Chapter I

PID

CONTROLLER

Chapter II

PSO & DE ALGORITHMS

Chapter III

TUNING

PID CONTROLLER

Chapter IV

SIMULATION AND RESULTS

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Power and Control

Final Year Project Report Presented in Partial Fulfilment of
the Requirements for the Degree of

MASTER

In **Electrical and Electronic Engineering**

Option: **Control**

Title:

PID Controller Tuning using DE and PSO Optimization Techniques

Presented by:

- **ALEM Mohamed Mounir.**
- **SAIDOUN Hakim.**

Supervisor:

Dr, Ouadi

Registration Number:……../2016

Dedications

First of all we thank Allah the most almighty and most merciful for guiding us during our life to follow the right ways and we pray him to show us the path to success.

Then, I dedicate this work to my parents who helped me to get the ball rolling from the beginning, and also to my sisters and my brothers and all the members of my family.

I shall add special dedications to my friends that I have met in the University of Boumerdes and all the others that I left years ago.

Finally, I dedicate this work to everyone who contributed from far or close to the realization of this project and all the good and beneficial things that I did in my life.

S.Hakim

I, dedicate this project to my loved parents, my brothers from the eldest to the youngest and all the members of my family for their much appreciated support and encouragement.

I would thank all my friends, the students who I studied with, and all the staff of teachers who encouraged me to do this work.

A.mohamed Monir

ACKNOWLEDGEMENT

First and foremost we are thankful to God almighty, for showering heavenly blessing upon us, as without that nothing would have been possible.

We would like to express our felt gratitude to our supervisor Dr. OUADI for believing in our ability to work and enriching us with knowledge throughout our project work that crowned our efforts with success. His profound insight and working style has inspired us. The invaluable guidance and support that he has offered us has deeply encouraged us.

Special thanks to all IGEE Teachers and Staff.

SAIDOUN Hakim

ALEM Mohamed Monir

Boumerdes University

Institute of Electrical and Electronics Engineering

Control Engineering

Abstract

The PID controllers are the most popular and widely used controllers in industry because of their simplicity; robustness and successful practical application that can provide excellent control performance despite the varied dynamic characteristics of plant .Determination or tuning of the PID parameters continues to be important as these parameters have a great influence on the stability and performance of the control system. Most of the processes are complex and nonlinear in nature resulting into their poor performance when controlled by traditional methods. The need for improved performance of the process has led to the development of optimal controllers. In this project, two modern heuristic techniques which are particle swarm optimization (PSO) and Differential evolution algorithm (DE) are used for tuning and optimizing PID parameters. A classical method, which is Ziegler and Nichols method, is also used. Finally, a comparison is made between classical method (ZN) and optimization methods (PSO and DE) are applied on high order system, the proposed methods were indeed more efficient and robust in improving the control performances of the PID algorithm.

Table of Contents

Chapter I: PID Controller

Chapter III: Tuning PID Controller

Chapter IV: Simulation and Results

Chapter I: PID Controller

Chapter II: PSO & DE Algorithms

Chapter III: tuning PID Controller

Chapter IV: Simulation and results

PID control is a generic feedback control technology and it makes up 90% of automatic controllers in industrial control systems. The PID control was first placed in the market in 1939 and has remained the most widely used controller in process control until today. Its widespread acceptability can be recognized by: the familiarity with which it is perceived amongst researchers and practitioners within the control community, relative ease and high speed of adjustment with minimal down-time and wide range of applications where its reliability and robustness produces excellent control performances. Other factors that attracted industries to choose PID could be due to low cost, easy to maintain, as well as simplicity in control structure and easy to understand. However, improper PID parameters tuning could lead to cyclic and slow recovery, poor robustness and the worst case scenario would be the collapse of system operation because many industrial plants are often burdened with problems such as high order, time delays, and nonlinearities. Traditionally, these parameters are determined by a trial and error approach. Manual tuning of PID controller is very tedious, time consuming and laborious to implement, especially where the performance of the controller mainly depends on the experiences of design engineers. In recent years, many tuning methods have been proposed for the tuning of PID controllers, such as Gain-phase margin, Cohen-Coon method and classical tuning rules proposed by Ziegler and Nichols, but in general, it is often hard to determine optimal or near optimal PID parameters using these methods [5].

For these reasons, it is highly desirable to increase the capabilities of PID controllers by adding new features. Differential Evolution (DE) and Particle Swarm Optimization (PSO) are one of the modern heuristic optimization algorithms. DE has been found to be a promising algorithm in numeric optimization problems. It has been proposed by Storn and Price. PSO first introduced by Kennedy and Eberhart. It was developed through simulation of a simplified social system, these techniques have been found to be robust in solving continuous nonlinear optimization problems. In addition, they can generate a high-quality solution within shorter calculation time and stable convergence characteristic than other stochastic methods.

In this work, tuning PID controller using two popular heuristic optimization approaches, the DE and PSO are proposed. The performance of DE and PSO in searching globally optimal PID parameters and its reliability to maintain the optimum value for several independent trials has been investigated for a high order open loop control systems, one of the most important step in applying these algorithms is to choose the fitness functions that are used to evaluate the goodness of each particle.

The overall performance (speed of convergence, efficiency, and optimization accuracy) of these algorithms depends on Fitness Function (FF). Our project constitutes the following chapters: the first chapter explains briefly the basic theory and the fundamental of PID controller. The second chapter introduces the PSO and DE methods and shows their importance in solving engineering optimization problems. The third chapter describes the implementation of PID tuning using PSO and DE techniques and Ziegler-Nichols tuning method. Finally the fourth chapter shows the result obtained through simulation on high order system.

1.1 Introduction:

Control system represents a very common class of embedded systems. A control system seeks to make a physical system's output track a desired reference input by setting physical system inputs. It consists of subsystem and process assembled for controlling the output of the process. One of the main components of a control system is PID controller, it is the brain of the control system. The controller receives the signal transmitted by the transmitter and compares it with the desired value. Depending upon the result of the comparison the controller decides what to do to maintain the output at the desired value. Through this chapter we will explain briefly the basic theory and the fundamental of PID controller.

1.2 PID controller definition

A proportional–integral–derivative controller (PID controller) is a control loop feedback mechanism (controller) commonly used in industrial control systems. The mnemonic PID refers to the first letters of the names of the individual terms that make up the standard three-term controller. These are P for the proportional term, I for the integral term and D for the derivative term in the controller. This means that the control action has three basic modes. The first one is directly interacts with a function or an input signal, the second one interacts directly with the integration of the input function, and the third one interacts directly with the derivative of the input function. These three basic modes can control any process separately, so that only one can be used, two or all of them in the process control because Some applications may require using only one or two terms to provide the appropriate system control. This is achieved by setting the other parameters to zero.

PID controller remains an important control tool for three reasons: past record of success, wide availability and simplicity in use. These reasons reinforce one another, thereby ensuring that the more general framework of digital control with higher order controllers has not really been able to displace PID control. It is really only when the process situation demands a more sophisticated controller or a more involved controller solution to control a complex process that the control engineer uses more advanced techniques. Even in the case where the complexity of the process demands a multi-loop or multivariable control solution, a network based on PID control building blocks is often used [1].

1.3 Main structure

PID control is the method of feedback control that use the PID controller as the main tool, the basic structure of conventional feedback control system is shown in Figure 1.1 using a block diagram representation. In this figure, the process is the object to be controlled .the purpose of control is to make the process variable y follow the set-point value r, to achieve this purpose, the manipulated variable μ is changed at the command of the controller.

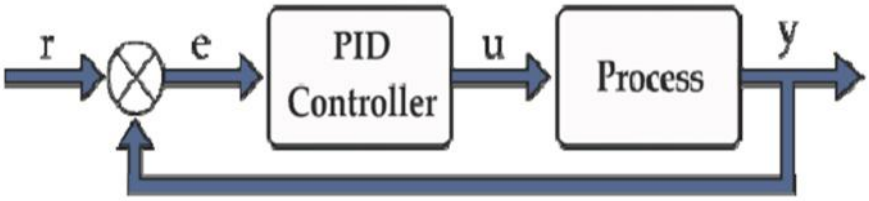

Figure 1.1: Block diagram of the simplest closed-loop system.

- The error e is defined by $e = r - y$

Early the PID control system had exactly the structure of Figure 1.1 .where the PID controller is used as the compensator C(s) .when used in this way, the three elements of the PID controller produce outputs with the following nature:

- P element: proportional to the error at the instant t, which is the “present" error. For example, if the error is large and positive, the control output will also be large and positive.
- I element: proportional to the integral of the error up to the instant t, which can be interpreted as the accumulation of the “past” error. For example, if the current output is not sufficiently strong, error will accumulate over time, and the controller will respond by applying a stronger action.
- D element: proportional to the derivative of the error at the instant t , which can be interpreted as the prediction of the “future” error , based on its current rate of change.

Thus, the PID controller can be understood as a controller that takes the present, the past, and the future of the error into consideration [3].

1.4 PID representation

Some applications may require using only one or two terms to provide the appropriate system control. This is achieved by setting the other parameters to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions [2].

The theoretical basis for analyzing the performance of PID control is considerably aided by the simple representation of an Integrator by the Laplace transform [$\frac{1}{s}$] and a Differentiator using [s]. Conceptually, the PID controller is quite sophisticated and three different representations can be given. First, there is a symbolic representation (Figure 1.2(a)), where each of the three terms can be selected to achieve different control actions. Secondly, there is a time domain operator form (Figure 1.2(b)), and finally, there is a Laplace transform version of the PID controller (Figure 1.2(c)). This gives the controller an *s*-domain operator interpretation and allows the link between the time domain and the frequency [1].

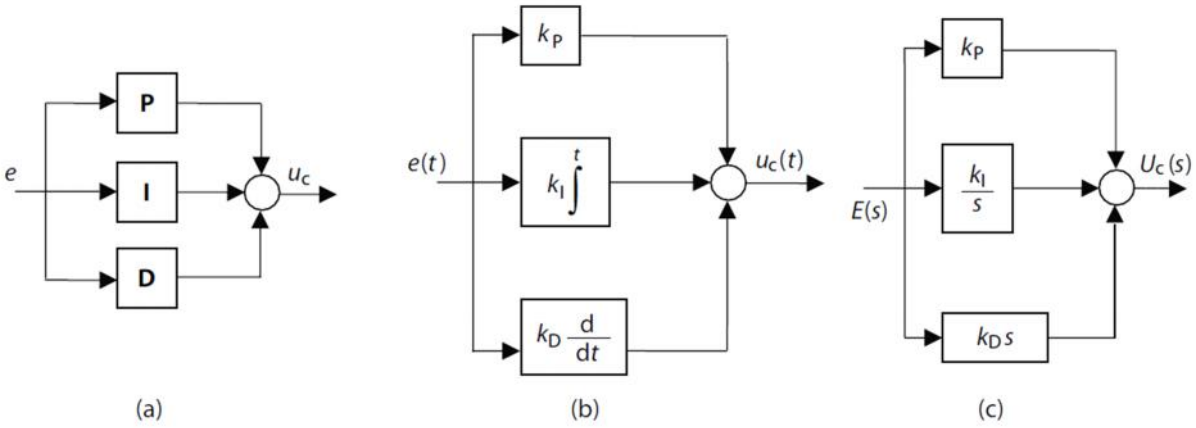

Figure 1.2: PID controller representations.

Such that:

- the Controller input (system error) is : e and the PID control signal U_C
- Time domain forms e(t), uc(t) and Laplace domain forms E(s), Uc(s)
- Proportional gain Kp , Integral gain Ki ,Derivative gain Kd.

1.5 PID controller theory

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining u(t) as the controller output, the final form of the PID algorithm is [2].

$$U(t) = MV(t) = Kp\, e(t) + Ki \int_0^t e(\tau)\, d\tau + Kd \frac{de(t)}{dt} \qquad (1.1)$$

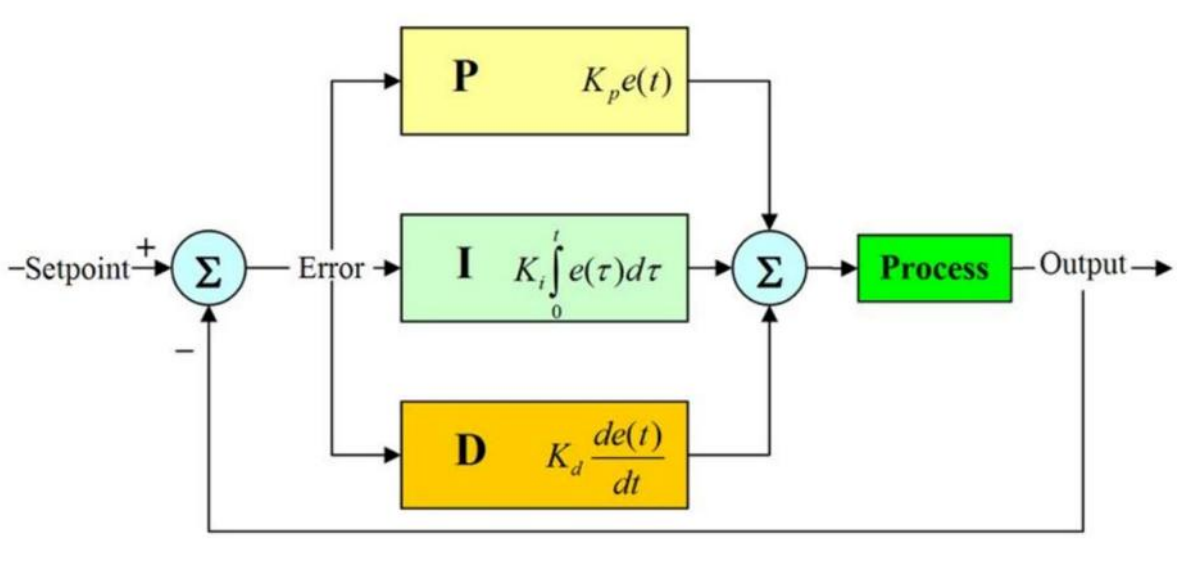

Figure 1.3: Block Diagram of PID Controller

Where:

Kp: Proportional gain, a tuning parameter

Ki: Integral gain, a tuning parameter

Kd: Derivative gain, a tuning parameter

e: Error =SP (set point) – PV (process variable)

t: Time or instantaneous time (the present)

τ: Variable of integration; takes on values from time 0 to the present t.

Equivalently, the transfer function in the Laplace Domain of the PID controller is

$$L(s) = Kp + Ki/s + Kd\,s \quad (1.2)$$

1.5.1 Proportional term

Proportional control is denoted by the P-term in the PID controller. It used when the controller action is to be proportional to the size of the process error signal $e(t) = r(t) - ym(t)$. The time and Laplace domain representations for proportional control are given as:

- Time domain $Uc(t) = Kp\,e(t)$ (1.3)
- Laplace domain $Uc(s) = Kp\,E(s)$ (1.4)

Where the proportional gain is denoted Kp. Figure 1.4 shows the block diagrams for proportional control [1].

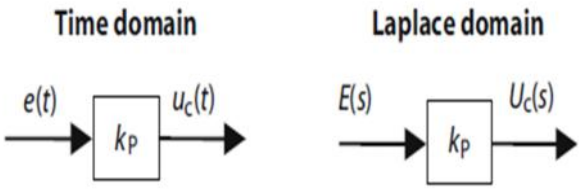

Figure 1.4: Block diagrams for proportional control term.

A high proportional gain results in a large change in the output for a given change in the error.

- If the proportional gain is too high, the system can become unstable
- If the proportional gain is too low, the control action may be too small when responding to system disturbances.

The Figure below shows a plot of the process variable versus time for three value of Kp (Ki and Kd held constant)

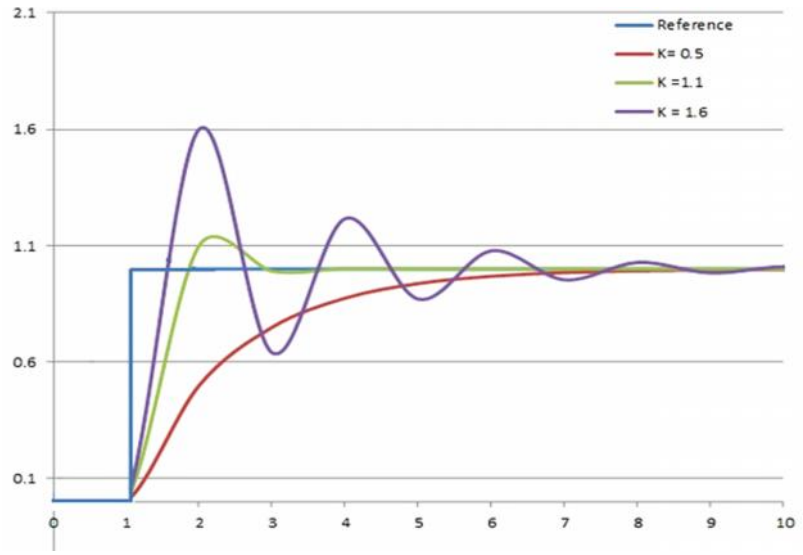

Figure 1.5: process variable versus time with different kp values.

