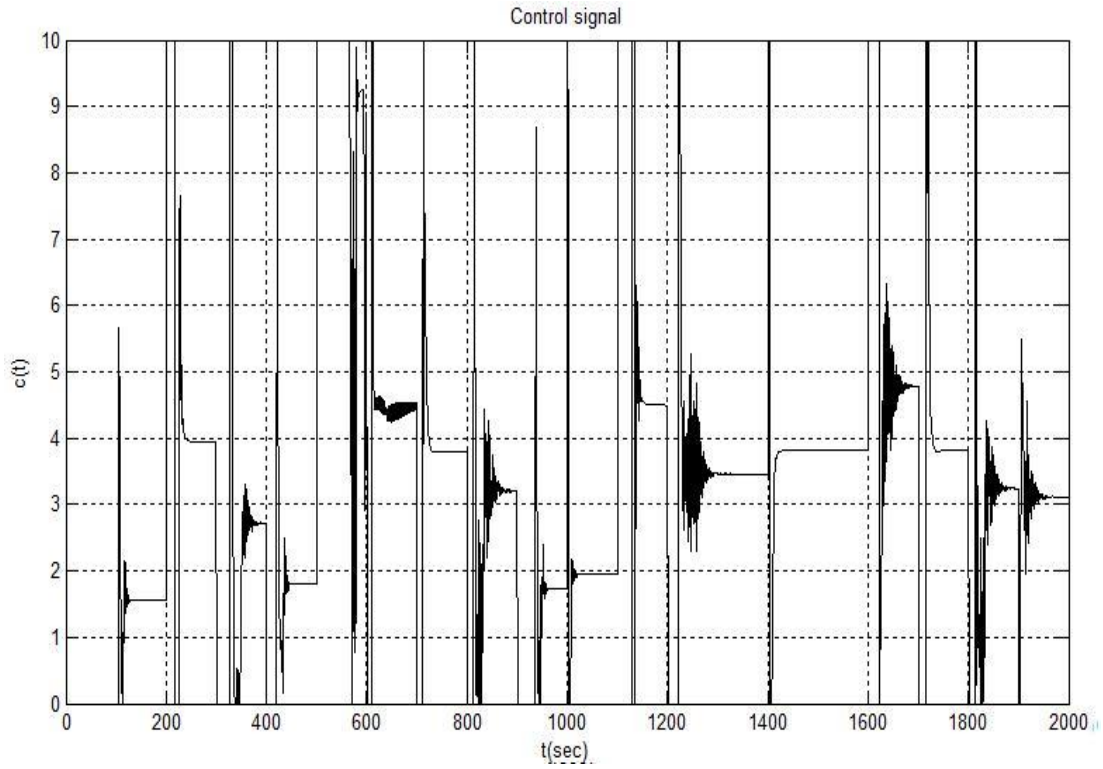

Figure 6.16 : Control signal c(t).

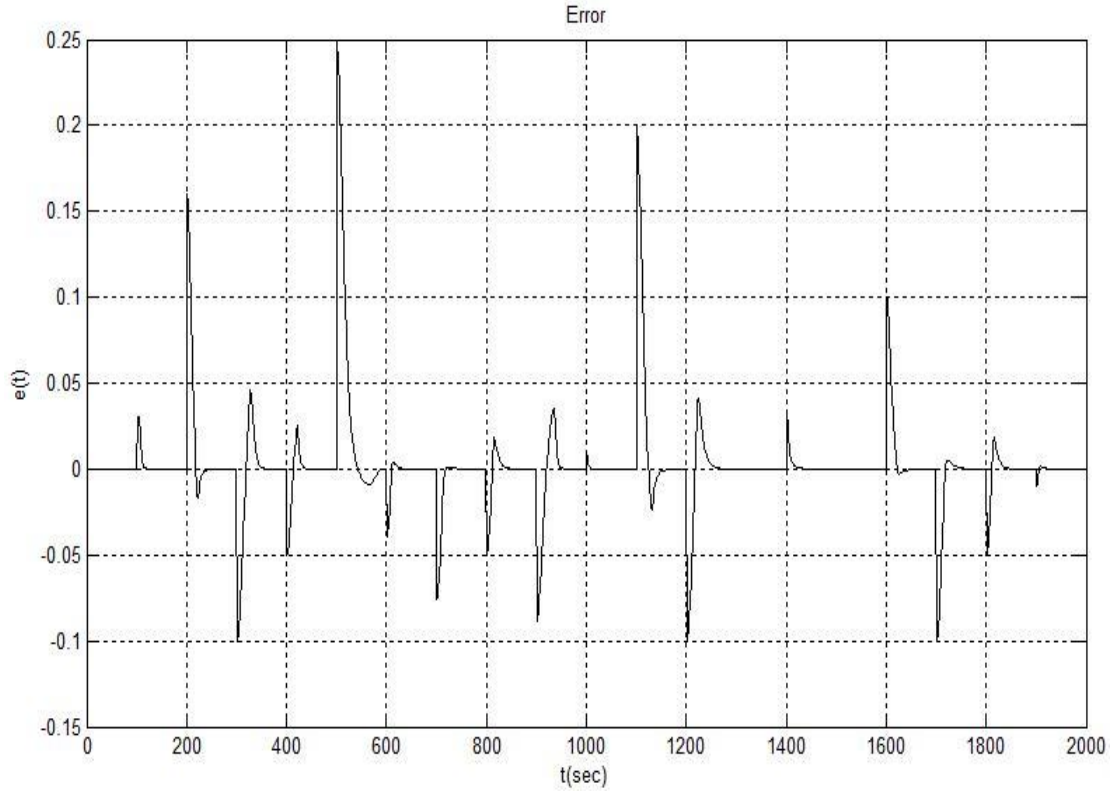

Figure 6.17 : Error e(t).

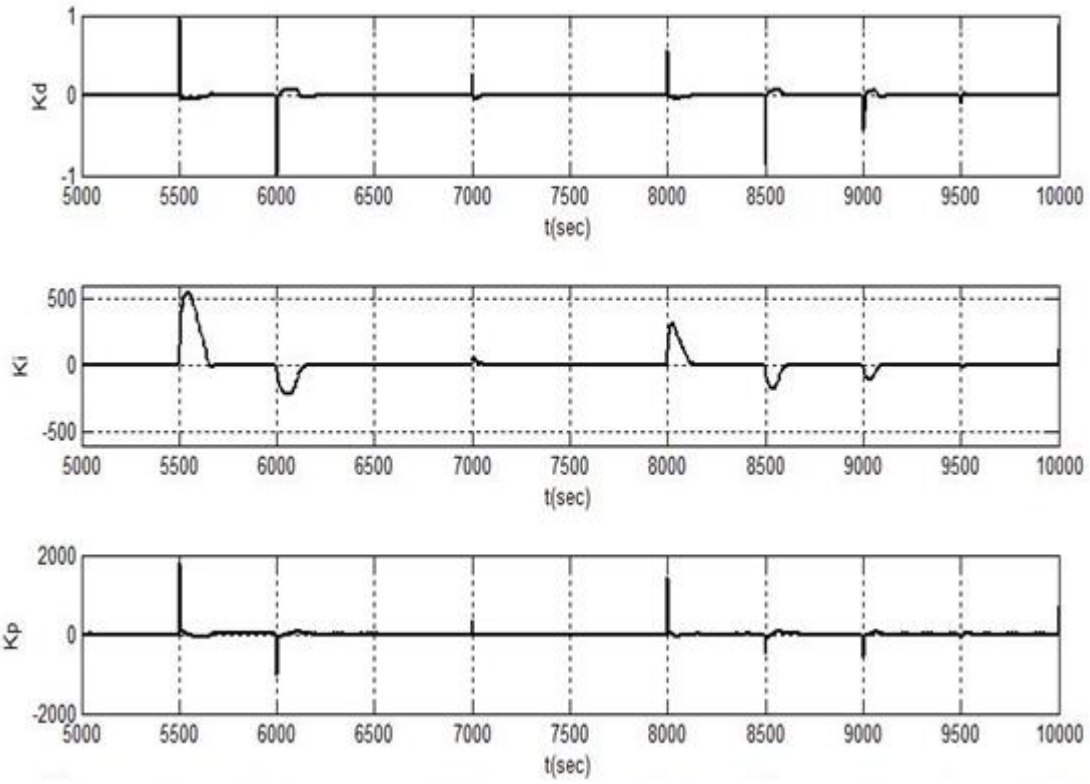

Figure 6.18 : Changes on K_p, K_i and K_d.

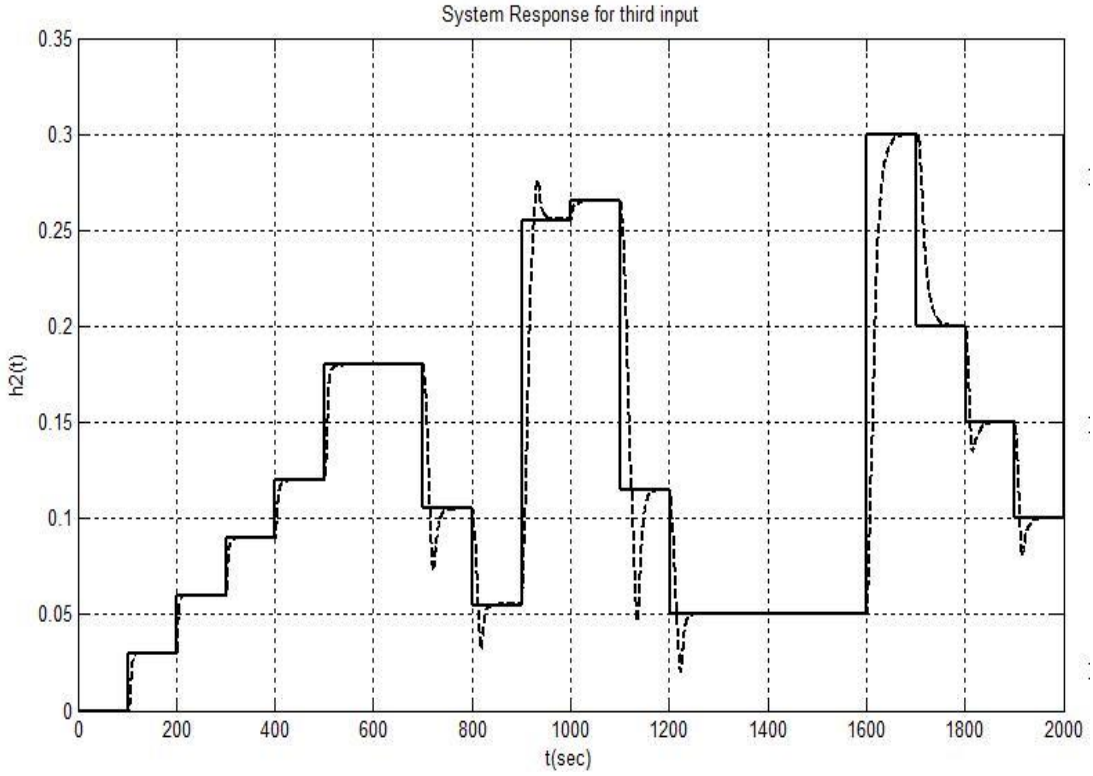

Figure 6.19 : System response for third input.