1.5.2 Integral term

Integral control is denoted by the I-term in the PID controller. The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error.

The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain (Ki) and added to the controller output.

The time and Laplace domain representations for integral control are given as:

- Time domain $\mu_c(t) = Ki \int^t e(\tau)d\tau$ (1.5)
- Laplace Domain $Uc(s) = [\frac{Ki}{S}]E(s)$ (1.6)

Where the integral controller gain is denoted ki .The time and Laplace block diagrams are shown in Figure 1.6.

Figure 1.6: Block diagrams for integral control term.

The integral term accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value [2].

The Figure below shows a plot of the process variable versus time for three value of Ki (Kp and Kd held constant).

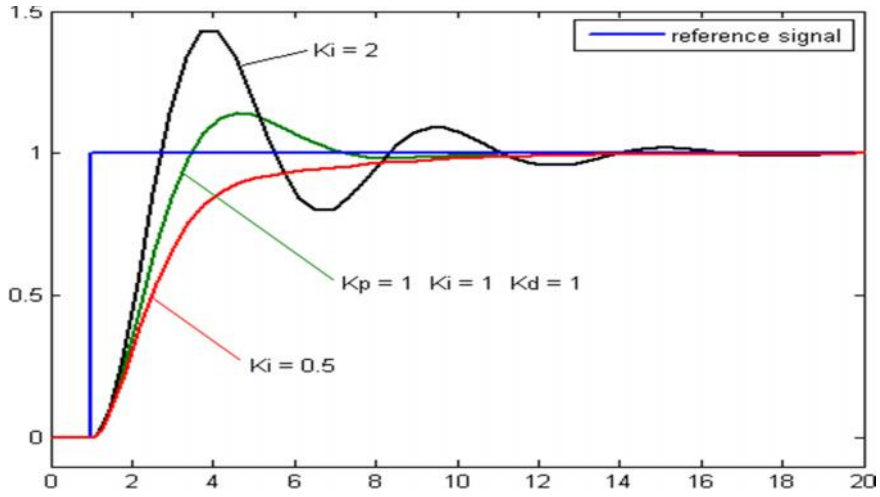

Figure 1.7: process variable versus time with different Ki values.

1.5.3 Derivative term

If a controller can use the rate of change of an error signal as an input, then this introduces an element of prediction into the control action. Derivative control uses the rate of change of an error signal and is the D-term in the PID controller. The time and Laplace domain representations for derivative control are given as [1]:

- Time domain: $\mu_c(t) = \mathrm{Kd}\frac{de}{dt}$ (1.7)
- Laplace domain: $Uc(s) = [kd\ s]E(s)$ (1.8)

Where the derivative control gain is denoted kd. This particular form is termed pure derivative control, for which the block diagram representations are shown in Figure 1.8.

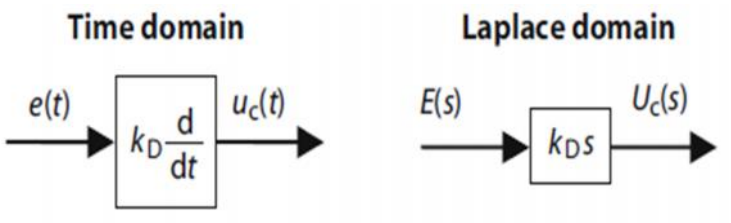

Figure 1.8: Block diagrams for derivative control term.

Derivative action predicts system behavior and thus improves settling time and stability of the system.

The Figure below shows a plot of the process variable versus time for three value of Kd (Kp and Ki held constant).

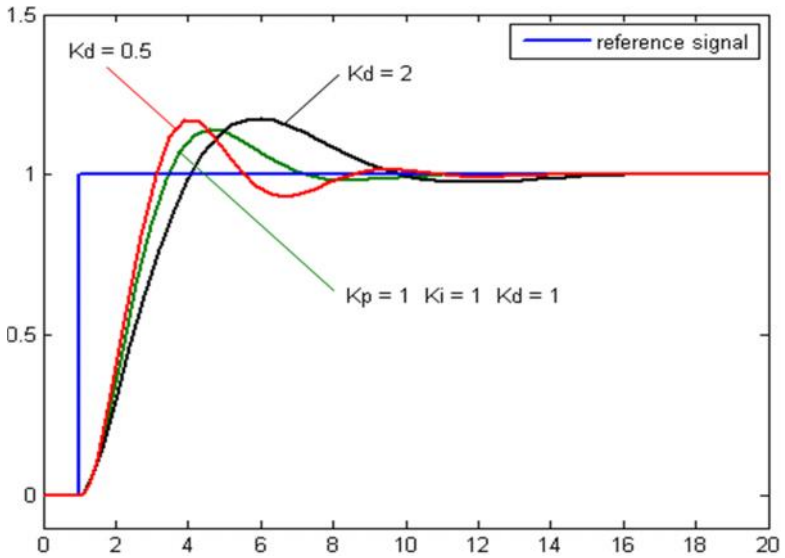

Figure 1.9: process variable versus time with different Kd values.

1.6 Stability

If the PID controller parameters (the gains of the proportional, integral and derivative terms) are chosen incorrectly, the controlled process input can be unstable, i.e., its output diverges, with or without oscillation, Generally, stabilization of response is required and the process must not oscillate for any combination of process conditions and setpoints.

Mathematically, the origins of instability can be seen in the Laplace domain. The total loop transfer function is [2].

$$H(s) = \frac{K(s)\,G(s)}{1+K(s)\,G(s)} \tag{1.9}$$

Where:

$K(s)$: PID transfer function

$G(s)$: Plant transfer function

1.7 The characteristics of P, I & D controller

A proportional controller (KP) will have the effect of reducing the rise time and will reduce, but never eliminate, the steady-state error. An integral control (KI) will have the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative control (KD) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. Effects of each of the three parameters on a closed-loop system are summarized in Table 1.1.

Table 1.1: The effects of PID parameters on a closed loop system.

Response	**Rise Time**	**Overshoot**	**Settling Time**	**Steady-State Error**	**Stability**
Increasing Kp	Decrease	Increase	Small Change	Decrease	Degrade
Increasing Ki	Decrease	Increase	Increase	Eliminate	Degrade
Increasing Kd	Small Change	Decrease	Decrease	Small Change	Improve

The correlations may not be exactly accurate, because Kp, Ki, and Kd are dependent of each other. In fact, changing one of these variables can change the effect of the other two. For this reason, the table1 should be used only as a reference when we are determining the values for Kp, Ki and Kd.

2.1 Introduction

Optimization problems are widely encountered in various fields in science and technology. Optimization algorithms are another area that has been receiving increased attention in the past few years by the research community as well as the industry. An optimization algorithm is a numerical method or algorithm for finding the maxima or the minima of a function operating with certain constraints. Computational intelligence (CI) is a successor of artificial intelligence relying on Evolutionary computation, which is a famous optimization technique. Computational intelligence combines elements of learning; adaptation and evolution to create programs that are, in some sense, intelligent. Computational intelligence research does not reject statistical methods, but often gives a complementary view. Computational intelligence finds its fundamental application in the area of fitness function design, methods for parameter control, and techniques for multimodal optimization. The importance of CI lies in the fact that these techniques often find optima in complicated optimization problems more quickly than the traditional optimization methods [5].

Recently, particle swarm optimization (PSO) and differential evolution (DE) have been introduced and particularly PSO has received increased interest from the EC community.
Both techniques have shown great promise in several real-world applications [6].

During this chapter, we will see the main steps of PSO and DE algorithms. We will focus on their working principles.

2.2 Particle swarm optimization algorithm

2.2.1 Historical background

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart (electrical engineer) and Dr. Kennedy (social psychologist) in 1995. It is a computational algorithm based on swarm intelligence. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. This method is motivated by the observation of social interaction and animal behaviors such as fish schooling and bird flocking. It mimics the way they find food by the cooperation and competition among the entire population. A swarm consists of individuals, called particles, each of which represents a different possible set of the unknown parameters to be optimized. The swarm is initialized with a population of random solutions. In a PSO system, particles fly around in a multi-dimensional search space adjusting its position according to its own experience and the experience of its neighboring particle. The goal is to efficiently search the solution space by swarming the particles towards the best

fitting solution encountered in previous iterations with the intention of encountering better solutions through the course of the process and eventually converging on a single minimum or maximum solution. The performance of each particle is measured according to a pre-defined fitness function, which is related to the problem being solved [7]. PSO has been regarded as a promising optimization algorithm due to its simplicity, low computational cost and good performance [5].

2.2.2 Theory of PSO

The idea of PSO is that, the system is initialized with a population of random solutions called particles (individuals) [8]. Each particle is treated as a point in a D-dimensional space which adjusts its "flying" according to its own flying experience as well as the flying experience of other particles. All particles have fitness values, evaluated through the fitness function and velocities. The two variables which are iteratively changed in PSO algorithm are the following ones [9]:

- **Pbest** (personal best): each particle keeps track of its coordinates in the solution space which are associated with the best solution (fitness) that has achieved so far by that particle;
- **Gbest** (global best): another best value that is tracked by the PSO is the best value obtained so far by any particle in the neighborhood of that particle;

The PSO concept consists of changing velocity of each particle toward its Pbest and the Gbes**t** locations at each time step. Figure 2.1 shows the general concept of modification of search point by PSO.

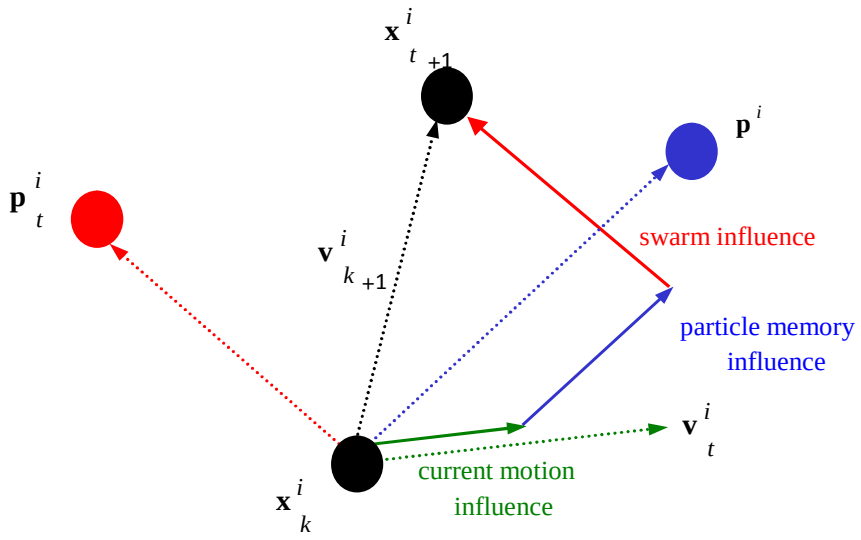

Figure 2.1: General Concept of modification of search point by PSO

Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward Pbest and the Gbest locations. For example, the ith particle is represented as $xi = (xi1, \dots, xid, \dots, xiD)$ in the D- dimensional search space. The best previous position of the ith particle is recorded and represented as $Pbesti = (Pbesti, 1, \dots, Pbesti, d, \dots, Pbesti, D)$. The index of best particle among all of the particles in the group is represented by the Gbestj The rate of the position change (velocity) for particle i is represented as $Vi = (Vi, 1, \dots, Vi, d, \dots Vi, D)$. The modified or the updated velocity and position of each particle can be calculated using the current velocity and the distance from $Pbesti, j$ to $Gbestj$ as shown by equations (2.1) and (2.2) below [10]:

$$Vi, j(t+1) = w.Vi, j(t) + c1r1\left[Pbesti, j(t) - xi, j(t)\right] + c2r2[gbestj(t) - xi, j(t)] \quad (2.1)$$

$$Xi, j(t+1) = Xi, j(t) + Vi, j(t+1) \quad (2.2)$$

Where i=1,2 ,…,n and j=1,2,…D

n number of particles in the swarm(or population);

D number of members in a particle(dimension of problem);

t current iteration number;

$Vi, j(t)$ velocity of particle i(dimension j) at iteration t, $Vjmin \leq Vi, j(t) \geq Vjmax$

w inertia weight factor;

$Xi, j(t)$ current position of particle i at iteration t;

$Pbesti, j(t)$ the individual best position of particle i until iteration t;

$Gbestj(t)$ the best particle in the population at iteration t.

c1, c2 acceleration constants;

r1, r2 random numbers between 0 and 1 regenerated for each velocity update.

The first equation is used to calculate i-th particle's new velocity by taking into consideration three terms: the particle's previous velocity, the distance between the particle's best previous and current position, and, finally, the distance between swarm's best experience (the position of the best particle in the swarm) and i-th particle's current position. Then, following the second equation, the i-th particle flies toward a new position. In general, the performance of each particle is measured according to a predefined fitness function (performance index), which is related to the problem to be solved [11].

2.2.3 PSO parameter selection

There are some parameters in PSO algorithm that may affect its performance. For any given optimization problem, some of these parameter's values and choices have large impact on the efficiency of the PSO method, and other parameters have small or no effect [12]. These are described as follows:

The parameter $Vmax$ determines the resolution, or fitness, with which regions are to be searched between the present position and the target position. If $Vmax$ is too high, then particles will move beyond a good solution, If $Vmax$ is too low, particles will be trapped in local minima. In many experiences, $Vmax$ was often set at 10-20% of the dynamic range of the variable on each dimension.

The constants c1 and c2 represent the weights of the stochastic acceleration terms that pull each particle toward the Pbest and Gbest positions. Low values allow particles to roam far from the target regions before being tugged back. On the other hand, high values result in abrupt movement toward, or past, target regions. Here, the acceleration constants c1 and c2 were often set to be 2.0 according to experiences [12].

Suitable selection of inertia weight w provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution. As originally developed, w often decreases linearly from about 0.9 to 0.4 during a run. In general, the inertia weight is set according to the following equation [13]:

$$w = wmax - \frac{Wmax - Wmin}{itermax} \times iter \qquad (2.3)$$

Where $itermax$ is the maximum number of iterations and $iter$ is the current number of iterations.

Population size is another parameter that necessitates careful selection. A big population generates larger parts of the search space to be covered per iteration. A large number of particles may reduce the number of iterations need to obtain a good optimization result. In contrast, huge amounts of particles increase the computational complexity per iteration, and more time consuming. From a number of empirical studies, it has been shown that most of the PSO implementations use an interval of n ϵ [20, 60] for the swarm size.

The number of iterations to obtain a good result is also problem-dependent. A too low number of iterations may stop the search process prematurely, while too large iterations has the consequence of unnecessary added computational complexity and more time needed [12].

2.2.4 Neighborhood Topologies

A neighborhood must be defined for each particle. This neighborhood determines the extent of social interaction within the swarm and influences a particular particle's movement. Less interaction occurs when the neighborhoods in the swarm are small. For small neighborhood, the convergence will be slower but it may improve the quality of solutions. For larger neighborhood, the convergence will be faster but the risk that sometimes convergence occurs earlier. To solve this problem, the search process starts with small neighborhoods size then the small neighborhoods size is increased over time. This technique ensures an initially high diversity with faster convergence as the particles move towards a promising search region.

The PSO algorithm is social interaction among the particles in the entire swarm.
Particles communicate with one another by exchanging information about the success of each particle in the swarm. When a particle in the whole swarm finds a better position, all particles move towards this particle. This performance of the particles is determined by the particles' neighborhood. Researchers have worked on developing this performance by designing different types of neighborhood structures [12]. Some neighborhood structures or topologies are discussed below:

a) Star topology

For star topology ach particle connects with every other particle. This topology leads to faster convergence than other topologies, but there is a susceptibility to be trapped in local minima. Because all particles know each other, this topology is referred to as the Gbest PSO.

b) Ring topology

For ring topology each particle is connected only with its immediate neighbors. In this process, when one particle finds a better result, this particle passes it to its immediate neighbors, and these two immediate neighbors pass it to their immediate neighbors, until it reaches the last particle. Thus the best result found is spread very slowly around the ring by all particles. Convergence is slower, but larger parts of the search space are covered than with the star topology.

c) Wheel topology

In wheel topology, only one particle (a focal particle) connects to the others, and all information is communicated through this particle. This focal particle compares the best performance of all particles in the swarm, and adjusts its position towards the best performance particle. Then the new position of the focal particle is informed to all the particles.

d) Four clusters topology

In four clusters topology four clusters (or cliques) are connected with two edges between neighboring clusters and one edge between opposite clusters. These neighborhood structures or topologies are illustrated in Figure 2.2

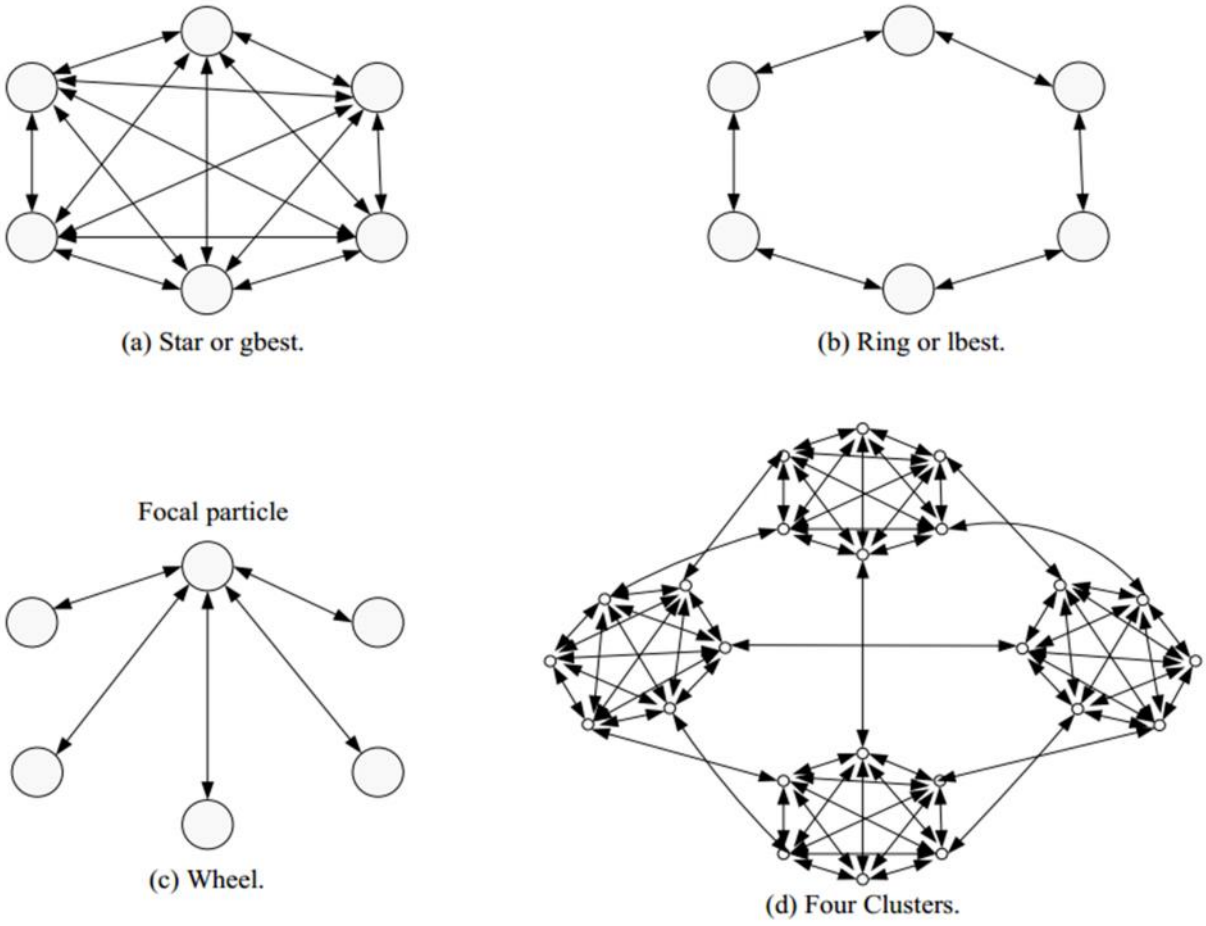

Figure 2.2: Neighborhood topologies [12].

2.2.5 PSO applications

Kennedy and Eberhart established the first practical application of Particle Swarm Optimization in 1995. It was in the field of neural network training and was reported together with the algorithm itself. PSO have been successfully used across a wide range of applications, for instance, telecommunications, system control, data mining, power systems design, combinatorial optimization, signal processing, network training, and many other areas.

Nowadays, PSO algorithms have also been applied to solve constrained problems, multi-objective optimization problems, problems with dynamically changing landscapes, and to find multiple solutions, while the original PSO algorithm was used mainly to solve unconstrained, single-objective optimization problems. Various areas where PSO is applied are listed below [12].

a) Power systems and plants

- Power control and optimization.
- Control of photovoltaic systems.
- Large-scale power plant control.
- Fault-tolerant control of compensators.

b) Control

- Automatic generation control tuning.
- Design of controllers.
- Power plants and systems control.
- Process and combustion control.

c) Communication networks

- Bluetooth networks.
- Auto tuning for universal mobile telecommunication system networks.

d) Robotics

- Control of robotic manipulators and arms.
- Motion planning and control.

e) Signal processing

- Pattern recognition of flatness signal.
- Design of IIR filters and nonlinear adaptive filters.

2.2.6 Advantages and Disadvantages of the PSO

It is said that PSO algorithm is the one of the most powerful methods for solving the non-smooth global optimization problems while there are some disadvantages of the PSO algorithm. The advantages and disadvantages of PSO are discussed below: [12]

2.2.6.1 Advantages of the PSO algorithm

- PSO is based on the intelligence. It can be applied into both scientific research and engineering use.
- PSO have no overlapping and mutation calculation. The search can be carried out by the speed of the particle. During the development of several generations, only the most optimist particle can transmit information onto the other particles, and the speed of the researching is very fast.
- The calculation in PSO is very simple. Compared with the other developing calculations, it occupies the bigger optimization ability and it can be completed easily.
- PSO adopts the real number code, and it is decided directly by the solution. The number of the dimension is equal to the constant of the solution.
- PSO is less dependent of a set of initial points.
- Very efficient global search algorithm.

2.2.6.2 Disadvantages of the PSO algorithm

- PSO easily suffers from the partial optimism, which causes the less exact at the regulation of its speed and the direction.
- The method cannot work out the problems of scattering and optimization.
- The method cannot work out the problems of non-coordinate system, such as the solution to the energy field and the moving rules of the particles in the energy field.
- Slow convergence in refined search stage (weak local search ability).

2.3 Differential evolution algorithm

2.3.1 Historical background

At present, a novel evolutionary algorithm called differential evolution is fashionable in a variety of evolutionary algorithms. Differential evolution was invented by K.Price' s and R.Storn attempt to solve the Chebychev Polynomial fitting problem in 1995 and in the same year differential evolution was outstanding at the First International Contest on Evolutionary Computation which was held in Nagoya. Differential evolution turned out to be the best evolution type of algorithm for solving the real-valued test function [14]. Differential Evolution (DE) algorithm is one of the optimization techniques and a kind of evolutionary computation technique. The method has been found to be an effective and robust in solving problems with non-linearity, non-differentiability, multiple optima, and high dimensionality [15]. DE is originated from Genetic algorithm and can be used with greatly convenience without needing of encoding and decoding, furthermore, it has no special requirement about initial values and has fast convergence and good adaptability to a variety of non-linear function. Therefore, the algorithm has gained increasing attention as it was proposed [16].

2.3.2 The theory of Differential evolution

Differential evolution is similar to the overall structure of the genetic algorithm. The main differential is mutation operation [14]. Differential evolution (DE) works with two populations; old generation and new generation of the same population. The size of the population is adjusted by the parameter NP. The population consists of real valued vectors with dimension D that equals the number of design parameters control variables. The population is randomly initialized within the initial parameter bounds. The optimization process is conducted by means of three main operations: mutation, crossover and selection. In each generation, individuals of the current population become target vectors. For each target vector, the mutation operation produces a mutant vector, by adding the weighted difference between two randomly chosen vectors to a third vector. The crossover operation generates a new vector, called trial vector, by mixing the parameters of the mutant vector with those of the target vector. If the trial vector obtains a better fitness value than the target vector, then the trial vector replaces the target vector in the next generation [17].

DE utilize mutation operator as a search mechanism and selection operation to search directly based on prospective regions on search space. DE algorithm is applied based on several steps. Figure 2.3 shows the flow graph of DE algorithm [18].

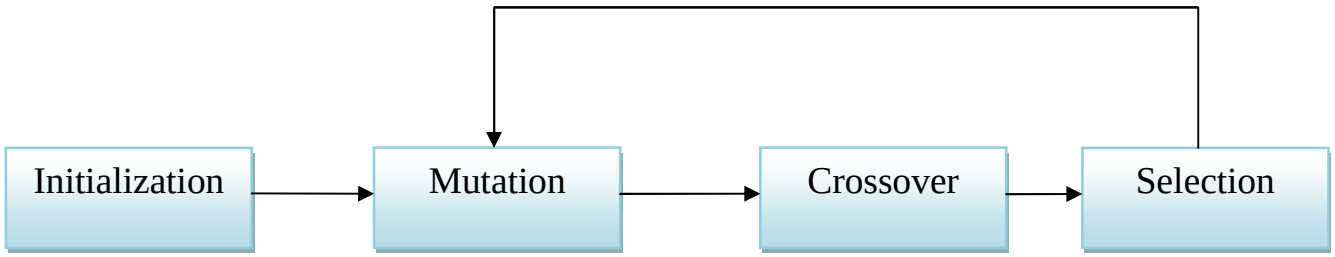

Figure 2.3: Flow graph of differential evolution algorithm.

Step 1: Initialization

An essential part of any evolutionary search is the generation of the initial population. When it is generated successfully, a good solution is found and the search convergences faster. Thus the computational time required to find the good enough solution is directly proportional to the quality of the initial population.

If there is no prior knowledge about the optimum, it is typical to generate the initial population randomly between the lower and upper bounds defined for each parameter [19]. The initial population is then

$$Xi,j(0) = XjL + randj(0,1).[XjH - XjL] \quad (2.4)$$

Where $randj(0,1)$ is a uniformly distributed random number within the range [0,1]

XjL and XjH indicate the lower and upper bounds of the jth parameter vectors Xi,j, respectively.

Step 2: Mutation

Mutation stands for sudden change. For each target vector Xi,G, a mutation vector Vi,G is generated

$$Vj,G+1 = Xr3,G + F.(Xr1,G - Xr2,G) \quad (2.5)$$

Where $Xr1$, $Xr2$ and $Xr3$ are randomly generated from the current population [1 NP], F is a real and constant factor which controls the amplification of the differential variation $(Xr1, G - Xr2, G)$ [20].

Step 3: Crossover (Recombination)

Now for each target vector , G , a trial vector Ui, G is generated. Here for the generation of trial vector Ui, G binomial crossover is used. The steps for crossover are

$$Uij, G+1 = \begin{cases} Uij, G+1 & if\,(randi \leq \mathrm{CR})v(Rnd = i) \\ \mathrm{Xj, G} & \text{Otherwise} \end{cases} \quad (2.6)$$

Where $i = 1, \ldots, D$ and Rnd is a random number between 0 and 1

CR is the crossover probability constant, which is the parameter of this algorithm and controls the diversity of the population to help the algorithm escaping from the local optimal solution, need to be ascertained in advance [21].

Step 4: Selection

After the crossover, DE uses simple one-to-one survivor selection where trial vector Ui, G competes against target vector Xi, G The vector with the lowest fitness function value survives into the next generation G+1 by

$$Xj, G+1 = \begin{cases} \mathrm{Ui, G} & \text{, if } \mathrm{f(Ui, G)} \leq \mathrm{f(Xi, G)} \\ \mathrm{Xi, G} & \text{, Otherwise} \end{cases} \quad (2.7)$$

Where $Xi, G + 1$ is the j-th individual at generation G+1.

Once the new population is installed, the process of mutation, crossover, and selection is repeated until the optimal individual is located or a pre-specified termination criterion is satisfied, e.g., all the individuals are the same [21].

2.3.3 DE Parameter selection [22]

The choice of control parameters has great impact on performance of DE search algorithm. Selecting the DE parameters that yield good performance has therefore been the subject of much research. The key parameters of control are: NP- the population size, CR- the crossover constant, F- the weight applied to random differential (scaling factor). It is worth noting that DE's control variables, NP, F and CR, are not difficult to choose in order to obtain

promising results. Storn have come out with several rules in selecting the control parameters. The rules are listed follow:

- The initialized population should be spread as much as possible over the objective function surface.
- Frequently the crossover probability CR ϵ [0,1] must be considerably lower than one (e.g. 0.3). If no convergence can be achieved, CR ϵ [0.8, 1] often helps.
- For many applications NP=10×D, where D is the number of problem dimension. F is usually chosen at [0.5, 1].
- The higher the population size, NP, the lower the weighting factor F should choose. These rules of thumb for DE's control variables which is easy to work with is one of DE's major contribution.

2.3.4 Advantages and disadvantage of DE algorithm

In the last decade, DE has been frequently used as an optimization algorithm because of its effectiveness in performing difficult optimization tasks. In addition, the scheme obtains better results in a faster and cheaper way compared to several other methods with fewer parameters to adjust, however there still are some disadvantages of the DE algorithm [23]. The advantages and disadvantages of DE are discussed below:

2.3.4.1 Advantages of the DE algorithm

- Ability to handle non-differentiable, nonlinear and multimodal cost functions.
- Ease of use, i.e. few control variables to steer the minimization. These variables should also be robust and easy to choose.
- Good convergence properties, i.e. consistent convergence to the global minimum in consecutive independent trials.

2.3.4.2 Disadvantage of DE algorithm

- Parameter tuning is necessary.
- Same parameters may not guarantee the global optimum solution.
- Easy to drop into the local optimum

2.3.5 DE Applications

The application of differential evolution algorithm can be easily found in real life problems in the field of [24]

- electronic engineering
- electrical engineering
- combinatorial mathematics
- civil engineering
- aeronautical engineering
- operation research
- education sector
- logistic design
- other soft computing techniques

3.1 Introduction

Determination or tuning of the PID parameters continues to be important as these parameters have a great influence on the stability and performance of the control system. Most of the processes are complex and nonlinear in nature resulting into their poor performance when controlled by traditional tuned PID controllers such as Ziegler-Nichols method [25]. For these reasons, it is highly desirable to increase the capabilities of PID controllers by adding new features. Many artificial intelligence (AI) techniques have been employed to improve the controller performances for a wide range of plants while retaining their basic characteristics. Al techniques such as neural network, fuzzy system, and neural-fuzzy logic have been widely applied to proper tuning of PID controller parameters [26].

In this chapter we will describe the implementation of PSO and DE techniques to optimally tune PID controller and Ziegler Nichols as classical tuning method.

3.2 The Ziegler-Nichols PID tuning rule

Ziegler and Nichols published in 1942 a paper [4] where they described two methods for tuning the parameters of P-, PI- and PID controllers. These two methods are the Ziegler-Nichols' closed loop method, and the Ziegler-Nichols' open loop method. In our project closed –loop method is presented .Ziegler and Nichols used the following definition of acceptable stability as a basis for their controller tuning rules: The ratio of the amplitudes of subsequent peaks in the same direction (due to a step change of the disturbance or a step change of the set point in the control loop) is Approximately 1/4, as shown in Figure 3.1

$$\frac{A_2}{A_1}=\frac{1}{4} \tag{3.1}$$

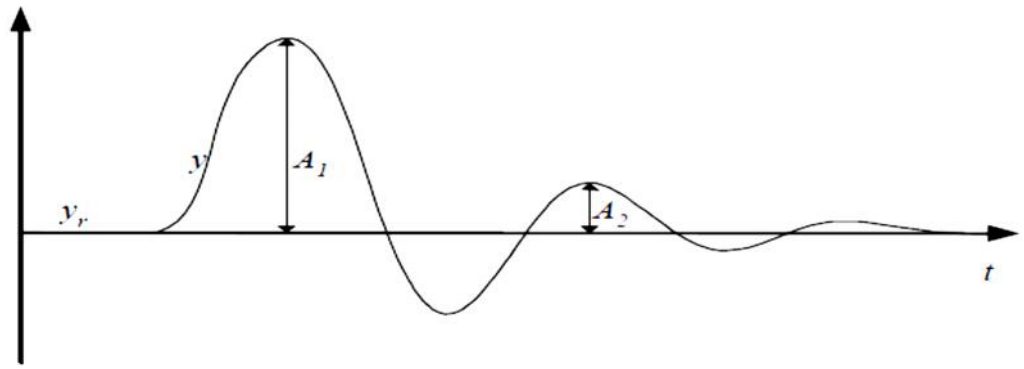

Figure 3.1: the stability of the system according to Ziegler and Nichols

However, there is no guaranty that the actual amplitude ratio of a given control system becomes 1/4 after tuning with one of the Ziegler and Nichols methods, but it should not be very different from 1/4.

3.2.1 The Ziegler-Nichols' PID tuning procedure

The tuning procedure is as follows:

1- Turn the PID controller into a P controller by setting: $T_I = \infty$ and $K_D = 0$. Initially set gain $K_P = 0$.

2- Increase K_P until there are sustained oscillations in the signals in the control system (The sustained oscillations correspond to the system being on the stability limit). This K_P value is denoted the ultimate (or critical) gain, K_{PU}.

3- Measure the ultimate (or critical) period Pu of the sustained oscillations as it is shown In Figure 3.2

4- Calculate the controller parameter values according to Table 3.1, and use these parameter values in the PID controller.