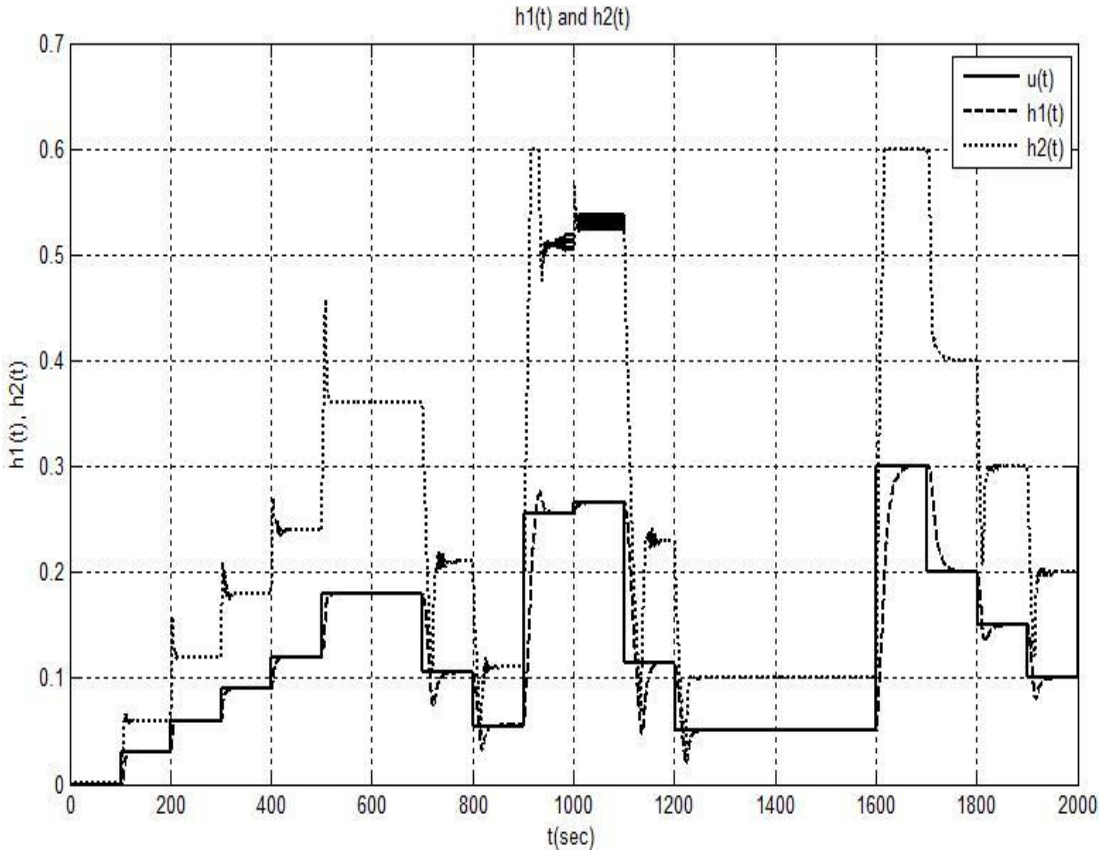

Figure 6.20 : Changes on $h_1(t)$ and $h_2(t)$.

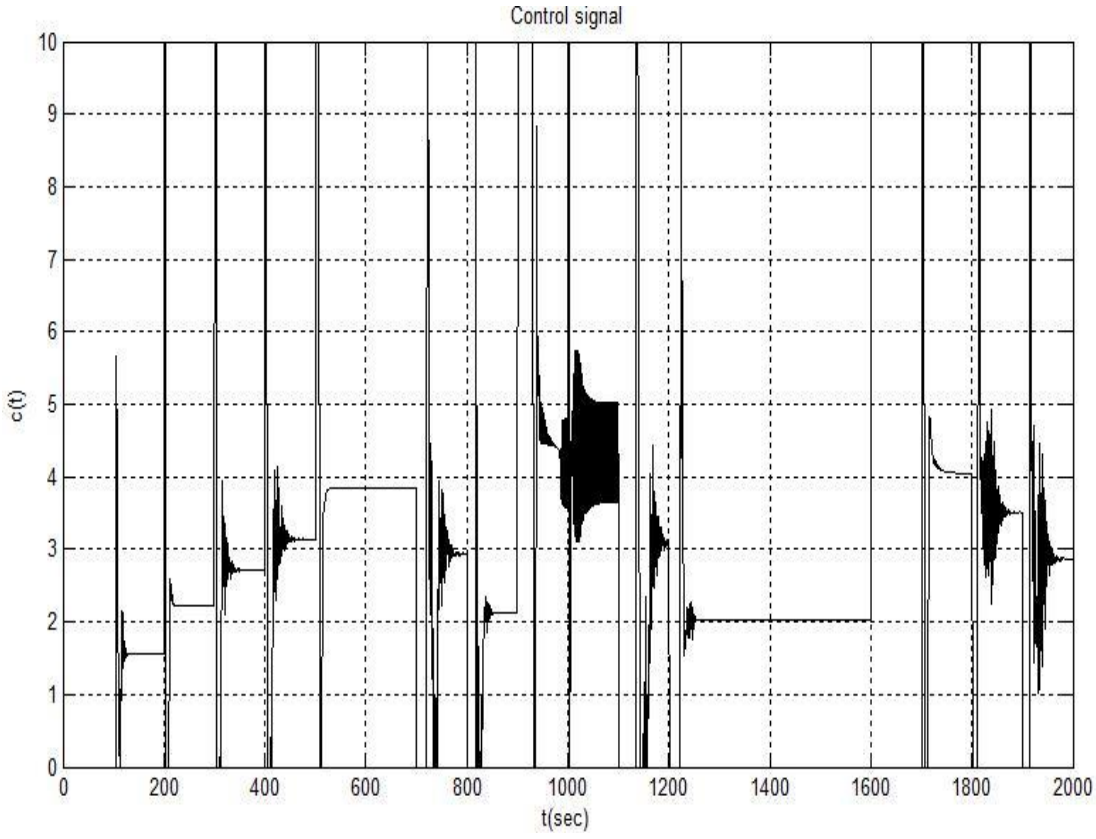

Figure 6.21 : Contol signal c(t).

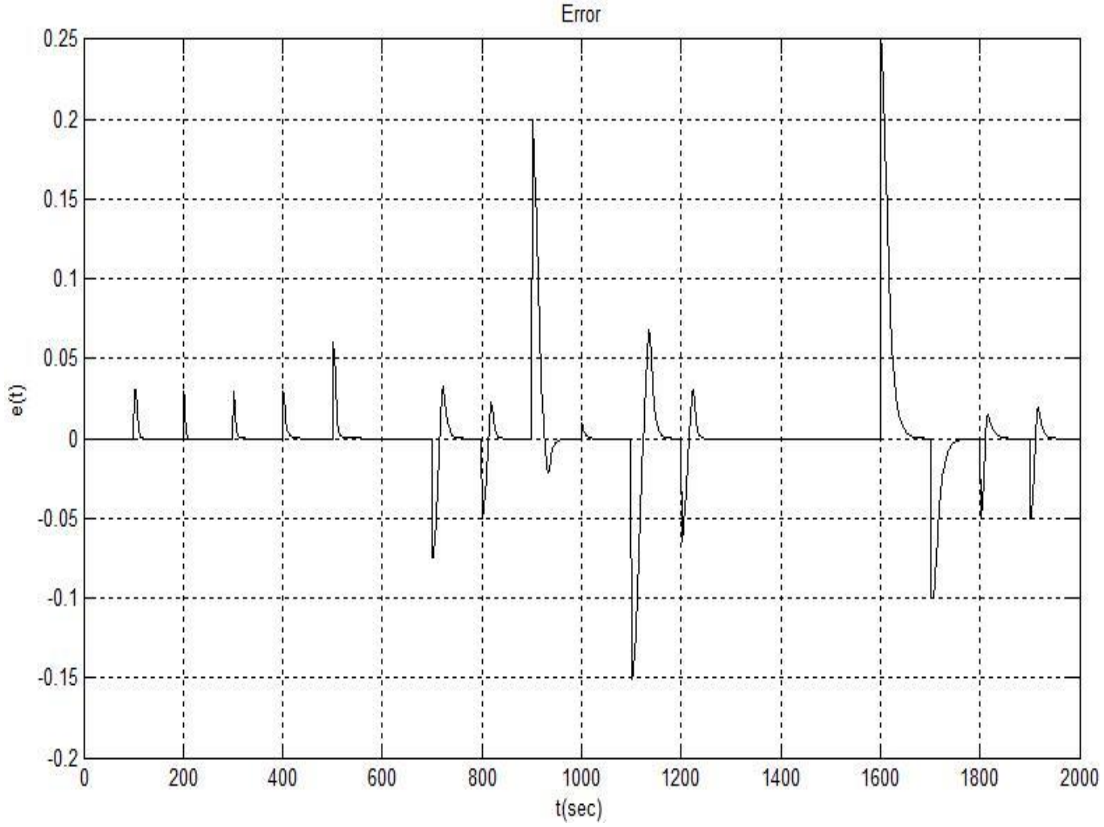

Figure 6.22 : Error e(t).

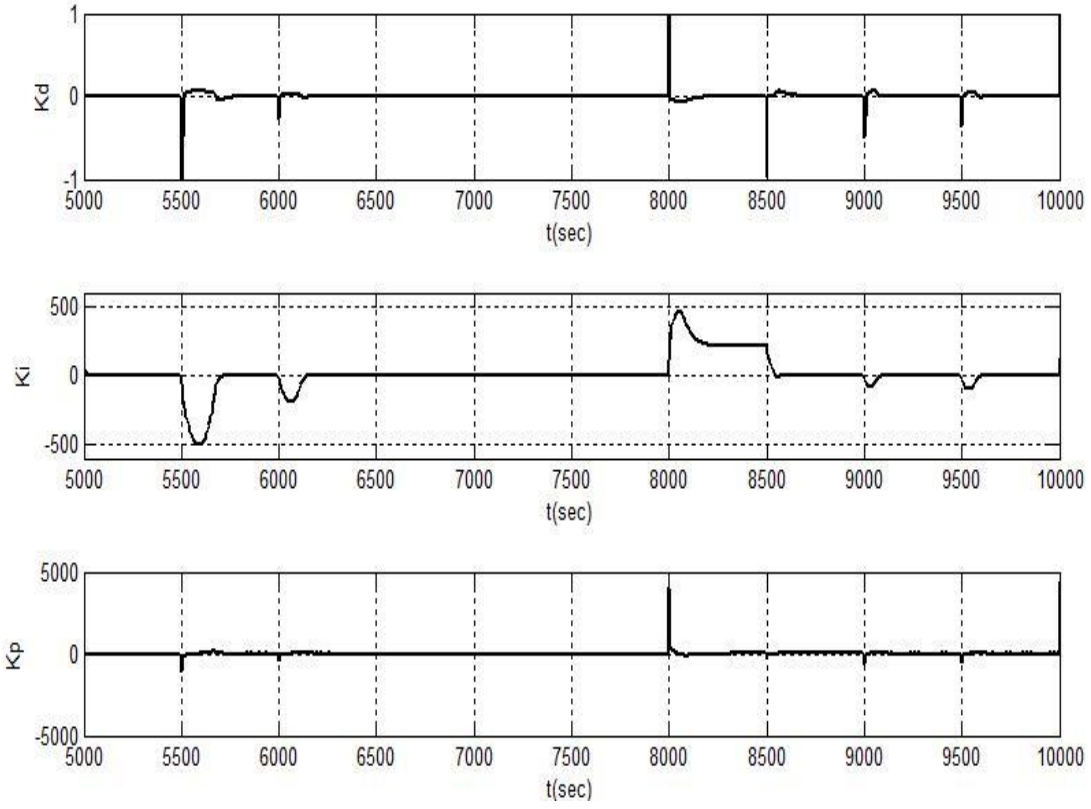

Figure 6.23 : Changes on K_p, K_i and K_d.

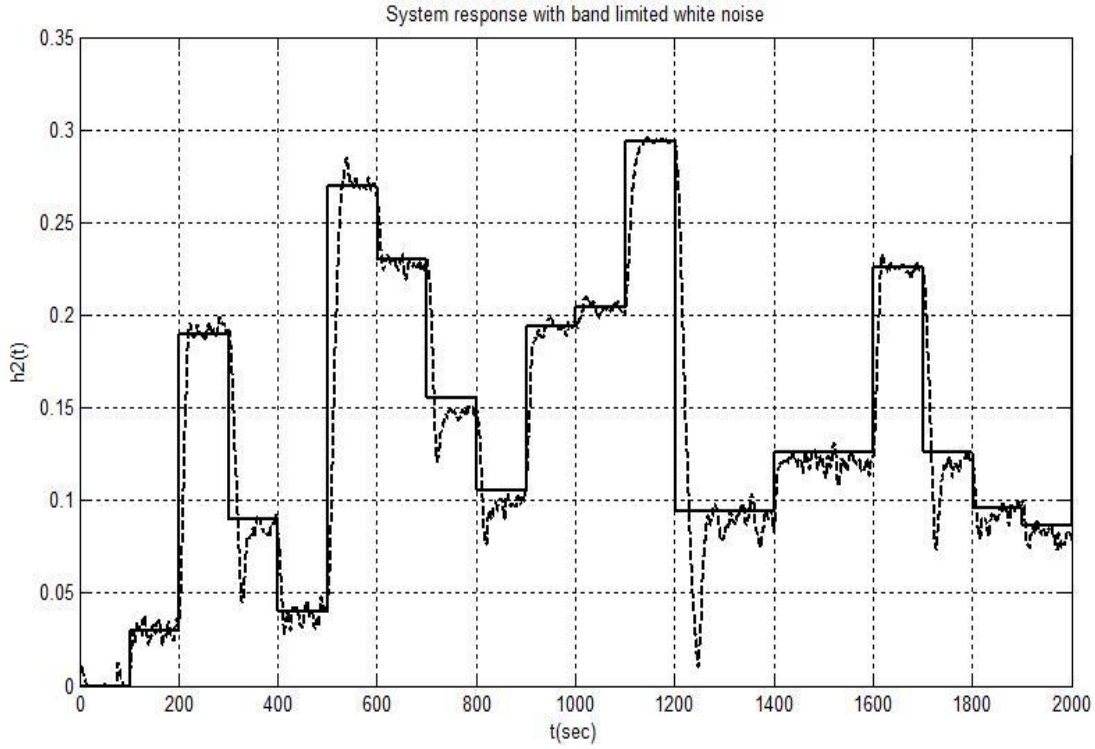

Figure 6.24 : System response with band limited white noise on sensor.

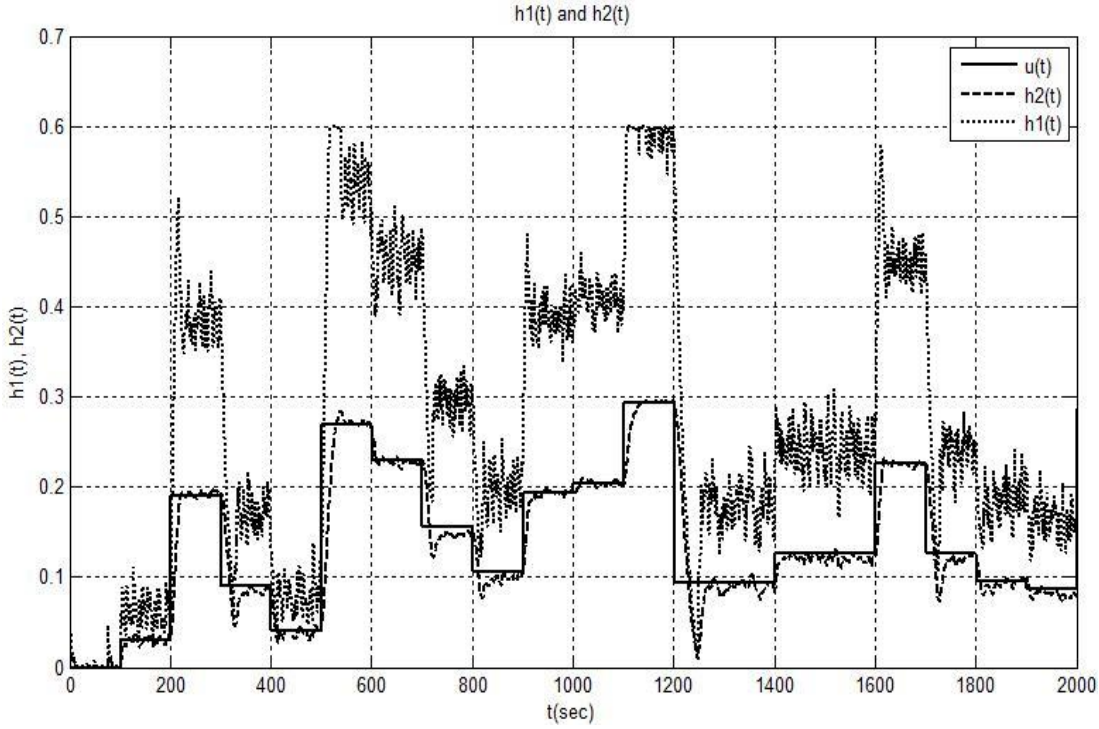

Figure 6.25 : Changes on $h_1(t)$ and $h_2(t)$ considering white noise.

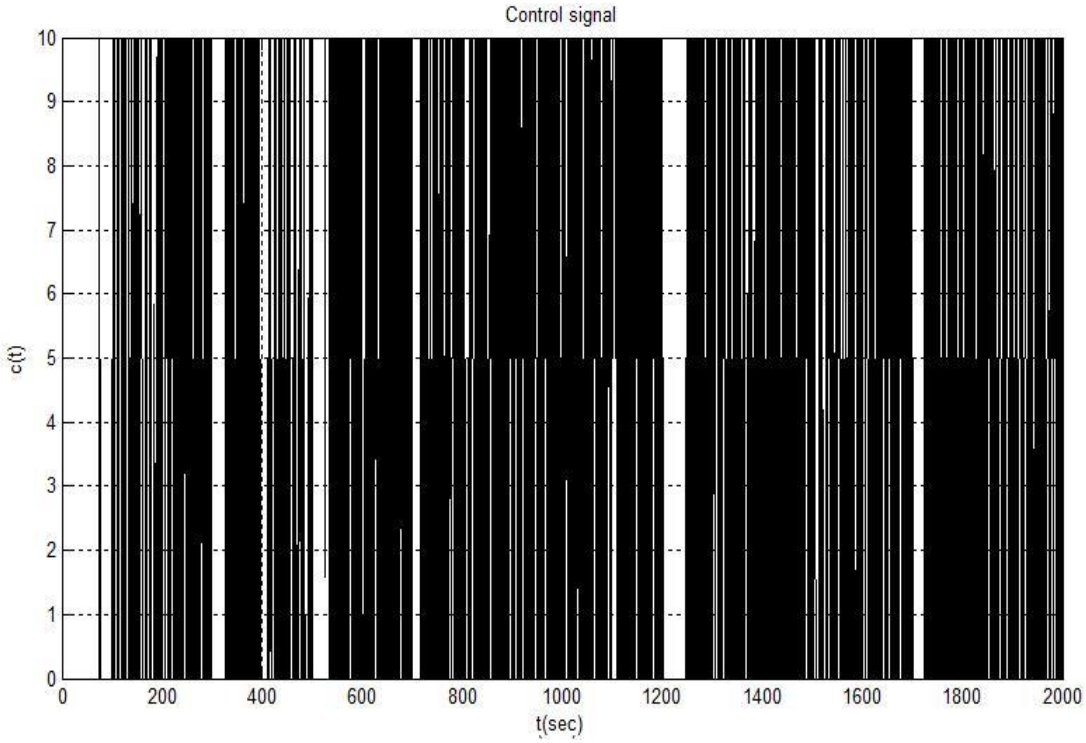

Figure 6.26 : Control Signal with band limited white noise c(t).