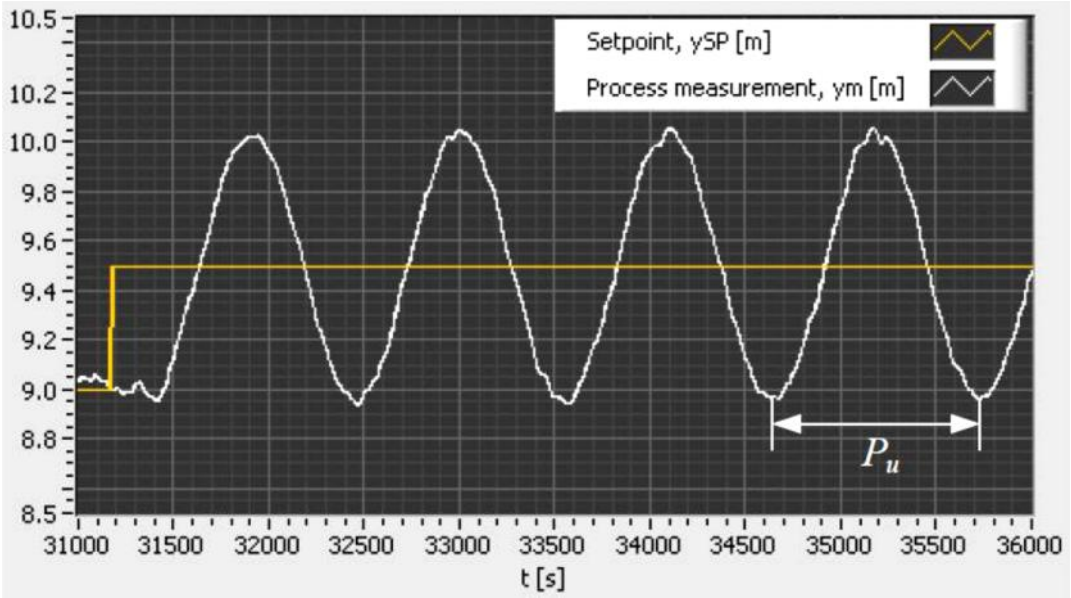

Figure 3.2: The tuning phase of the Ziegler-Nichols' closed-loop

Using the parameters Kpu and Pu, we can set the values of K_P, K_I and K_D according to the formula shown in Table 3.1

Table 3.1: Controller parameters for closed loop Ziegler-Nichols method

Controller	K_P	T_I	T_D
P	$0.5K_{PU}$	∞	0
PI	$0.45K_{PU}$		0
PID	$0.6K_{PU}$		

These parameters will typically give a response with an overshoot about 25% and good settling time. We may then start fine-tuning the controller using the basic rules that relate each parameter to the response characteristics.

3.3 Implementation of PSO-PID Controller

In chapter 2 we discussed the general structure of PSO algorithm, how it is being established, programmed and created. In this section we discuss PSO as an alternative method for tuning the PID controller. Here we define a three dimensional search space in which all the three dimensions represent three different parameters of the PID. Each particular point in the search space represent a particular combination of [KP KI KD] for which a particular response is obtained .The performance of the point or the combination of PID parameters is determined by a fitness function or the cost function [27]. Figure 3.3 shows the basic block diagram of PSO algorithm based PID controller tuning.

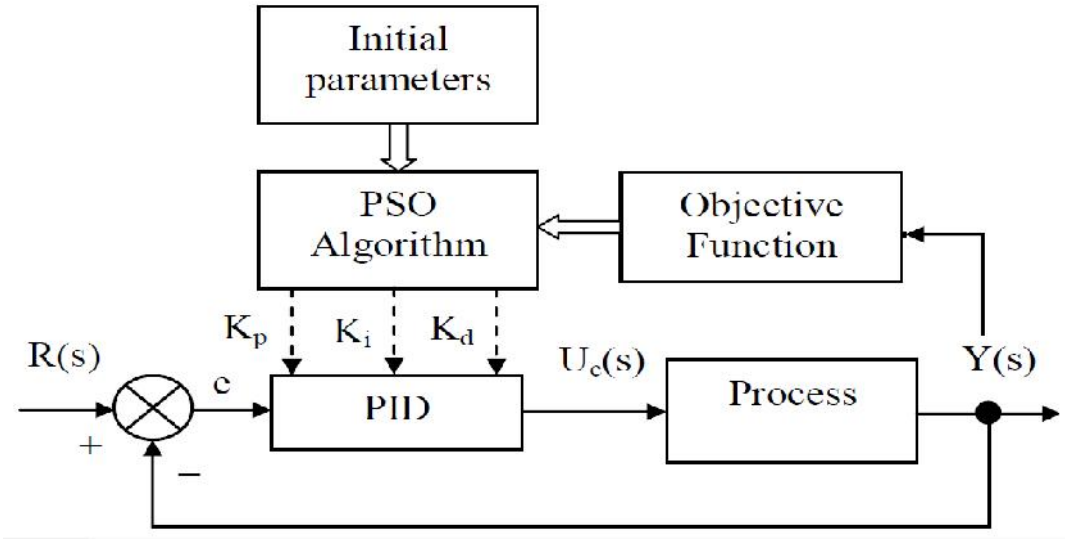

Figure3.3: PSO based PID [27]

3.3.1 Fitness function

In PID controller design methods, the most common performance criteria are Integrated Absolute Error (IAE), Integrated of Time weight Square Error (ITSE) and Integrated of Square Error (ISE) that can be evaluated analytically in frequency domain. Each criterion has its own advantage and disadvantage. For example, disadvantage of IAE and ISE criteria is that its minimization can result in a response with relatively small overshoot but a long settling time, because the ISE performance criteria weights all errors equally independent of time. Although, ITSE performance criterion can overcome this is the disadvantage of ISE criterion. The IAE, ISE, and ITSE performance criterion formulas are as follows: [28]

$$\text{IAE} = \int_0^\infty |e(t)|\,dt \tag{3.2}$$

$$\text{ISE} = \int_0^\infty e^2(t)\,dt \tag{3.3}$$

$$\text{ITSE} = \int_0^\infty te^2(t)\,dt \tag{3.4}$$

$$\text{ITAE} = \int_0^\infty t|e(t)|\,dt \tag{3.5}$$

In this project a time domain criterion is used for evaluating the PID controller. A set of good control parameters P, I and D can yield a good step response that will result in performance criteria minimization in the time domain. These performance criteria in the time domain include the overshoot, rise time, settling time, and steady-state error. In our case, another FF (Fitness Function) is given by equation (3.6) to serve as performance criterion for selection optimal PID controller parameters [29].

$$F(k) = (1 - e^{-\beta})(\mathrm{M}_p + \mathrm{E}_{ss}) + e^{-\beta}(t_s - t_r) \tag{3.6}$$

Where F: Fitness function and k is $[kp, ki, kd]$

M_P: Peak Overshoot

t_s: Settling Time

t_r :Rise Time

Ess: steady state error.

β :Scaling Factor (Depends upon the choice of designer).

3.3.2 Tuning procedure

PSO-PID controller for searching the optimal or near optimal controller parameters kp, ki, and kd, with the PSO algorithm. Each individual K contains three members kp, ki, and kd. The searching procedures of the proposed PSO-PID controller were shown as below [28].

Step1:

We specify the lower and upper bounds of the three controller parameters and initialize randomly the individuals of the population including searching points, velocities, Pbests, and Gbest.

Step 2:

We calculate the evaluation value of each individual in the population using the evaluation function.

Step 3:

We compare each individual‘s evaluation value with its Pbest. The best evaluation value among the Pbest is denoted as Gbest.

Step 4:

We modify the member velocity v of each individual K According to equation (2.1)

$$Vi,j(t+1) = w.Vi,j(t) + c1r1[Pbesti,j(t) - xi,j(t)] + c2r2[Gbestj(t) - xi,j(t)] \quad (2.1)$$

Where $i = 1,2,3.....n.\ \ j = 1,2,3.....D$

And we set the value of w using equation (2.3).

$$w = wmax - \frac{Wmax - Wmin}{itermax} \times iter \quad (2.3)$$

When j is 1, $Vi,1$ represents the change in velocity of kd controller Parameter. When j is 2, Vi,2 represents the change in velocity of kp controller parameter. When j is 3, represents the change in velocity of ki controller Parameter. This means that for an optimization case D=3.

Step 5:

If $Vi,j(t+1) > Vjmax$, then $Vi,j(t+1) = Vjmax$

If $Vi,j(t+1) < Vjmin$, then $Vi,j(t+1) = Vjmin$

Step 6:

We modify the member position of each individual i according to equation (2.2)

$$Xi,j(t+1) = Xi,j(t) + Vi,j(t+1) \quad (2.2)$$

$$Xjmin < Xi,j(t+1) < Xjmax$$

Where $Xjmin$ and $Xjmax$ represent the lower and upper bounds, respectively, of member j of the individual X.

Step 7:

If the number of iterations reaches the maximum, then we go to **Step 9**. Otherwise, we go to **Step2**

Step 8:

We set the members of the individual that generates the latest $Gbest$ as the optimal controller parameters.

Figure 3.4 shows the general flow chart of the PSO algorithm tuning PID controller [5]

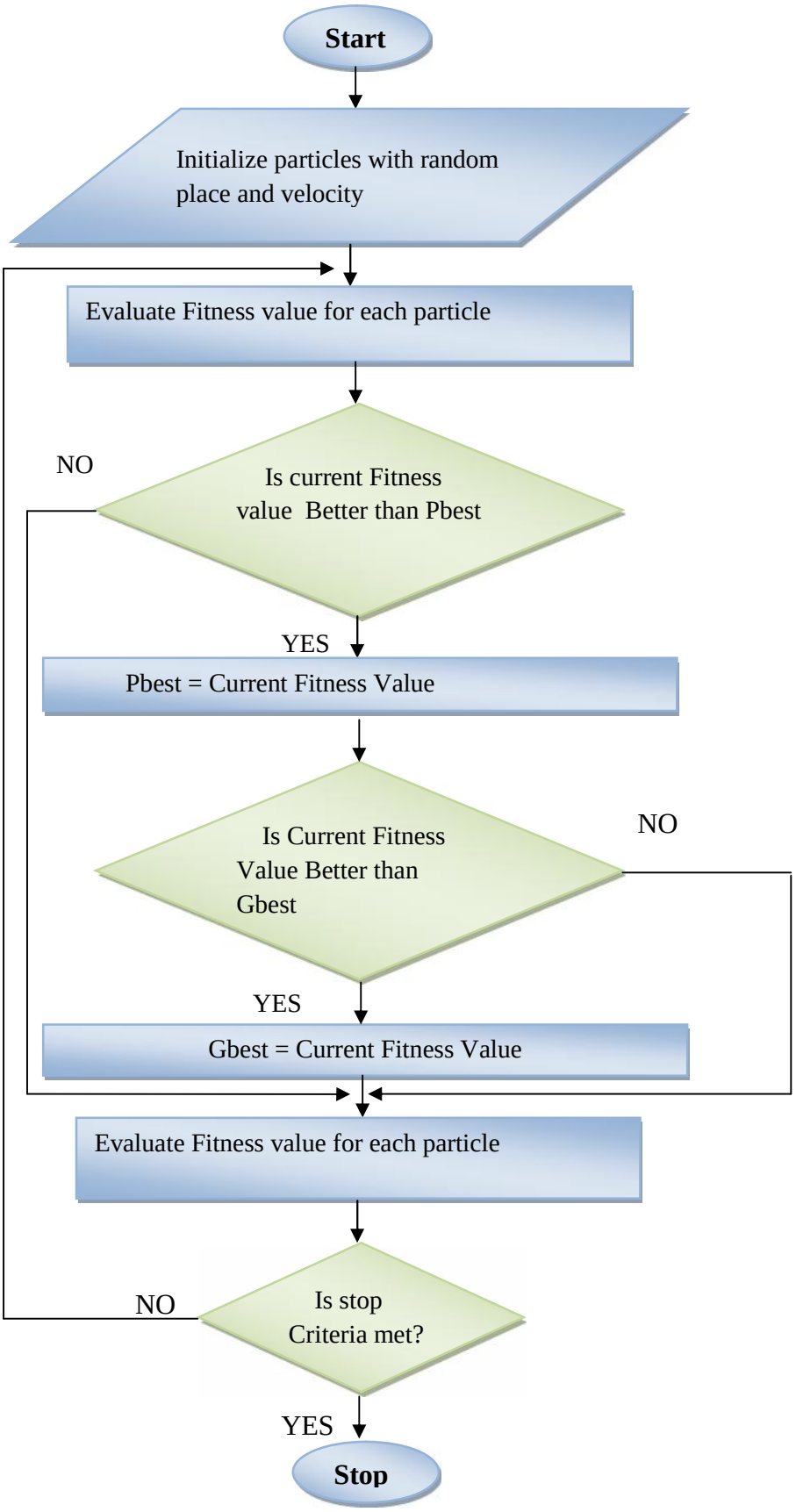

Figure 3.4: Flow chart of PSO-BASED PID tuning

3.4 Implementation of DE-PID Controller

The main objective of this section is to apply the DE algorithm in order to search an optimized configuration of PID controller parameters. A new control system optimization called DE-PID control system is proposed [30]. Figure3.5 shows the proposed controller block diagram.

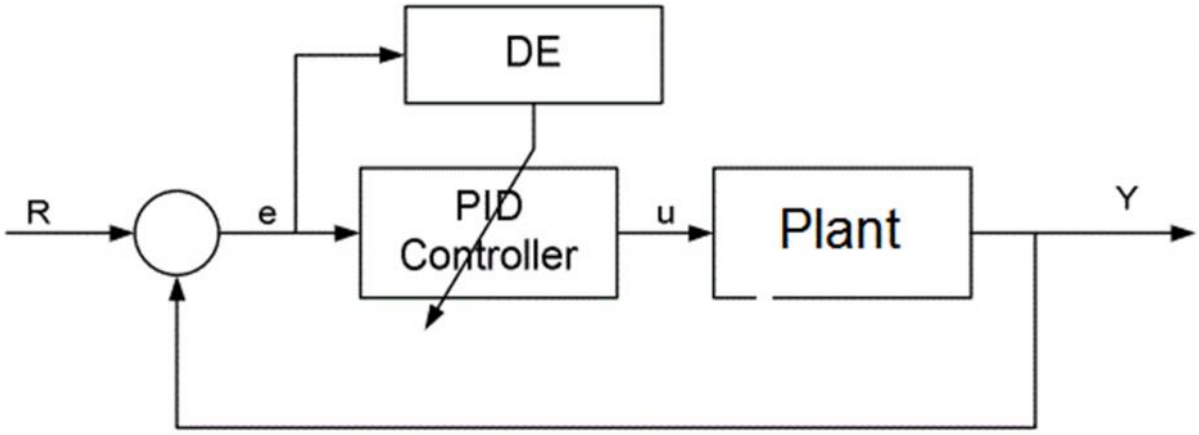

Figure 3.5: DE-PID controller block diagram [30].

The Differential Evolution (DE) algorithm will be used to search optimized PID controller parameters.

3.4.1 Optimization procedures of DE-PID [31]

The detailed Differential Evolution algorithm used in tuning the PID controller is presented below:

a) Setting DE optimization parameters

All the DE optimization parameter required for optimization process is listed below:

- **D** - Problem dimension, is set based on the number of parameters used in the objective function is presented using Eq. (3.5). In this case, problem dimension refers to the number of PID parameters Kp, Ki and Kd which is equal to 3. The needed parameters of DE algorithm are specified in chapter 2 which are:
- **NP, CR, F –** control parameters
- **G –** Number of generation or stopping condition
- **L,H –** boundary constraints are set based on the PID parameters range

b) Vector population initialization

Initialize all the vector population randomly in the given upper & lower bound and evaluate the fitness of each vector.

$$Popi,j = L + (H - L).randij\ (0,1) \quad (3.6)$$

$$Fit = f(Popj) \quad (3.7)$$

Where i=1,...,D, j=1,…,NP

$rand_{ij}$ (0,1) – random number between 0 and 1

Before the optimization starts the population needs to be initialized and their fitness values need to be evaluated. The population is initialized randomly within its boundary constraints is done using Eq. (3.6). Each of the individual in the population is used to compute the fitness value which is computed by using Eq. (3.7). Figure 3.6 shows the block diagram of population and its corresponding fitness value.

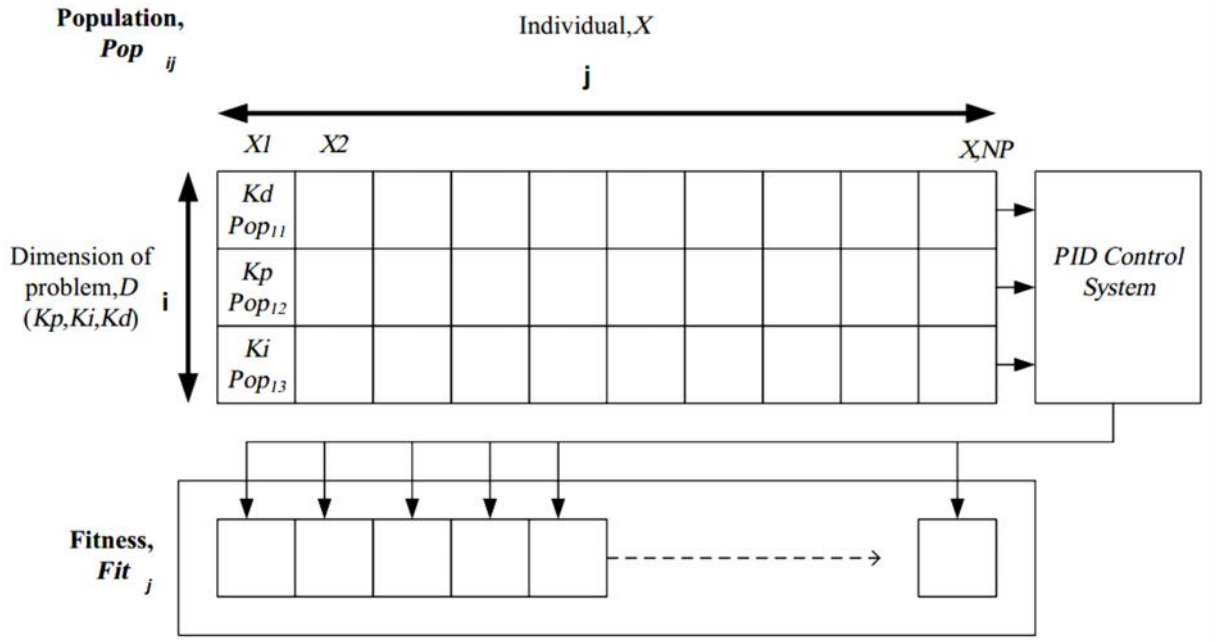

Figure 3.6: the block diagram of population and its corresponding fitness value.

c) Perform mutation & crossover

Whenever initialization process is done, now the optimization process starts. The optimization process will run iteratively until the end of generations. The first individual fitness value from the current population is set to be the target vector as shown in Figure 3.7.

Then the trial vector is created by selecting three individuals randomly from the current population, mutate using Eq. (2.5) and crossover with the target vector. The fitness value of the trial vector is computed by sending its individuals to the fitness function.

❖ **Mutant vector**

For each target vector Xj, G, a mutant vector is generated according to equation (2.5)

$$Vj, G + 1 = Xr3, G + F.(Xr1, G - Xr2, G) \quad (2.5)$$

Where the three distinct vectors Xr1, Xr2 and Xr3 randomly chosen from the current population other than target vector Xj,G. The detail example how the mutant vector is determined is shown in Figure 3.7.

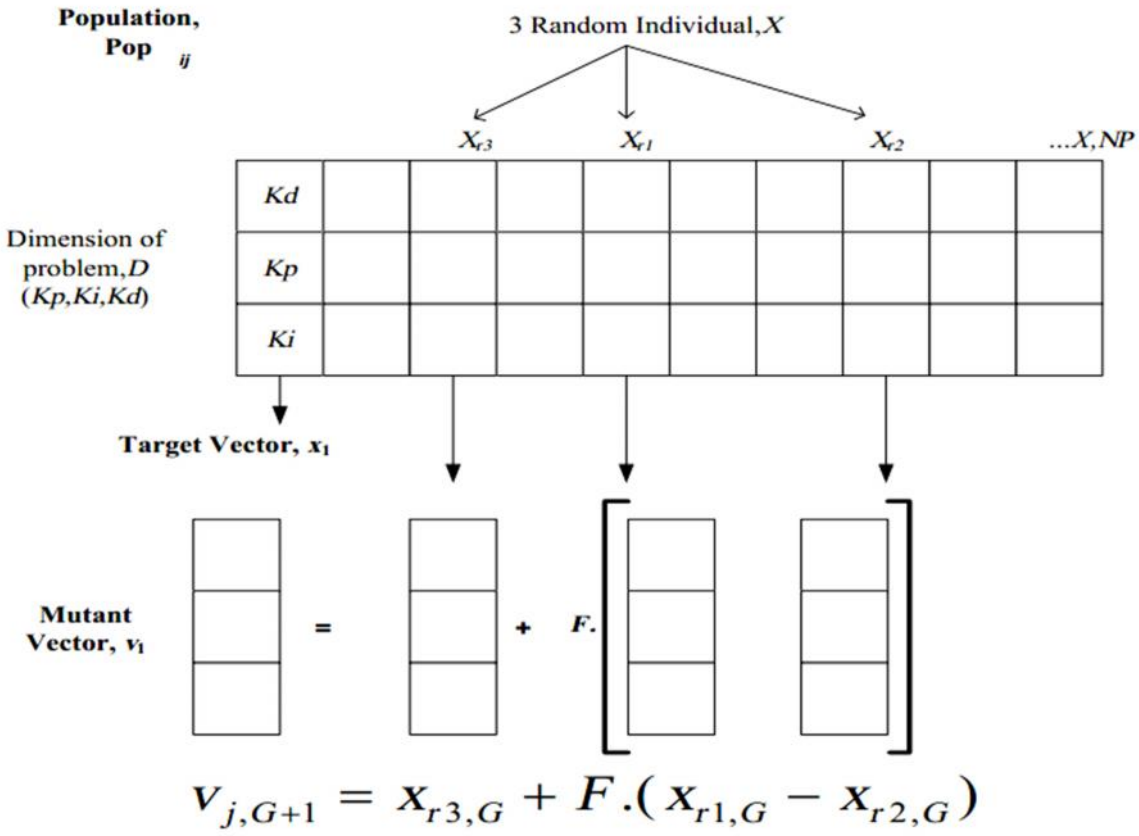

Figure 3.7.Mutation process

❖ **Crossover**

The target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector $Uj, G + 1$

$$Uij, G + 1 = \begin{cases} \text{Uij, G} + 1 & \text{if (randi} \leq \text{CR) v (Rnd = i)} \\ \text{Xj, G} & \text{otherwise} \end{cases} \quad (2.6)$$

Where $i = 1, \ldots, D$, $j = 1, \ldots, NP$

Crossover is done in order to increase the diversity of the perturbed PID parameters for each individual in the population. The block diagram on how this process is done is shown in Figure 3.8.

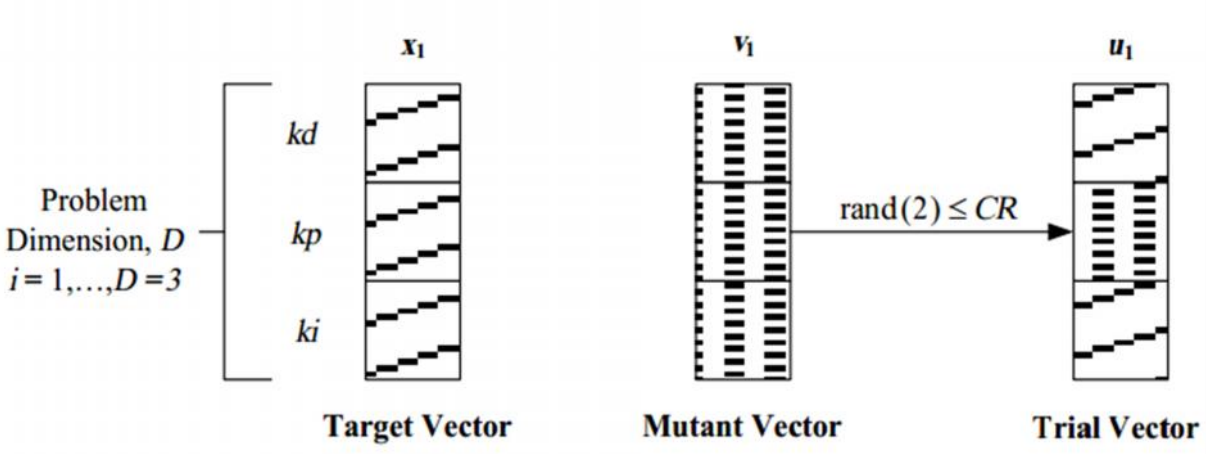

Figure 3.8: Crossover process

d) Verifying the boundary constraint

If the bound (i.e. lower & upper limit of a variable) is violated then it can be brought in the bound range (i.e. between lower & upper limit) either by forcing it to lower/upper limit (forced bound) or by randomly assigning a value in the bound range (without forcing).

$$If\ \ Xi \notin [L, H], \qquad Xi\ =\ L +\ (H - L).randi\ (0,1) \tag{3.8}$$

Eq. (3.8) purposely used in order to make sure that all the parameter vectors (PID parameters) are within its boundary constraints

.

e) Selection

Selection is performed for each target vector, Xj, G by comparing its fitness value with that of the trial vector, Uj, G and the one with the better fitness value is admitted to the next generation. Figure 3.9 shows how the selection process is performed.

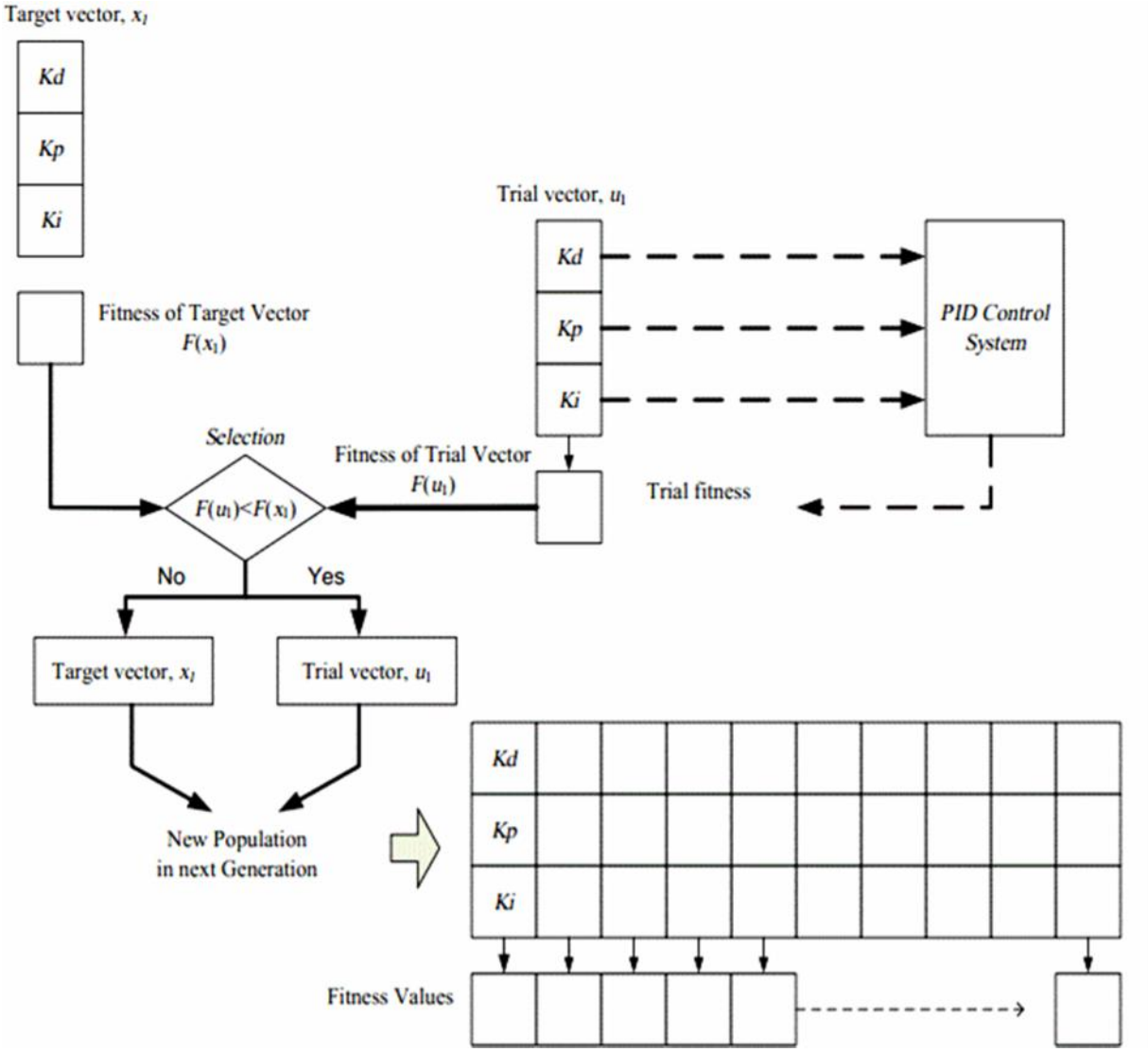

Figure 3.9: Selection process

f) Repeat step 3 to 5 until new population completed

When the first individual in the new population has produced, then the optimization process will repeat for the second individual in population as it now becomes the second target vector in the first generation, V2,1. This process will follow step 3 to 5 until new second individual in the new population is produced. This process will repeat until all individuals in the new population are updated.

g) Repeat step 6 until end of generations

The process in step 6 is repeated until the end of generation. At this stage, the optimization process is completed. The global minimum of fitness value is achieved which is referred to optimum parameter of PID controller.

Figure 3.10 shows the general flow chart of the DE algorithm tuning PID controller

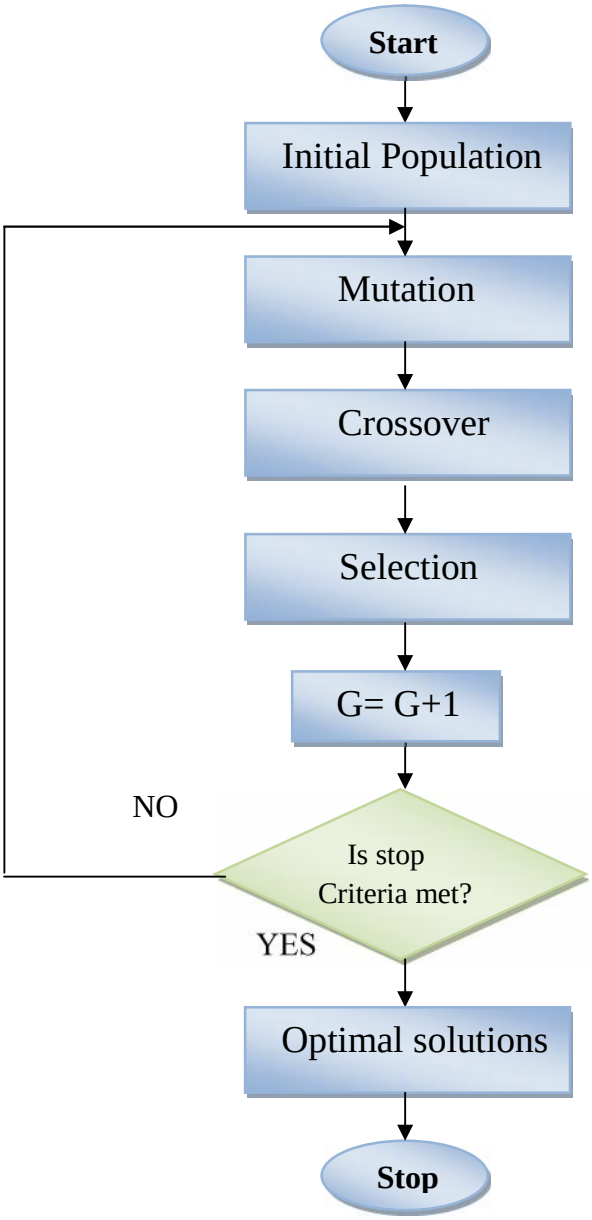

Figure 3.10: Flow chart of DE-BASED PID tuning

4.1 Introduction:

After presenting our algorithms (PSO, DE) theoretically and how they implemented to optimally tuning PID controller parameters, in this chapter, the simulation is carried out in order to study the performance between the these heuristic methods and classical Ziegler-Nichols methods for high order system given by Eq. (4.1)

- **Case study [31]**

$$G(s) = \frac{25.2s^2+21.2\,s+3}{s^5+16.58\,s^4+25.41s^3+17.18s^2+11.70s+1} \quad (4.1)$$

4.2 Simulation results without PID

The step response of the system without PID controller is shown in Figure 4.1

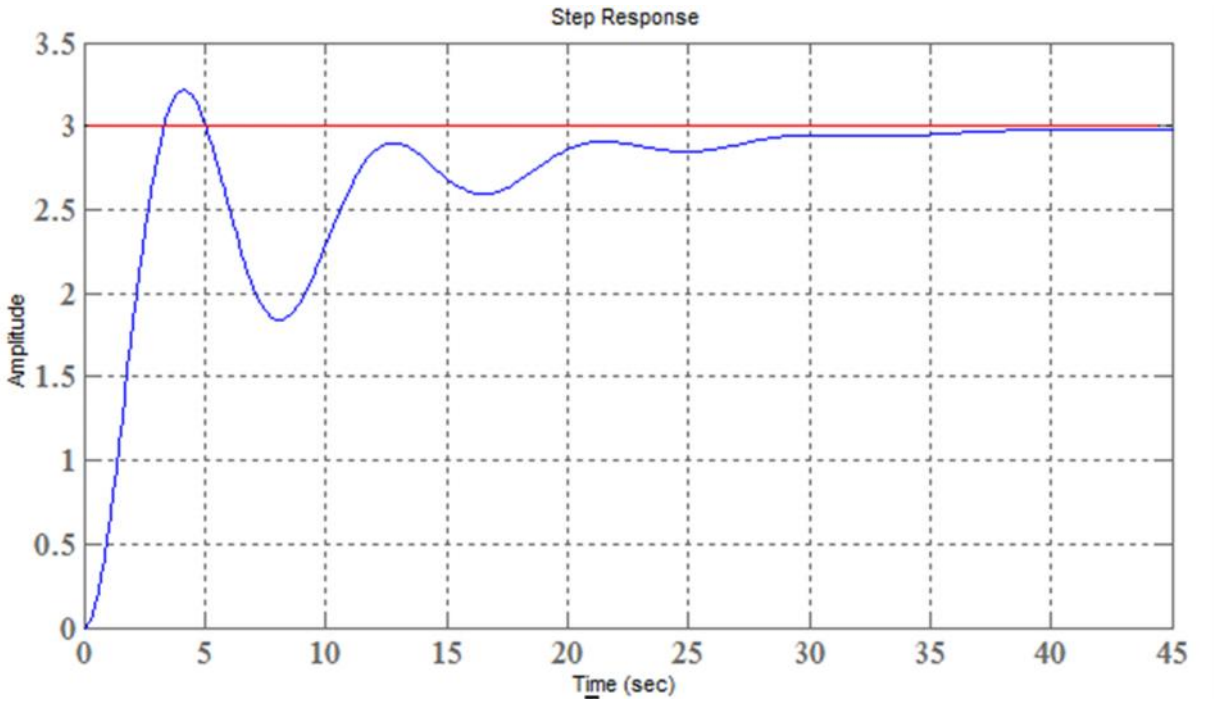

Figure 4.1: step response of the system without PID controller.