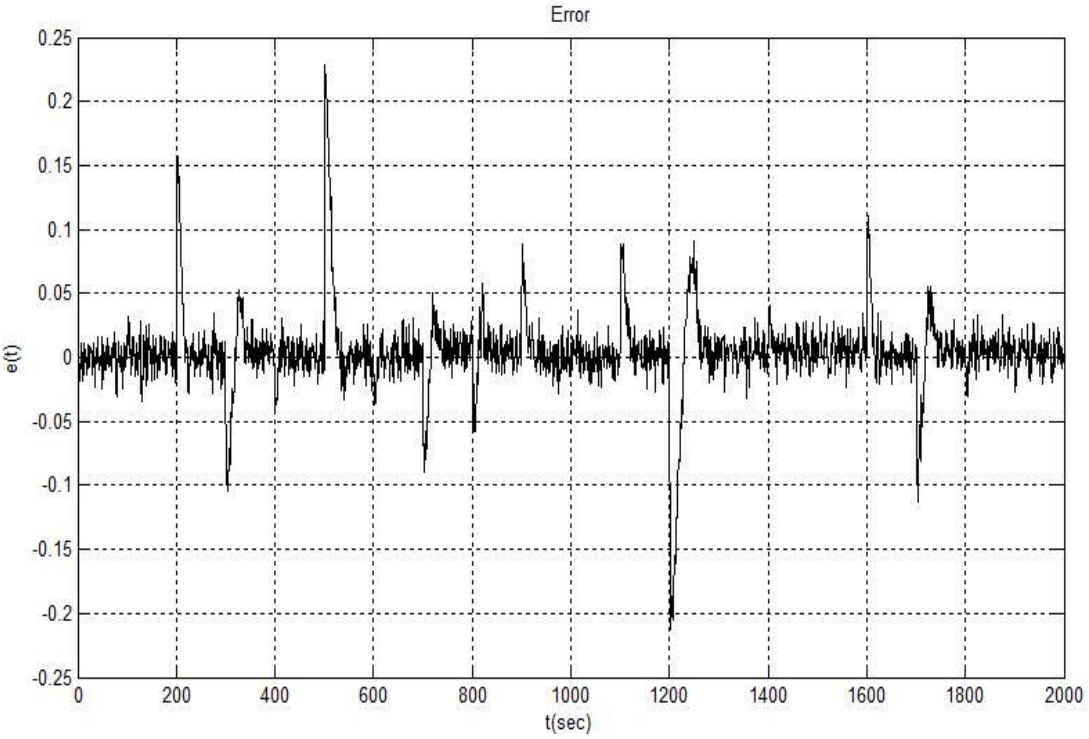

Figure 6.27 : Error with band limited white noise e(t).

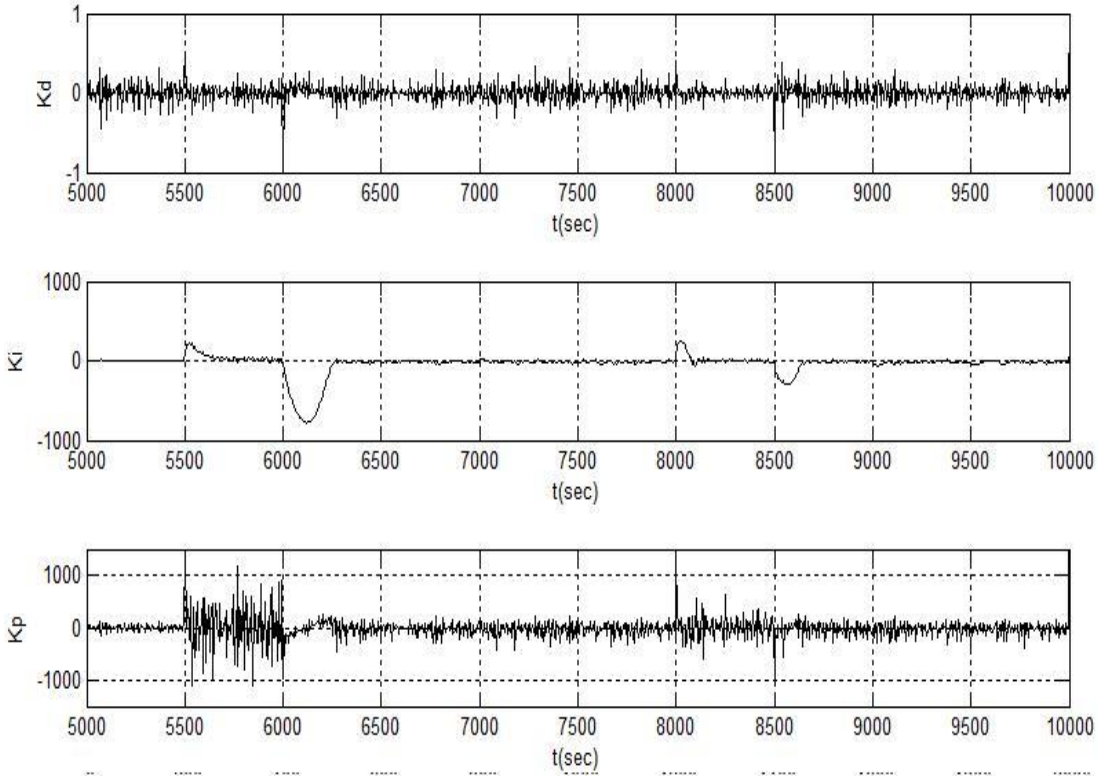

Figure 6.28 : Changes on K_d, K_i, and K_p with white noise.

It needs to be noticed that above simulations have been derived with the assumption that there are no any faults in the system. The second part of the simulations are performed assuming there are faults in the system [21].

The fault classes under consideration are defined as follows;

- Pump actuator fault; the equation given in **(6.7)** is the true entrance rate which includes $K_f \in [0,1]$. It is considered that if $K_f = 0.2$ then the pump has 20% actuator fault and if $K_f = 0.5$ then the pump has 50% fault in the actuator. When the simulations are performed with this type of fault in the pump, the below result are obtained.

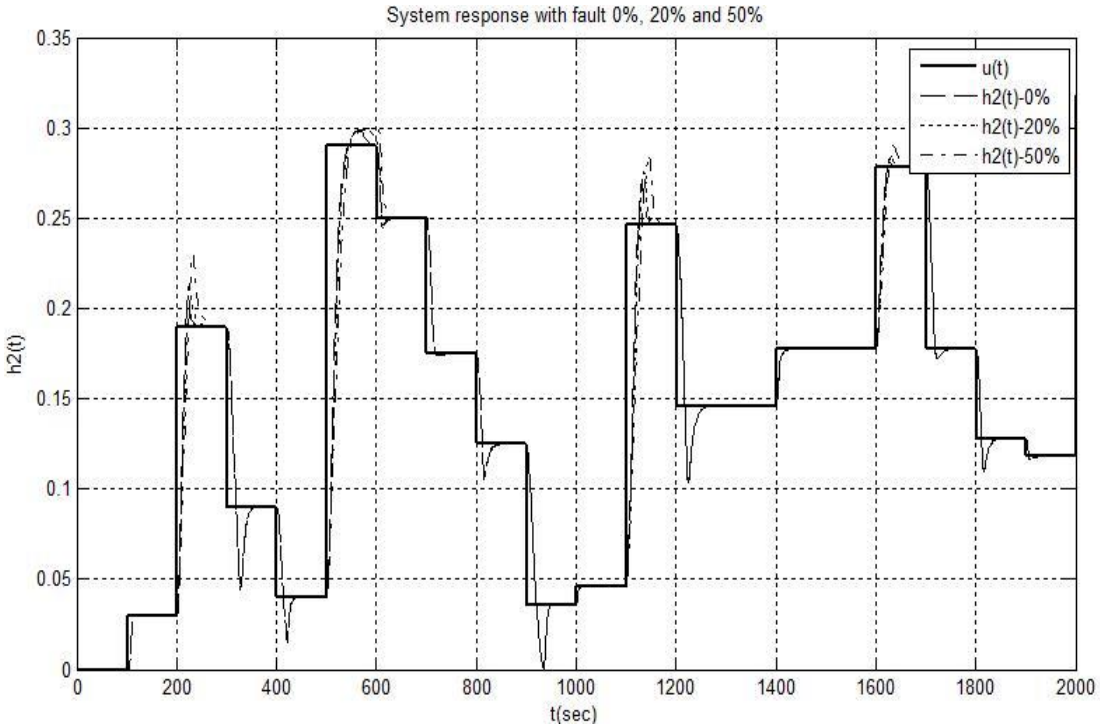

Figure 6.29 : System response with 0%, 20% and 50% fault on the actuator.

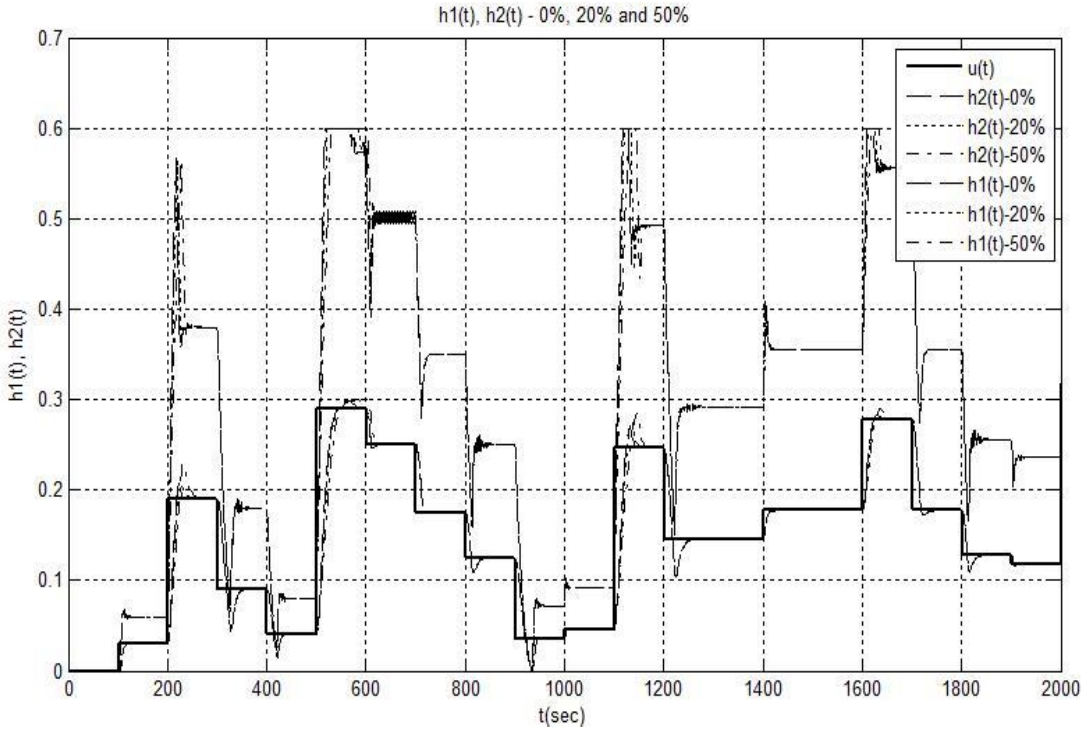

Figure 6.30 : Changes on heights with 0%, 20% and 50% fault on the actuator.