The performances of the system are:

- Rise Time (**tr**) : 2.1972 sec
- Settling Time (**ts**) : 33.513 sec
- Overshoot : 7.1023 %
- Peak value : 3.2131
- Peak Time : 4.1789 sec
- Steady state value : 3

The step response of our system is highly oscillating with a long settling time (33.5 sec) and rise time also (2.19 sec). Furthermore, the steady-state error is very large.

During this simulation, the tuning performance of PID controller is evaluated using different fitness functions, first we choose the fitness function to be IAE, ITAE, ISE, ITSE, F(k), then we choose it to be F(k)*IAE, F(K)*ITAE, F(k)*ISE, F(k)*ITSE , finally we compared the step response of each one of them . According to chapter 3 F(K) and the four performance indices are listed below:

$$F(k) = \big(1 - exp(-\beta)\big)(Mp + Ess) + \big(exp(-\beta)\big)(Ts - Tr) \quad (3.6)$$

$$IAE = \int_0^\infty |e(t)|\, dt \quad (3.2)$$

$$ITAE = \int_0^\infty t\,|e(t)|\, dt \quad (3.5)$$

$$ISE = \int_0^\infty e^2(t)dt \quad (3.3)$$

$$ITSE = \int_0^\infty t\, e^2(t)dt \quad (3.4)$$

Where k is [Kp Ki Kd], and β is the weighting factor, we can set β to be larger than 0.7 to reduce the overshoot and steady-state error. On the other hand, we can set β to be smaller than 0.7 to reduce the rise time and settling time, in our project we set it to be 0.5.

PSO and DE will heuristically find the optimum value of the controller parameters where the smaller value of objective function the fitter is the individual. Finally the transient performance of the system tuned by PSO and DE is compared with Ziegler-Nichols method.

4.3 Simulation results using DE PID controller

To start up with DE, certain parameters need to be defined. Selection of these parameters decides to a great extent the ability of global minimization. In our work we set population size, NP=50, crossover constant, CR=0.9, differentiation constant, F=0.6, and number of generation is set to be 100. Since the problem is to find the optimal PID three parameters so the dimension D is set to be 3.

Table 4.1: DE parameter selection

DE algorithm
Population size = 50
Crossover rate = 0.9
Differentiation constant = 0.6
Generation number = 100

a) Simulation results with IAE, ITAE, ISE, ITSE and F(k)as fitness function:

- **IAE**

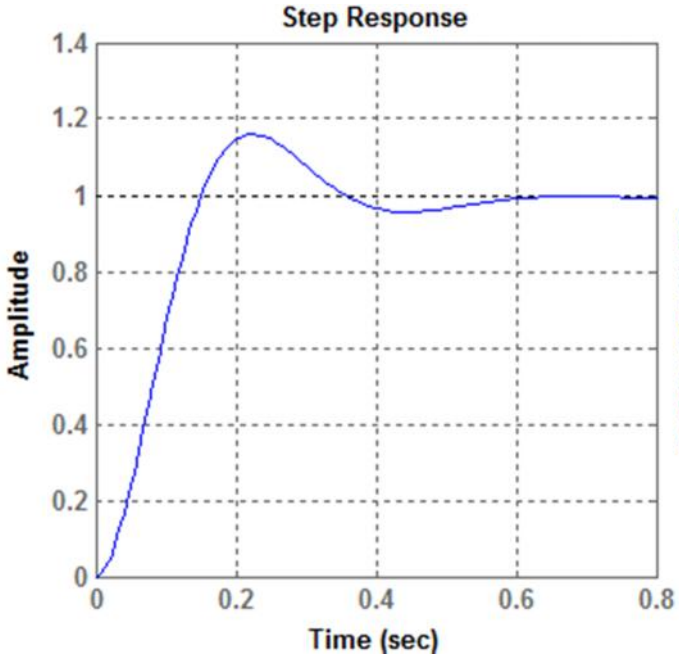

Figure 4.2: step response with Fitness function = IAE

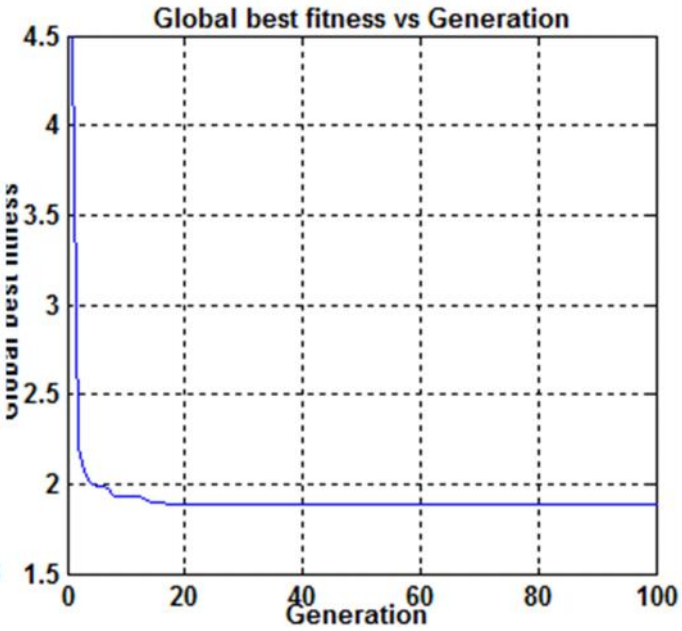

Figure 4.3: convergence tendency with Fitness function =IAE

Mp = 15.9% t_r= 0.102 s t_s= 0.552s **Best Fitness** = 1.885

- **ITAE**

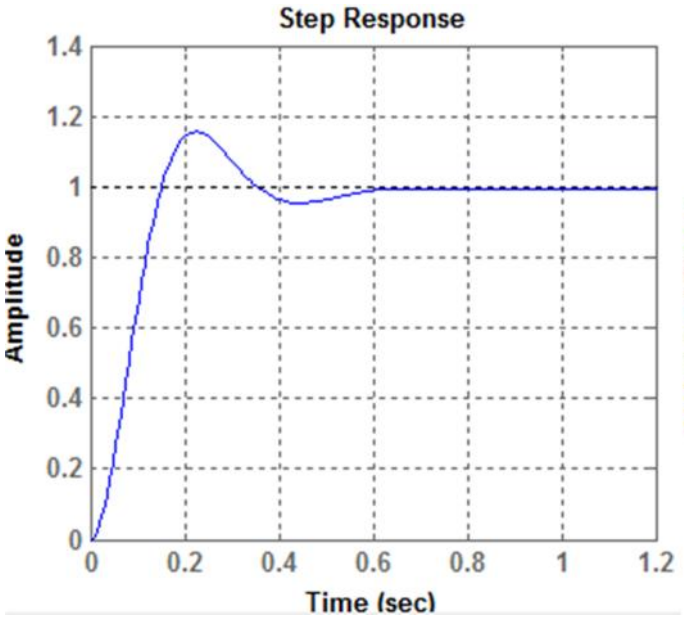

Figure 4.4: step response with Fitness function = ITAE

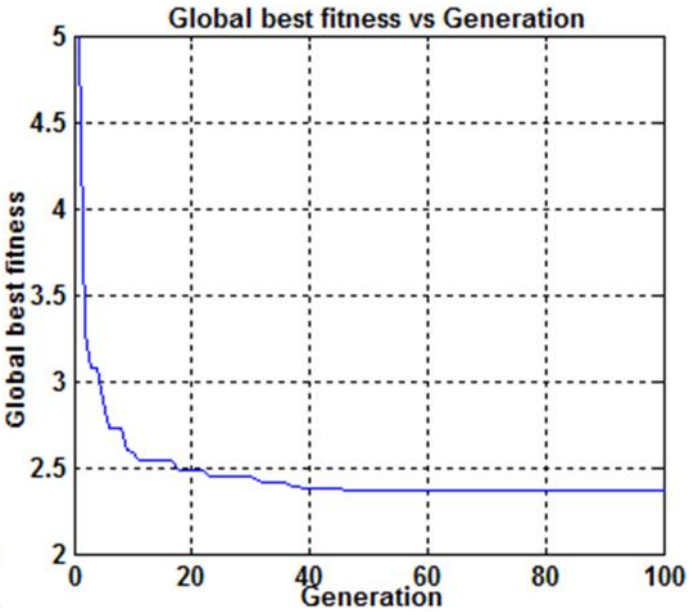

Figure 4.5: convergence tendency with Fitness function =ITAE

Mp = 15.7% t_r= 0.102 s t_s= 0.557s **Best Fitness** = 2.36

- **ISE**

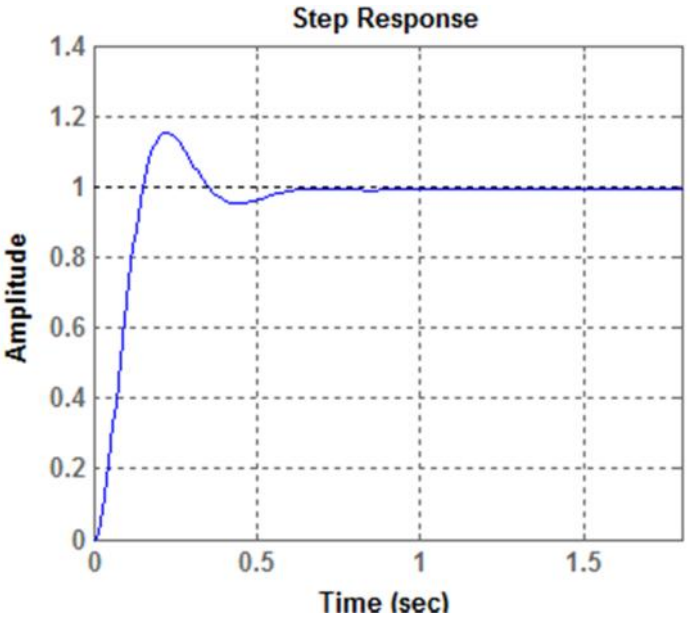

Figure 4.6: step response with Fitness function = ISE

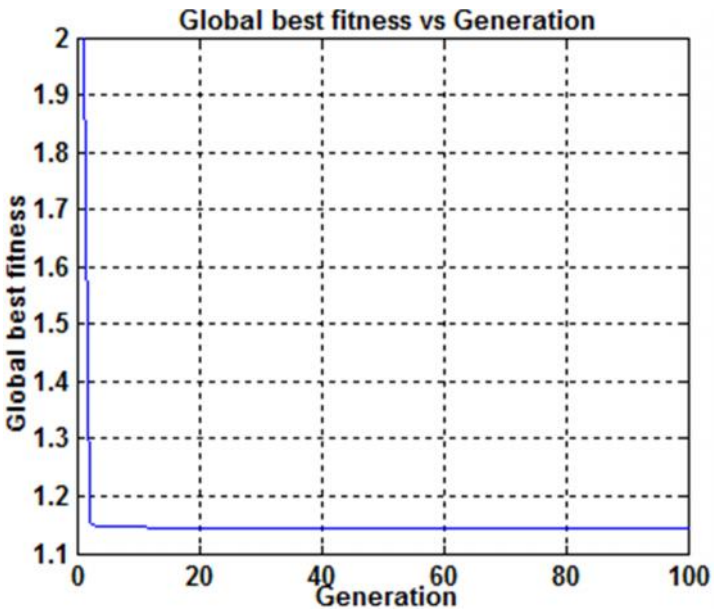

Figure 4.7: convergence tendency with Fitness function =ISE

Mp = 15% t_r= 0.11 s t_s= 0.57s **Best Fitness** = 1.14

- **ITSE**

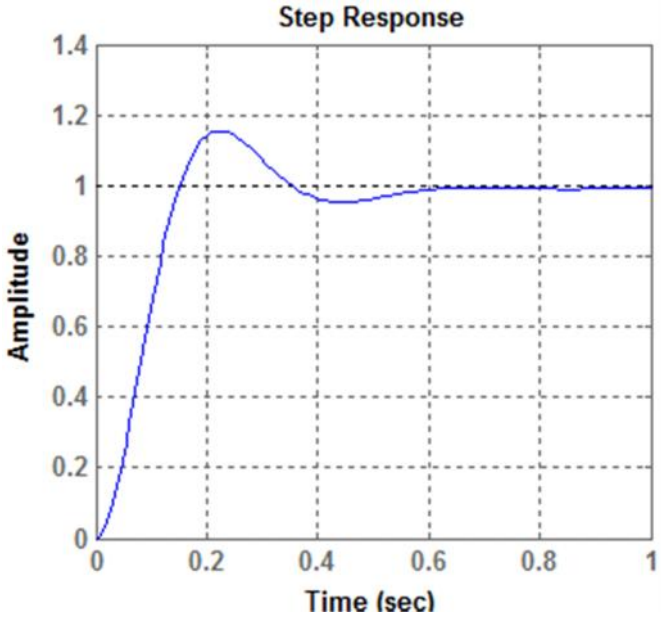

Figure 4.8: step response with Fitness function = ITSE

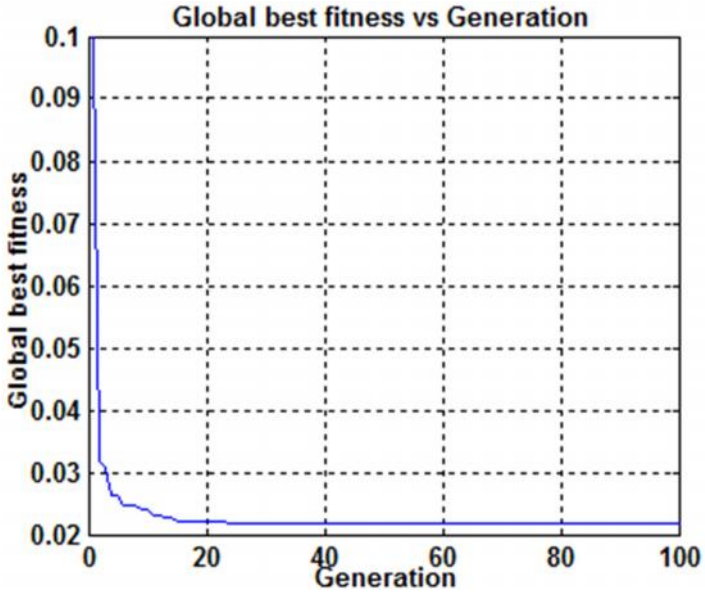

Figure 4.9: convergence tendency with Fitness function =ITSE

Mp = 15.5% t_r= 0.103 s t_s= 0.56s **Best Fitness** = 0.0219

➢ **F(k)**

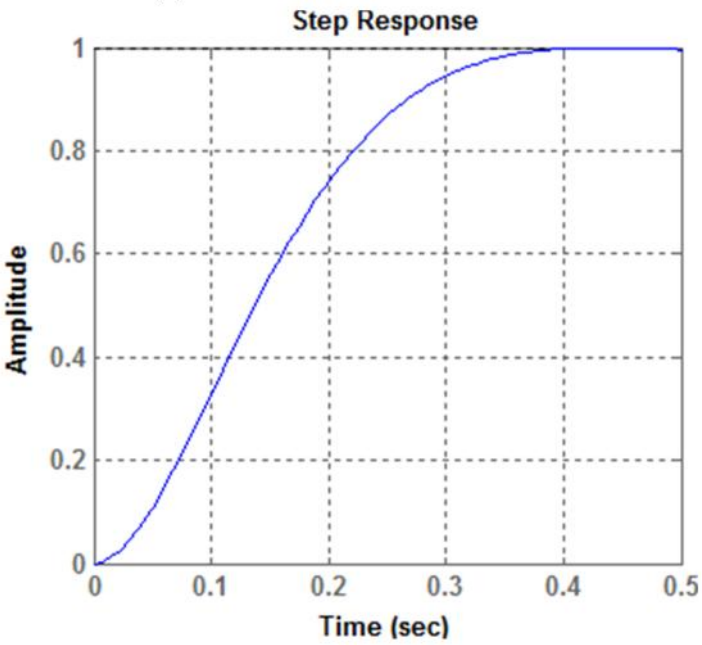

Figure 4.10: step response with Fitness function = F(k)