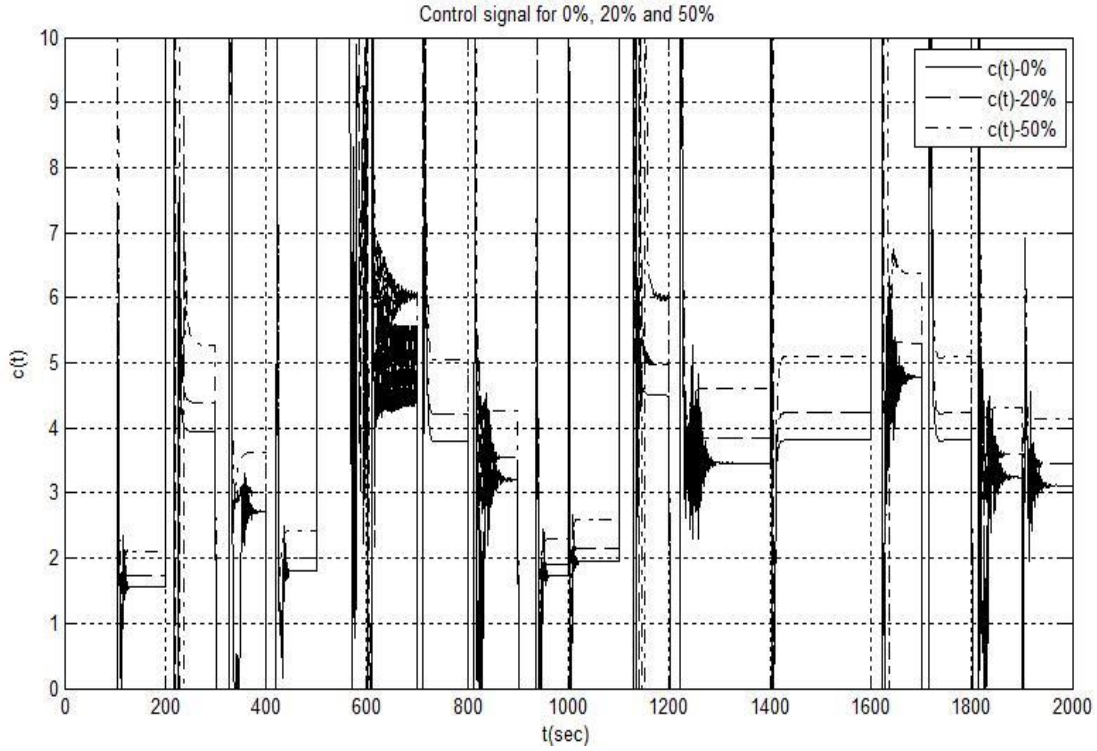

Figure 6.31 : Control signal with 0%, 20% and 50% fault on actuator.

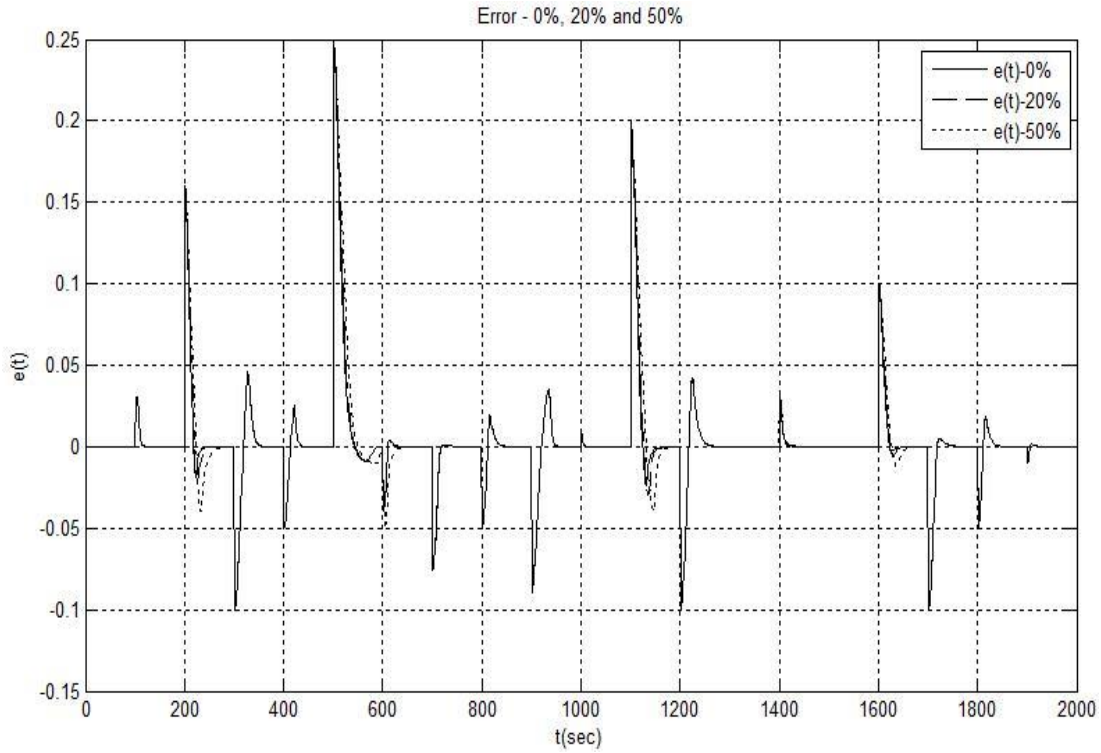

Figure 6.32 : Error with 0%, 20% and 50% fault on actuator.

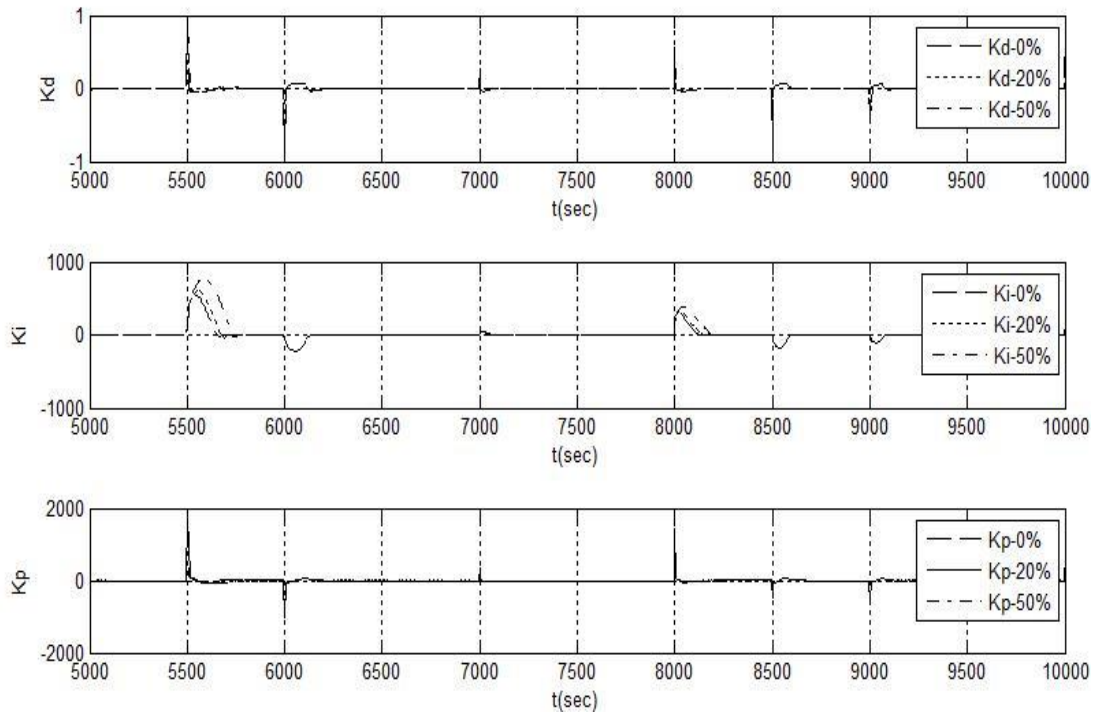

Figure 6.33 : Changes on K_d, K_i and K_p with 0%, 20%, 50% fault on actuator.

- Leakage in Tank 1; We assume that the leak is circular in shape and of known radius r_1 . The outflow rate of the leak in tank 1 is given by:

$$q_{f1}(t) = a_1 \pi {r_1}^2 \sqrt{2gh_1(t)} \quad \textbf{(6.14)}$$

- Leakage in Tank 2; We assume that the leak is circular in shape and of known radius r_2 . The outflow rate of the leak in tank 2 is given by:

$$q_{f2}(t) = a_2 \pi {r_2}^2 \sqrt{2gh_2(t)} \quad \textbf{(6.15)}$$

The following simulation results have been obtained assuming there is a %20 fault on pump actuate and leakages on both tanks 1 and 2 of 7 mm radius. As it can be seen, fuzzy PID with particle swarm optimization has good performance against the system faults in terms of overshoot, rise time etc..

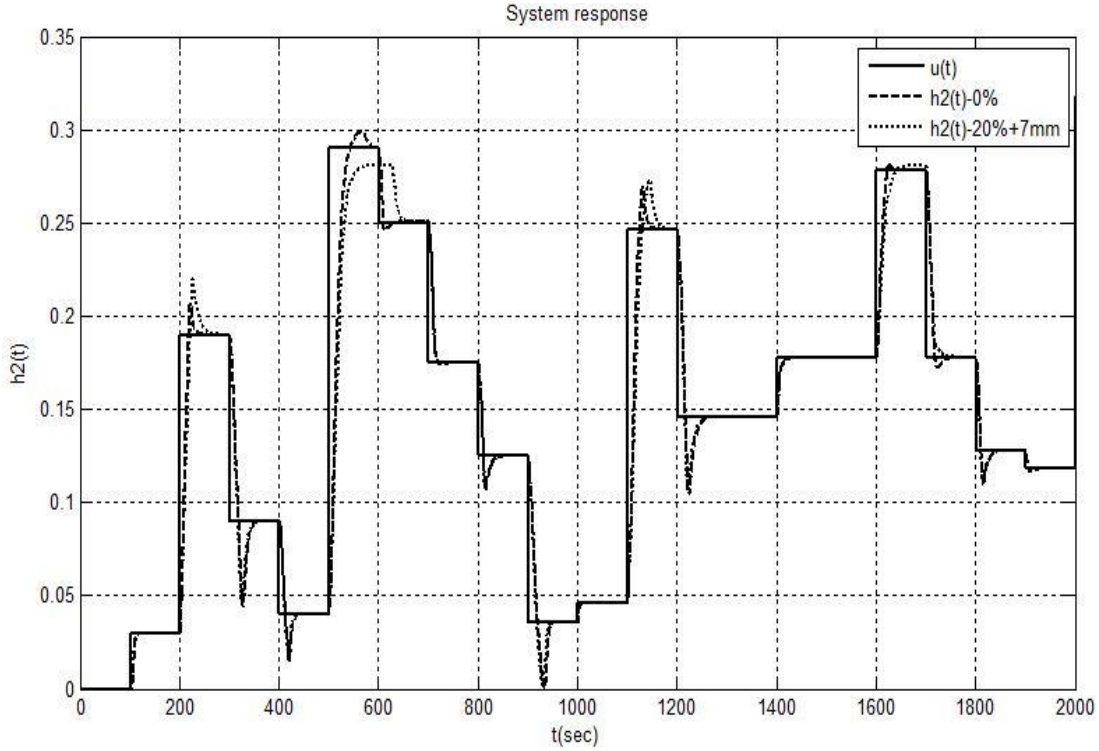

Figure 6.34 : System response, 0% fault, 20% actuator fault, 7mm dia. holes.

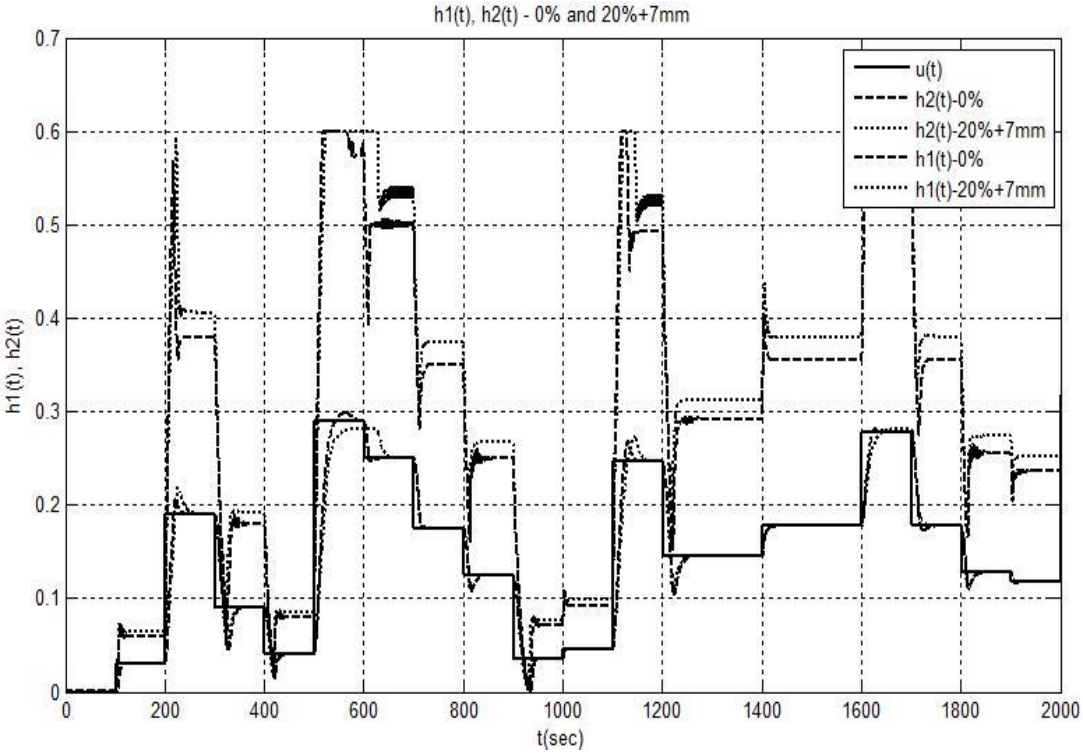

Figure 6.35 : Change on heights, 0% fault, 20% actuator fault, 7mm dia. holes.

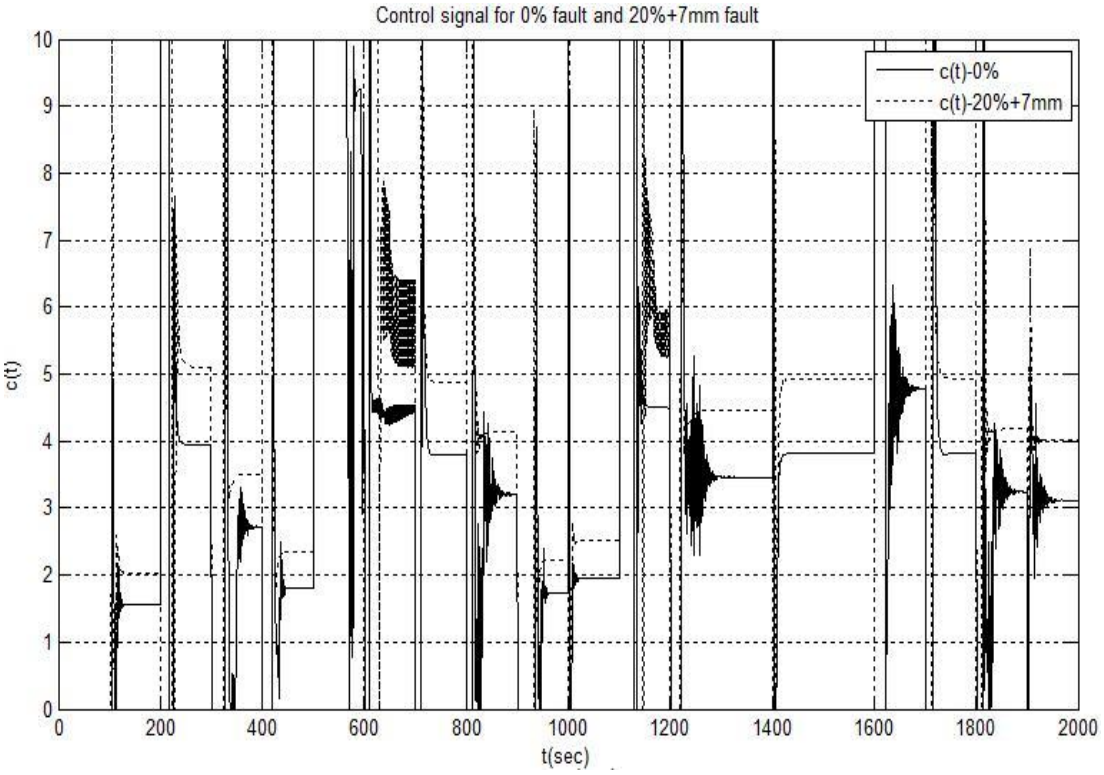

Figure 6.36 : Control signals, 0% fault and 20% actuator fault, 7mm dia. holes.

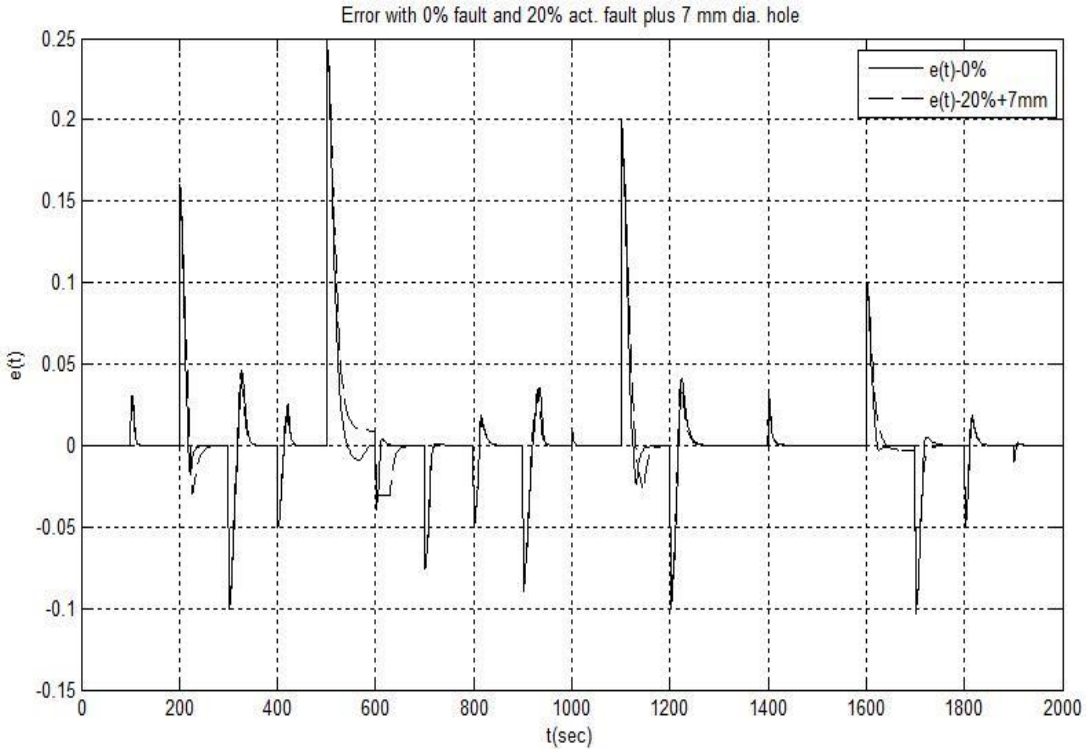

Figure 6.37 : Errors, 0% fault,20% actuator fault, 7mm dia. holes.

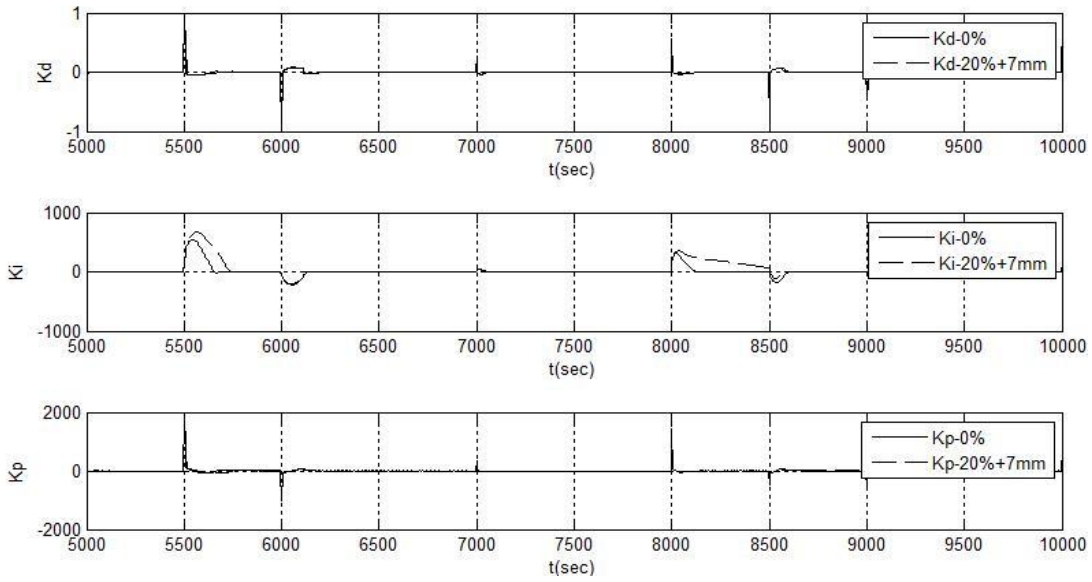

Figure 6.38 : K_d, K_i, K_p, 0% fault, 20% actuator fault, 7mm dia. holes.