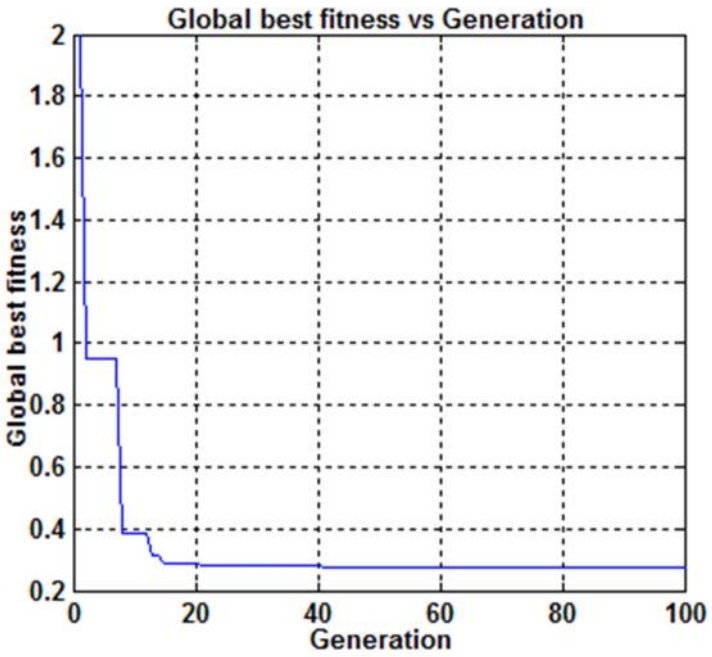

Figure 4.11: convergence tendency with Fitness function =F(k)

Mp = 0% t_r= 0.22 s t_s= 0.343s **Best Fitness** = 0.2742

b) Simulation results with F(k)*IAE ,F(k)*ITAE, F(k)*ISE and F(k)*ITSE as Fitness Functions

➢ **F(k)*IAE**

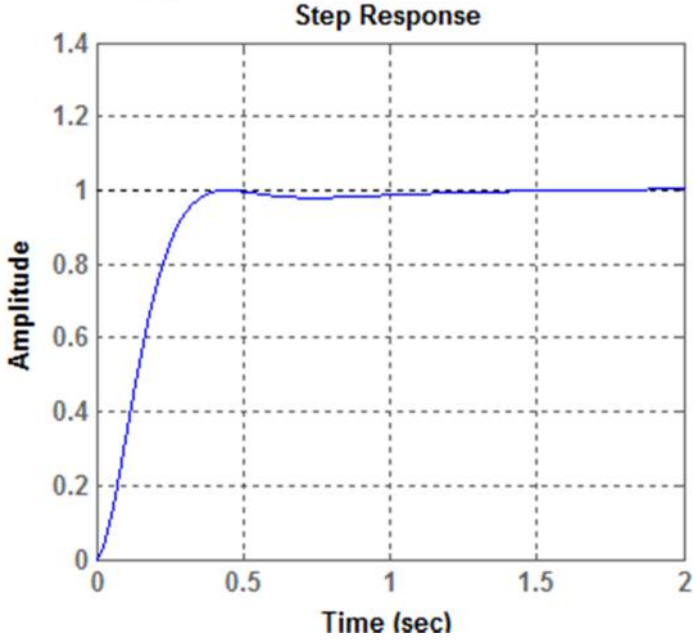

Figure 4.12: step response with Fitness function = F(k)*IAE

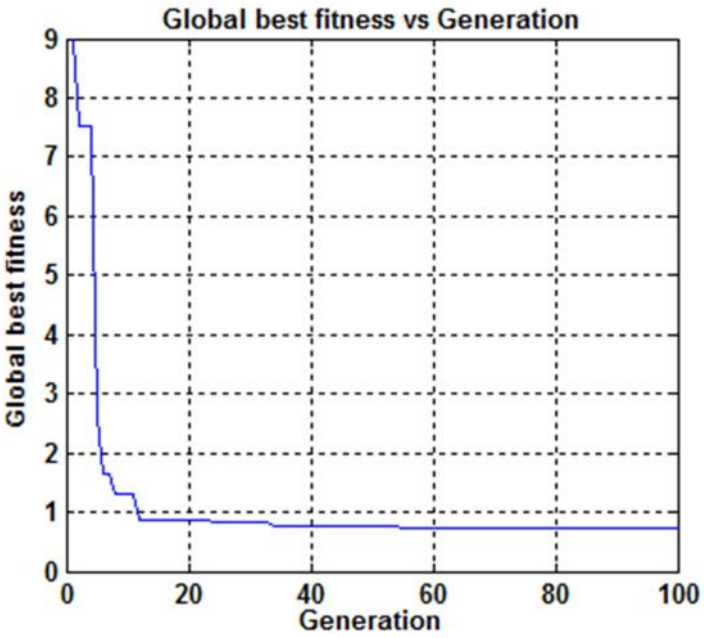

Figure 4.13: convergence tendency with Fitness function=F(k)*IAE

Mp = 0.361% t_r= 0.23 s t_s= 0.78s **Best Fitness** = 0.7443

- **F(k)*ITAE**

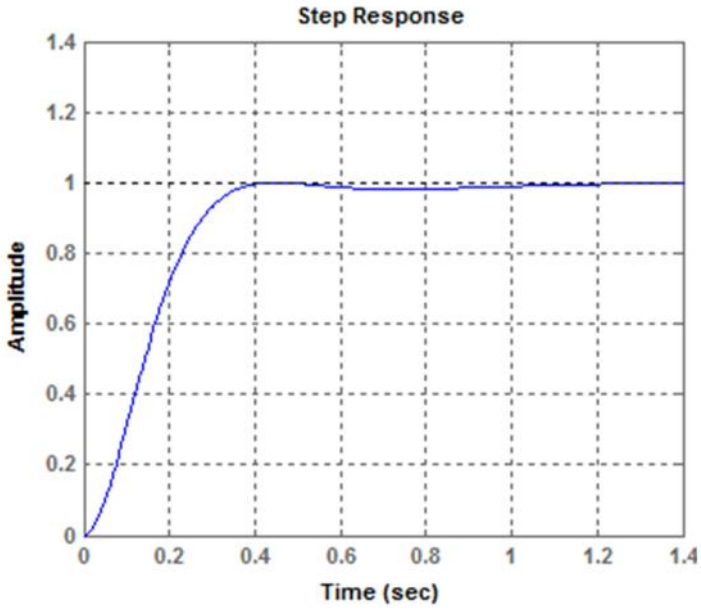

Figure 4.14: step response with Fitness function = F(k)*ITAE

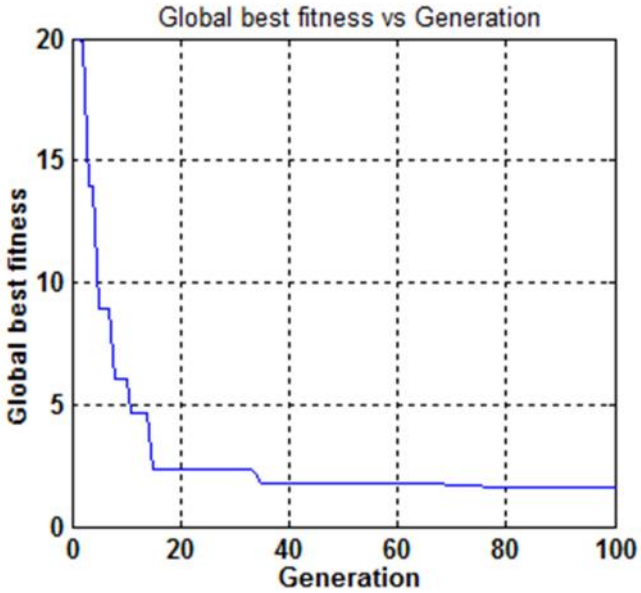

Figure 4.15: convergence tendency with Fitness function =F(k)*ITAE

Mp = 0 % t_r= 0.239 s t_s= 0.356s **Best Fitness** = 1.58

- **F(k)*ISE**

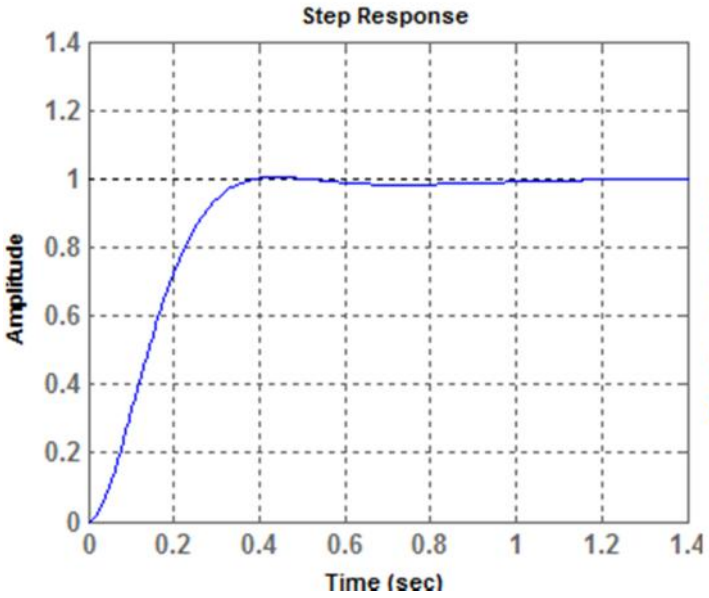

Figure 4.16: step response with Fitness function = F(k)*ISE

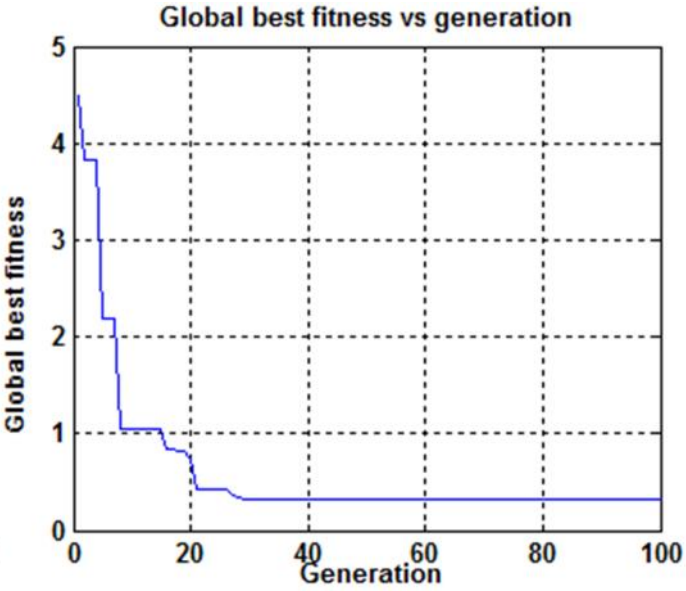

Figure 4.17: convergence tendency with Fitness function =F(k)*ISE

Mp = 0.44 % t_r= 0.224 s t_s= 0.344s **Best Fitness** = 0.3176

- **F(k)*ITSE**

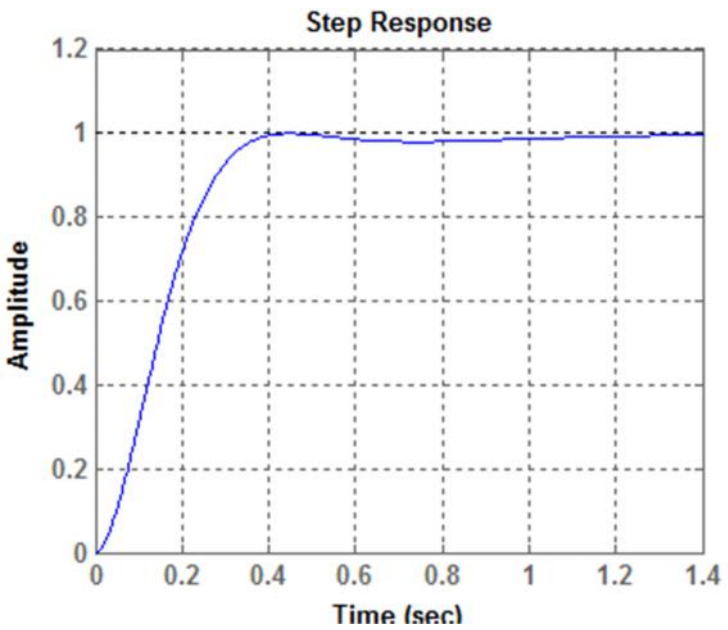

Figure 4.18: step response with Fitness function = F(k)*ITSE

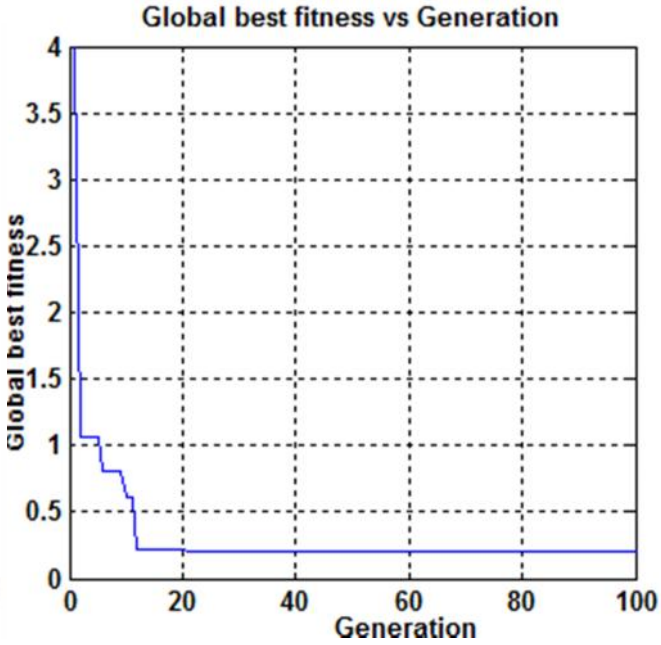

Figure 4.19: convergence tendency with Fitness function =F(k)*ITSE

Mp = 0 % $\mathbf{t_r}$= 0.23 s $\mathbf{t_s}$= 0.78s **Best Fitness** = 0.2056

The best obtained results of closed-loop step response for DE PID tuning method are summarized in Table 4.2

Table 4.2: Step response performance for DE PID controllers.

Fitness function	Rise time (s)	Settling time (s)	Over shoo (%)	Kp	Ki	Kd	Best fitness
IAE	0.102	0.552	15.9	4.07	6.23	10	1.885
ITAE	0.102	0.557	15.6	3.71	6.32	10	2.36
ISE	0.11	0.57	15	3.45	6.30	10	1.14
ITSE	0.103	0.56	15.5	3.60	6.15	10	0.0219
F(k)	0.22	0.34	0	1.08	5.51	4.50	0.2742
F(k)*IAE	0.23	0.78	0.361	1.83	2.69	4.21	0.7443
F(k)*ITAE	0.239	0.356	0	1.81	2.83	4.23	1.58
F(k)*ISE	0.224	0.344	0.44	1.90	2.88	4.30	0.3176
F(k)*ITSE	0.23	0.78	0	1.83	2.68	4.20	0.2056

4.4 Simulation results using PSO PID controller:

In our work, in order to acquire better performance and fast convergence of the PSO algorithms, parameters which are used in these algorithms have been initialized according to Table 4.3

Table 4.3: PSO parameters selection.