6.4 Conclusion

In this study, a new global optimization method called particle swarm optimization was implemented via matlab to use in control of linear and non-linear systems. The most important advantage of particle swarm optimization algorithm is that it requires less number of iterations and it enables us to deal with a few lines of computer code in a cheapest manner. It requires only primitive mathematical operators in terms of both necessity of more available memory and speed. Then particle swarm optimization method has been successfully applied to the design of coupled tanks system control with meaningfull time domain criteria.

Since the system to be controlled is non-linear, it is not possible to find a single set of parameters for all operating conditions. Therefore, some predetermined operating points have been chosen and the optimum control parameter values for the operating points are evaluated while keeping the Takagi-Sugeno crisp values constant for all operating points within the different ranges. Then, different functions are identified for each controller paramenter (K_p,K_i,K_d) for different operating points and regions based on the reference height of tank 2 by using the predetermined points and least squares curve fitting algorithm. It has been observed that these functions, which derive fuzzy controller parameters, have achieved very satisfactorly systems responses.

7. CONCLUSIONS, DISCUSSIONS AND RECOMMENDATIONS

This study is mainly composed of three parts; first part introduces the classical tuning methods, fuzzy control structure and PSO algorithm. Our focus is to control linear and non-linear systems by fuzzy-PID controller tuned with PSO technique. After studying the evolution of the PSO technique, there was a requirement of revising standard PSO into improved particle swarm optimization in order to ensure robust control on systems and find best performance index. Some simulations on benchmark problems have been performed and it is decided to use the improved PSO for the rest of the studies because of the efficiency of the improved method in terms of ensuring the performance index much more accurate and ensuring less computational time. It is observed that improved PSO has the best performance on the convergence rate and convergence precision compared to standart PSO.

The second part of this thesis is a preliminary study for the third part of the study. The aim is to implement improved PSO technique with fuzzy PID controller on different types of systems such as linear first order plus dead time system, second order plus dead time system and finally second order oscillatory process model. According to the systems that have been taken into account in the analysis, the parameters of PID and the crisp values of the rule base have been tuned offline for minimizing the performance criteria given as IAE integral absolute error. The performance results of the proposed approach have been depicted and it is seen that PSO has been successfully applied to the systems with good performance in terms of maximum over shoot, settling time and rise time. The algorithm is flexible to generalize in the sense that all of the parameters that have been tuned offline can be optimized with online opearation. By the guidance of the work on these systems and motivated by the good performances achieved, it is decided to implement the proposed method on nonlinear-coupled tanks system to understand the applicability of the proposed study to control of the height of the water in second tank, which is the structure of the third part of the study.

In the third part of the study, fuzzy PID controller with PSO technique has been applied to the control of non-linear and time varying characteristics of the coupled tank water system. The water level of h_2 is taken as input variable. The water levels between 0-0.15, 0.15-0.20 and 0.20-0.30 are chosen respectively as 3 typical operating regions of h_2 and input space is divided into three fuzzy subspaces based on operating regions. Then, the PID parameters are calculated by the proposed approach. Under the simulation tool of Simulink, the control effects of fuzzy PID with PSO algorithm are simulated for different ranges of water level as different operating points. After finding Takagi-Sugeno crisp values for each region by improved PSO technique with offline application, a Matlab function block, which includes functions of PID parameters, has been used to combine different crisp values for different ranges, which makes the fuzzy PID adaptable for possible operating conditions. Different set values of Takagi Sugeno in the Matlab function block are activated according to the height conditions. Also proper fuzzy PID parameters are set automatically during run with online. As a conclusion, the structural parameters are determined during offline design while the tuning parameters are calculated during online adjustment of fuzzy PID controller to enhance the process performance, as well as to accommodate the adaptive capability to system uncertainty and process disturbances. It is important to indicate that in this study, learning process for crisp values is offline and in order to find the proper crisp values with respect to different inputs in terms of height of tank 2, different algorithm runs have been required. Fuzzy PID with PSO algorithm is implemented in this study, which combines fuzzy PID, PSO technique, fuzzy control and PID control to arrange PID parameters and crisp values according to dynamics of controlled plant to achieve fast transient response, high steady state accuracy, good robustness and self-adaptation. The proposed architecture is also tested in the case of sensor noise and systems faults, simulation results showed that the coupled tank systems was successfully controlled with acceptable performance in both cases. The proposed algorithm is generalized in the sense that it tunes all the parameters of the PID controller online and it is expected that better performance is achieved in almost all type of applications especially in processes where disturbances are frequently encountered.

REFERENCES

[1] **Aström, K.** (2002). Control System Design, pp. 216, McGrawHill

[2] **Ogata, A.** (2002). Modern Control Engineering, pp. 683, Prentice Hall

[3] **Aström, K., Hagglund, T.** (1994). PID Controllers p. 135, McGrawHill.

[4] **Jong, J.** (2004). An Internal Model Based Loop Controller Desing for Peak-Current-Mode controlled AC/DC Power Regulators, *IEEE Conference* Volume D, Issue, 21-24 Nov. 2004 pp. 324-328.

[5] **Fang, G., Kwok, N. M., Ha, Q.** (2008). Automatic Fuzzy Membership Function Tuning Using The Particle Swarm Optimization. *Computational Intelligence and Industrial Application*, pp. 324–328.

[6] **Wang, D., Wang, G and Hu, R.** (2008). Parameters Optimization of Fuzzy Controller Based on Particle Swarm Optimization. Intelligent Information Hiding and Multimedia Signal Processing, pp. 917–921.

[7] **Kennedy, J. and Eberhart, R**. (1995). Particle Swarm Optimization. Neural Networks, *IEEE International Conference*, Vol. 4.

[8] **Hassan, R., Cohanim, B., Weck, O.** (2005). A Comparison of Particle Swarm Optimization and The Genetic Algorithm. *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials* Conference.

[9] **Gao, F., Tong, H. Q.** (2006). A Novel Optimal PID Tuning and Online Tuning Based on Particle Swarm Optimization. Computational Intelligence and Software Engineering, pp. 1–4.

[10] **Bandyopadhyay, R., Chakraborty, U. K., Patranabis, D.** (2001). Autotuning a PID Controller: A Fuzzy – Genetic Approach. *Journal of Systems Architecture*, Vol. 47, Issue 7, pp. 663–673.

[11] **Genc, H. M., Yeşil, E., Eksin, I., Güzelkaya, M., Tekin, Ö. A.** (2009). A Rule Base Modification Scheme in Fuzzy Controllers for Time-Delay Systems. *Expert System with Applications*. Vol. 36, Issue 4, pp. 8476–8486.

[12] **Esmin, A. A. A., Aoki, A. R., Lambert-Torres, G.** (2002). Particle Swarm Optmization for Fuzzy Membership Functions Optimization Systems, Man and Cybernetics, 2002 *IEEE International Conference*, pp. 12, Vol. 3.

[13] **Pillay, N., Govender, P.** (2007). A Particle Swarm Optimizatin Approach for Model Independent Tuning of PID Control Loops. Africon, pp. 1–7.

[14] **Mathworks Inc.** (1998). *Optimization Toolbox User's Guide*.

[15] **Kabaoğlu, R. O**. (2009). Support Vector Machines Based Fault Detection, Diagnosis and Fault Tolerant Control Methods, pp. 28-45: *PhD Thesis*, Istanbul Techical University.

[16] **Eksin, I., Gökaşan, M. and Demiröz, M. A.** (1995). Optimum Settings of PID Parameters Via Hybrid Global Optimization Method in Speed Control of An Induction Machine, *Turkish Journal of Electrical Engineering and Computer Sciences*.

[17] **Wang, D., Fan F.** (2009). Parameters Tuning of Fuzzy Controller for Rotated Pendelum Based on Improved Particle Swarm Optimization. *Computational Intellegence and Software Engineering*, 2009. CISE 2009, pp. 1–5.

[18] **Chang, W., Shih, S.** (2010). PID Controller Design of Nonlinear Systems using an Improved Particle Swarm Optimization. Elsevier, pp. 3632–3639, Vol.15, Issue.11.

[19] **Tahir. F., Iqbal, N., Mustafa, G.** (2009). Control of a Nonlinear Coupled Three-Tank System Using Feedback Linearization. *Electrical Engineering, ICEE'09. Third additional Conference*, pp. 1–6.

[20] **Yanwei, C., Hui, Y. and Huidang, Z**. (2009). PID Controller Parameters Tuning in Servo System Based on Chaotic Particle Swarm Optimization. *IT in Medicine & Education, ITIME'09 IEEE International Symposium*, pp. 276–280, Vol. 1.

[21] **Sehab, R., Remy, M., Renotte, C.** (2001). An Approach to Desing Fuzzy PI Supervisor for Nonlinear System. *IFSA World Congress and 20th NAFIPS International Conference*, pp. 89–899, Vol. 2.

[22] **Trelea, I.** (2003). The Partical Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection. Inf. Process. Lett, Vol 85, no. 6, pp. 317-325.

[23] **Corne, D., Dorigo, M., Glover, F.** (2004). New Ideas in Optimizations, Advance Topics in Computer Science, McGraw-Hill.

APPENDICES

APPENDIX A: Particle Swarm Optimization Code

APPENDIX A

```
function [fxmin, xmin, Swarm, history,iw] = pso(psoOptions)
global SwarmSize;
%Initializations
if nargin == 0
    psoOptions = get_psoOptions;
end
%For Displaying
if psoOptions.Flags.ShowViz
    global vizAxes; %Use the specified axes if using GUI or
create a new global if called from command window
    vizAxes = plot(0,0, '.');
    axis([-1000 1000 -1000 1000 -1000 1000]);   %Initially set to
a cube of this size
    axis square;
    grid off;
    set(vizAxes,'EraseMode','xor','MarkerSize',15); %Set it to
show particles.
    pause(1);
end
%End Display initialization

% Initializing variables
success = 0; % Success Flag
iter = 0;   % Iterations' counter
fevals = 0; % Function evaluations' counter

% Using params---
% Determine the value of weight change
w_start = psoOptions.SParams.w_start;   %Initial inertia weight's
value
w_end = psoOptions.SParams.w_end;       %Final inertia weight
w_varyfor =
floor(psoOptions.SParams.w_varyfor*psoOptions.Vars.Iterations);
%Weight change step. Defines total number of iterations for which
weight is changed.
w_now = w_start;
inertdec = (w_start-w_end)/w_varyfor; %Inertia weight's change
per iteration

% Initialize Swarm and Velocity
SwarmSize = psoOptions.Vars.SwarmSize;
Swarm = rand(SwarmSize, psoOptions.Vars.Dim)*(psoOptions.Obj.ub-
psoOptions.Obj.lb) + psoOptions.Obj.lb;% there might be a wrong
it need a dot before *
VStep = rand(SwarmSize, psoOptions.Vars.Dim);

f2eval = psoOptions.Obj.f2eval; %The objective function to
optimize.

%Find initial function values.
fSwarm = feval(f2eval, Swarm);
fevals = fevals + SwarmSize;

% Initializing the Best positions matrix and
% the corresponding function values
PBest = Swarm;
```

```
fPBest = fSwarm;

% Finding best particle in initial population
[fGBest, g] = min(fSwarm);
lastbpf = fGBest; % last best point function (holding islemi
goruyor)
Best = Swarm(g,:); %Used to keep track of the Best particle ever
fBest = fGBest;% ???
history = [0, fGBest];
iw = [w_start, w_now];
if psoOptions.Disp.Interval & (rem(iter,
psoOptions.Disp.Interval) == 0)
    disp(sprintf('Iterations\t\tfGBest\t\t\tfevals'));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                    THE  PSO  LOOP                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while( (success == 0) & (iter <= psoOptions.Vars.Iterations) )
    iter = iter+1;

    % Update the value of the inertia weight w ilk iterasyon 0.95
le
    % basliyor sonra 2. iterasyondan sonra degisme vuku buluyor.
    if (iter<=w_varyfor) & (iter > 1)
        w_now = w_now - inertdec; %Change inertia weight
    end


    %%%%%%%%%%%%%%%%%%
    % The PLAIN PSO %

    % Set GBest
    A = repmat(Swarm(g,:), SwarmSize, 1); %A = GBest. repmat(X,
m, n) repeats the matrix X in m rows by n columns.
    B = A; %B will be nBest (best neighbor) matrix

    % Generate Random Numbers (Notice normally R1 and R2 must be
a number between 0 and 1)
    R1 = rand(SwarmSize, psoOptions.Vars.Dim);
    R2 = rand(SwarmSize, psoOptions.Vars.Dim);

    % Calculate Velocity
    %SWPSO-Random
    VStep = w_now*VStep + psoOptions.SParams.c1*R1.*(PBest-Swarm)
+ psoOptions.SParams.c2*(1-w_now/2)*R2.*(A-Swarm);
    %SWPSO
    %VStep = w_now*VStep + psoOptions.SParams.c1*R1.*(PBest-
Swarm) + psoOptions.SParams.c2*(1-w_now/2)*(A-Swarm);
    %WPSO
    %VStep = w_now*VStep + psoOptions.SParams.c1*R1.*(PBest-
Swarm) + psoOptions.SParams.c2*R2.*(A-Swarm);
    % Apply Vmax Operator for v > Vmax
    changeRows = VStep > psoOptions.SParams.Vmax;
    VStep(find(changeRows)) = psoOptions.SParams.Vmax;
    % Apply Vmax Operator for v < -Vmax
    changeRows = VStep < -psoOptions.SParams.Vmax;
    VStep(find(changeRows)) = -psoOptions.SParams.Vmax;
```

```
    % ::UPDATE POSITIONS OF PARTICLES::
    Swarm = Swarm + psoOptions.SParams.Chi * VStep;    % Evaluate
new Swarm

    fSwarm = feval(f2eval, Swarm);
    fevals = fevals + SwarmSize;

    % Updating the best position for each particle
    changeRows = fSwarm < fPBest;
    fPBest(find(changeRows)) = fSwarm(find(changeRows));
    PBest(find(changeRows), :) = Swarm(find(changeRows), :);

    lastbpart = PBest(g, :);
    % Updating index g
    [fGBest, g] = min(fPBest);

    %Update Best. Only if fitness has improved.
    if fGBest < lastbpf
        [fBest, b] = min(fPBest);
        Best = PBest(b,:);
    end

    %%OUTPUT%%

        history((size(history,1)+1), :) = [fGBest];
        iw((size(iw,1)+1), :) = [w_now];

    if psoOptions.Disp.Interval & (rem(iter,
psoOptions.Disp.Interval) == 0)
        disp(sprintf('%4d\t\t\t%.5g\t\t\t%5d', iter, fGBest,
fevals));
    end

    if psoOptions.Flags.ShowViz
        [fworst, worst] = max(fGBest);
        DrawSwarm(Swarm, SwarmSize, iter, psoOptions.Vars.Dim,
Swarm(g,:), vizAxes);
    end

    %%TERMINATION%%
    if abs(fGBest-psoOptions.Obj.GM) <= psoOptions.Vars.ErrGoal
%GBest
        success = 1;
    elseif abs(fBest-psoOptions.Obj.GM)<=psoOptions.Vars.ErrGoal
%Best
        success = 1
    else
        lastbpf = fGBest; %To be used to find Best zamansingoto
zamazingo to zamazingo to zamazingo to
    end


end
history;
[fxmin, b] = min(fPBest);
xmin = PBest(b, :);
history = history(:,1);
```

```
iw = iw(:,1);
figure;
plot(history);
figure;
plot(iw);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                    VISULATION OF PARTICLES                        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DrawSwarm >> Internal function.
% Purpose: To draw a visual display of the Swarm.
%
function DrawSwarm(Swarm, SwarmSize, Generation, Dimensions,
GBest, vizAxes)
X = Swarm;
if Dimensions >= 3
    set(vizAxes,'XData',X(1, :),'YData', X(2,:), 'ZData',
X(3,:));
elseif Dimensions == 2
    set(vizAxes,'XData',X(1, :),'YData', X(2,:));
end

GenDiv = 100;
xAx = GBest(1);
yAx = GBest(2);
zAx = GBest(2);

zf = 100 * 50/Generation; %zoom factor

if rem(Generation, GenDiv) == 0
    axis([xAx-zf xAx+100 yAx-zf yAx+zf zAx-zf zAx+zf]);
end

title(Generation);
drawnow;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  INITIAL VALUES DEFINED IN TERMS of Structure Algorithm  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% get_psoOptions :   A structure type of function which is going
to be used
%                    for setting the necessary values from
outside of the
%                    PSO Algorithm.
%
% Usage          :   psoOptions = get_psoOptions
% Arguments      :   None
% Return Values  :   psoOptions

function psoOptions = get_psoOptions()

psoOptions = struct('Flags', struct('ShowViz',1),...
                    'Vars',
struct('SwarmSize',15,'Iterations',250,'ErrGoal',1e-
10,'Dim',3),...

'SParams',struct('c1',2,'c2',2,'w_start',0.95,'w_end',0.4,'w_vary
for',1,'Vmax',50,'Chi',1),...
```

```
                    'Obj',struct('f2eval','cost1','GM',0,'lb',-
100,'ub',100),...
                    'Terminate',struct('Iters',1,'Err',1),...

'Disp',struct('Interval',10,'Header',1,'Progress',1,'Footer',1),.
..

'Save',struct('File','Results','IncludeFnName',1,'IncludeDim',1,'
IncludeSwarmSize',1,'Interval',10,'Header',1,'Footer',1)...
                    );
% Flags = It is used for visulation of the particles while
running ( it
%         takes either 1 or 2 to activate visulation )
% Vars  = Particle Swarm Optimization variables
%
%         SwarmSize----->Number of the paricles defined in swarm
%         Iterations---->Maximum iterations used for stopping
criteria
%         ErrGoal------->It is used for terminating the algorithm
%         Dim----------->Dimension of the particle which is the
coordinates
%                        of the individual particle
%
% SParameters = Strategic parameters for PSO
%
%         c1 = self confidence factor, cognitive acceleration
range in [1.5-2]
%              usually takes 2
%         c2 = swarm confidence factor, Social acceleration range
in [2-2.5]
%              usually takes 2 as well
%         w_start = value of velocity weight at the begining
%              usually takes 0.95
%         w_end = value of velocity weight at the end of the pso
algorithm
%              usually takes 0.4
%         w_varyfor = The fraction of maximum iterations, for
which w is linearly varied
%              usually takes 1 or 0.7
%         Vmax = Maximum velocity step used for limiting the
velocity of
%                the particle, it is used like as below
%
%
%                    Vij,  if |Vij|<= Vmax
%             Vij=  -Vmax, if Vij<-Vmax      REF F. Gao and
H.Q.Tong 2006
%                    Vmax, if Vij>Vmax
%
%         Chi = Constriction Factor used for evaluating the
positions of
%               the particles and formulated below
%
%             Chi = 2k/|2-@-sqrt(@^2-4*@)| where k=1, @=c1+c2 so
that
%             usually takes Chi=1
%
% Obj = Objective Function Options
```

```
%
%         f2eval = Function or system to be optimized ie
Benchmark
%                  functions as a function or Systems embedded in
Simulink
%         GM = value of the global minima which is used for
stopping
%         criteria as well.
%         lb = Lower bound of initialization of the swarm
coordinates
%         ub = Upper bound of initialization of the swarm
coordinates

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  COMMUNICATION WITH SIMULINK & PSO ALGORITHM                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function z=cost(x)
global SwarmSize
global  k1
global k2
global  k3
global k4
global  k5
global k6
global  k7
global k8
global  k9
global Kp
global Ki
global Kd
A=[];
for i=1:SwarmSize
k1=x(i,1);
k2=x(i,2);
k3=x(i,3);
k4=x(i,4);
k5=x(i,5);
k6=x(i,6);
k7=x(i,7);
k8=x(i,8);
k9=x(i,9);
Kp=x(i,10);
Ki=x(i,11);
Kd=x(i,12);
sim('tolgakaya071212');
A=[A IAE(2001)];
end
z=A';
```

CURRICULUM VITAE

Name Surname:	**Tolga KAYA**
Place and Date of Birth:	**04.11.1983**
Address:	**Turgut Mh. Merkez Cad. No:7 Yuvam Turgut Konutları A-2/5 Daire 6, Kocaeli**
E-Mail:	**tkaya10@ford.com**
B.Sc.:	**Yeditepe University – Systems Control Engineering**