PSO parameter
Population size = 100
Acceleration constants,C1 =C2= 2
Inertia weight factor :Wmin= 0.4,Wmax=0.9
Iteration number = 100

a) Simulation results with IAE, ITAE, ISE, ITSE and F(k)as fitness function:

- **IAE**

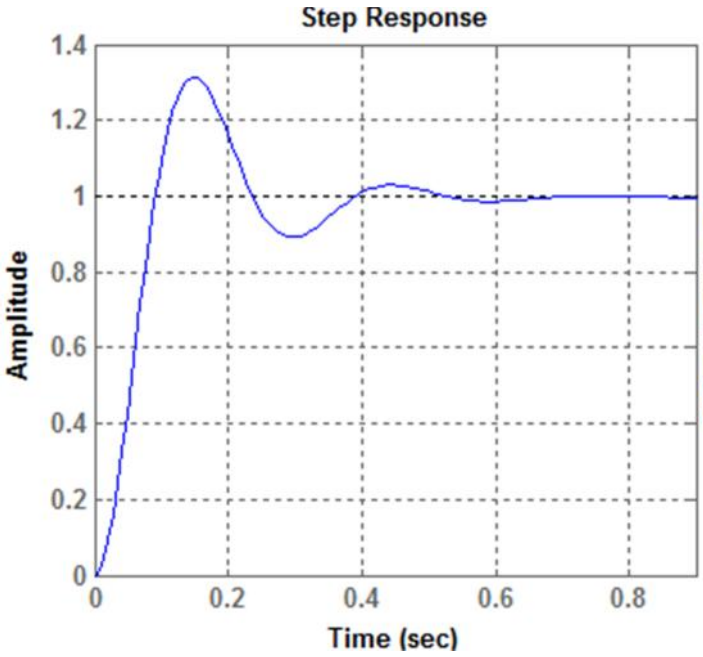

Figure 4.20: step response with Fitness function = IAE

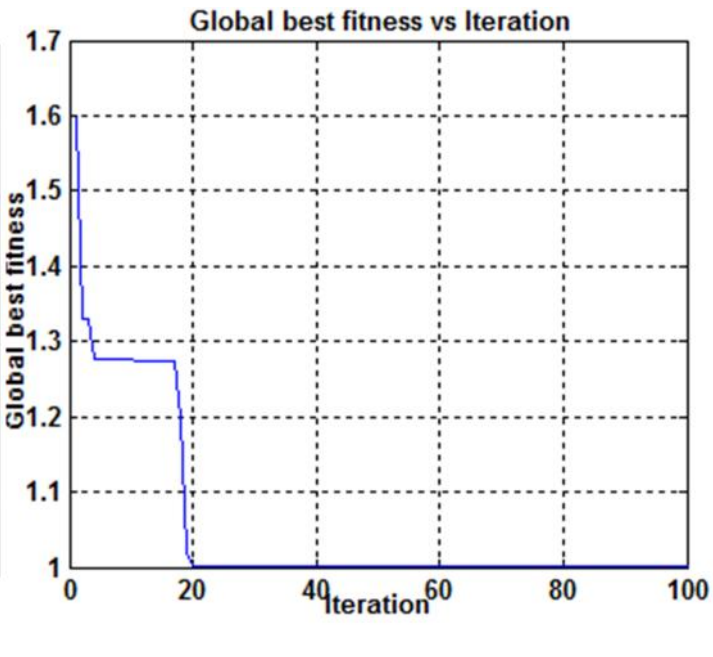

Figure 4.21: convergence tendency with Fitness function =IAE

Mp = 30 % t_r= 0.06 s t_s= 0.48s **Best Fitness** = 1.00

- **ITAE**

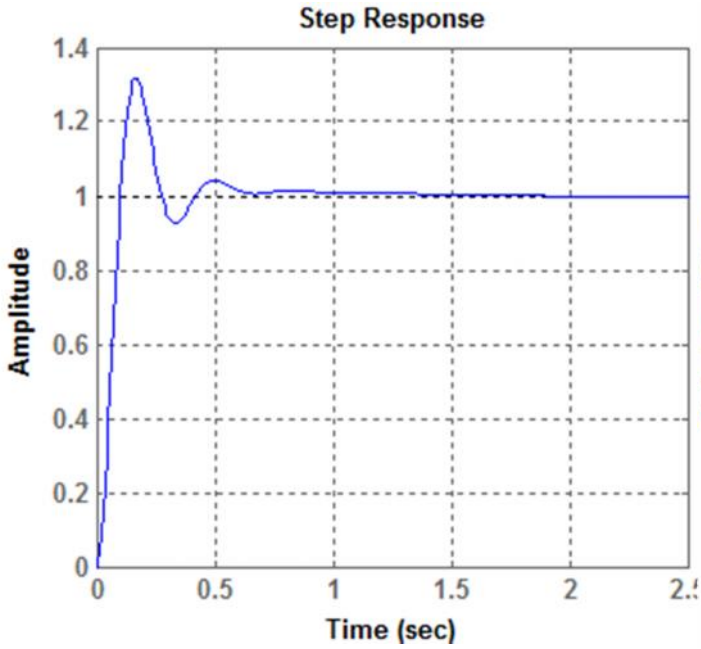

Figure 4.22: step response with Fitness function = ITAE

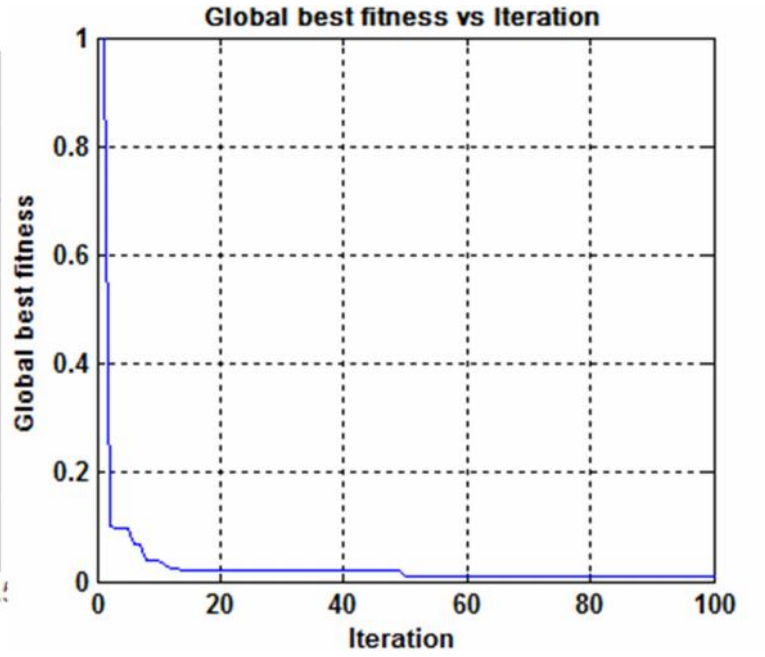

Figure 4.23: convergence tendency with Fitness function =ITAE

Mp = 31.9 % **t_r**= 0.068 s **t_s**= 0.58s **Best Fitness** = 0.01

- **ISE**

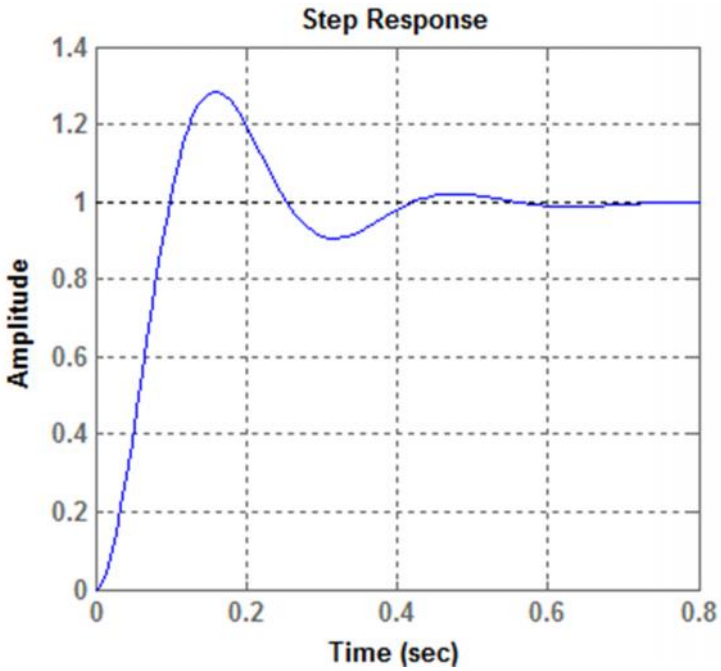

Figure 4.24: step response with Fitness function = ISE

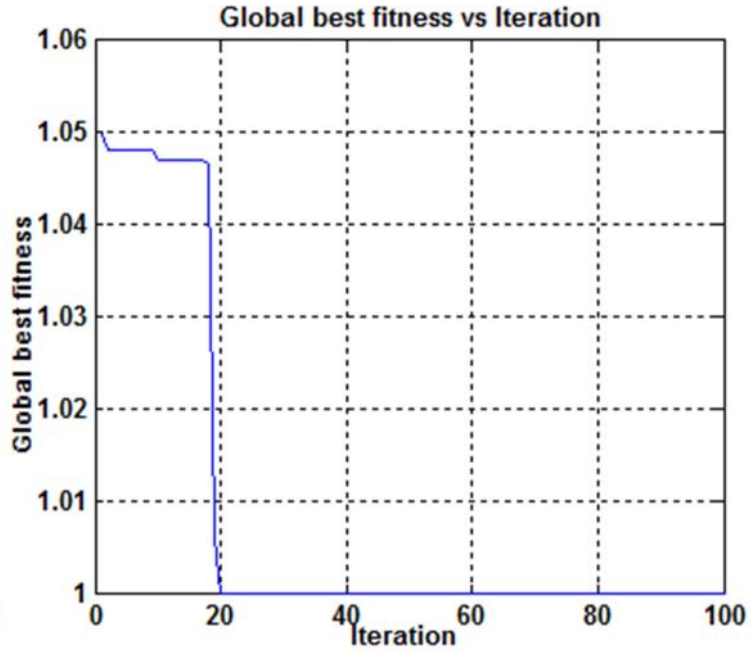

Figure 4.25: convergence tendency with Fitness function =ISE

Mp = 29 % **t_r**= 0.062 s **t_s**= 0.46s **Best Fitness** = 1.00

- **ITSE**

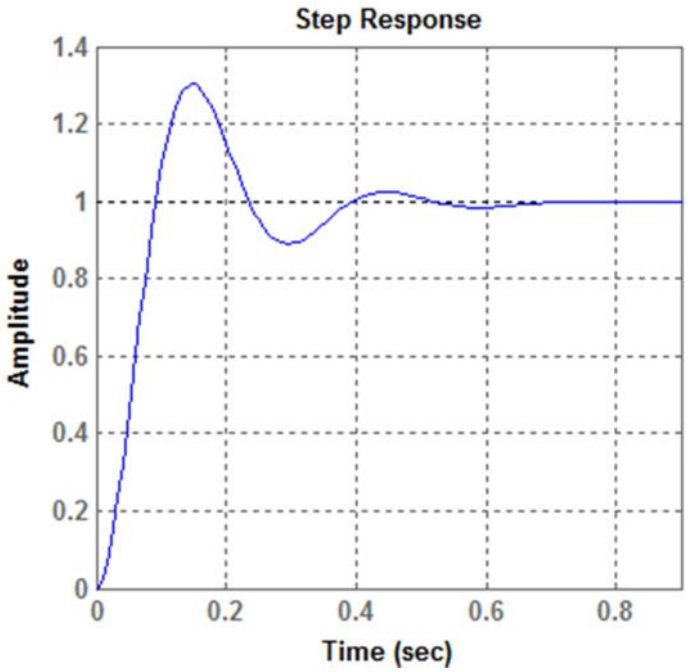

Figure 4.26: step response with Fitness function = ITSE

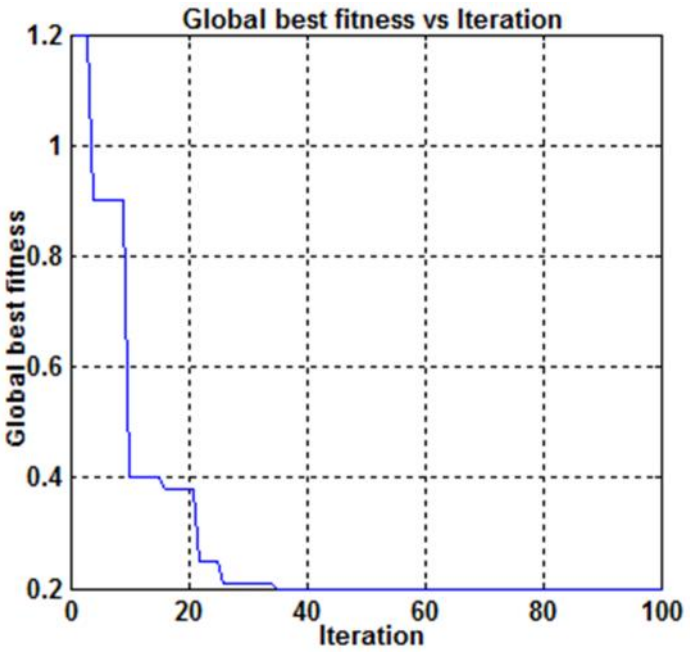

Figure 4.27: convergence tendency with Fitness function =ITSE

Mp = 30 % t_r= 0.061 s t_s= 0.46s **Best Fitness** = 0.2

- **F(k)**

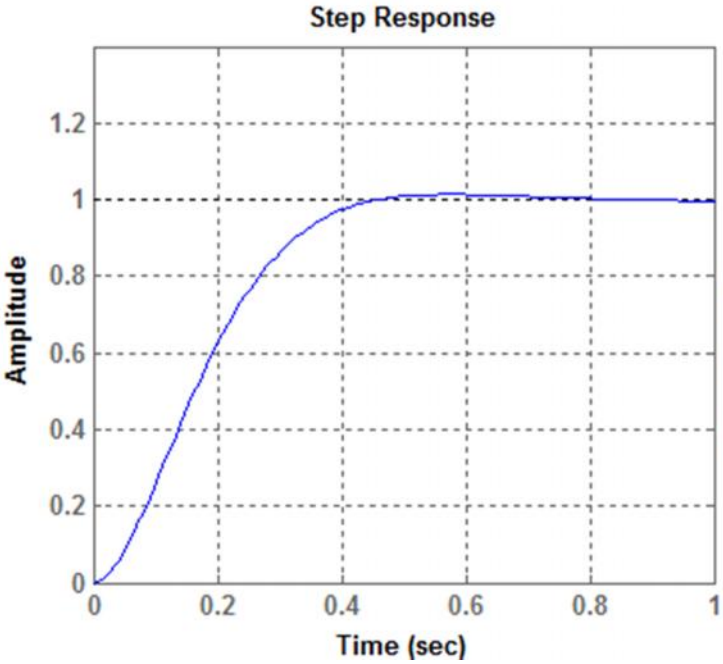

Figure 4.28: step response with Fitness function = F(k)

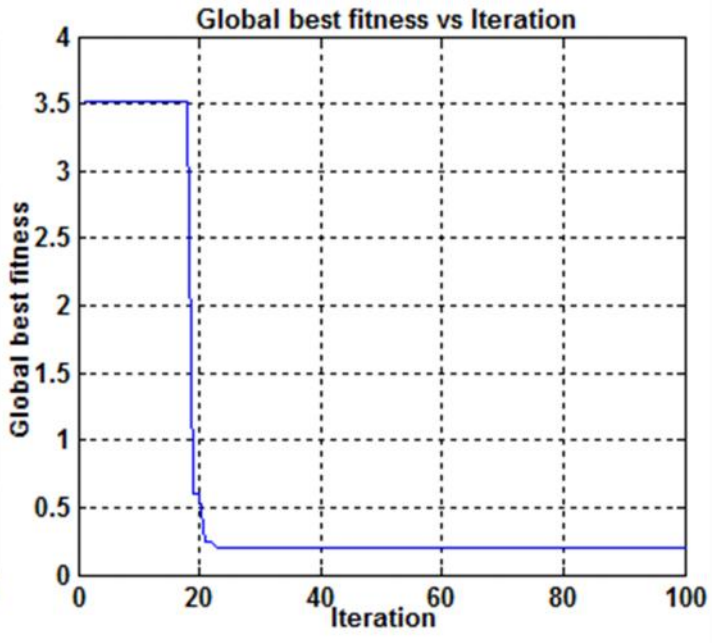

Figure 4.29: convergence tendency with Fitness function =F(k)

Mp = 1.3 % t_r= 0.27 s t_s= 0.41s **Best Fitness** = 0.1967

b) Second simulation results with F(k)*IAE ,F(k)*ITAE, F(k)*ISE and F(k)*ITSE

- **F(k)*IAE**

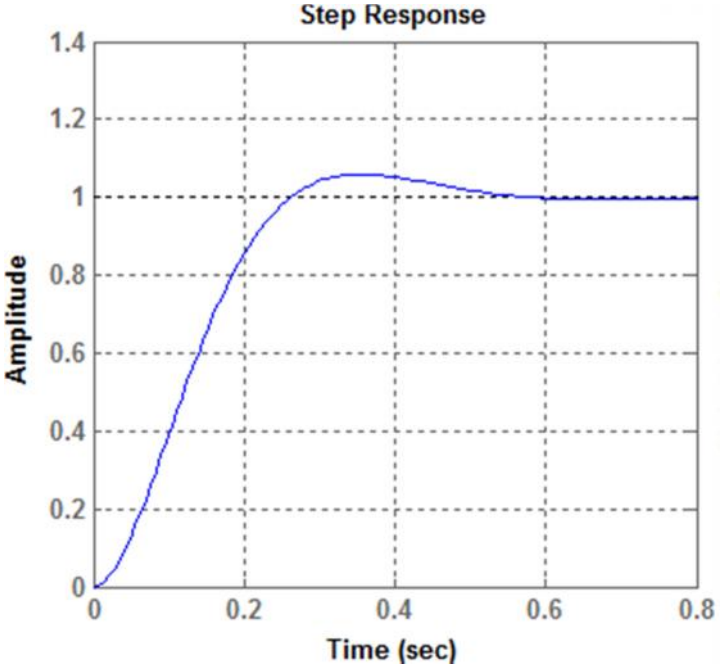

Figure 4.30: step response with Fitness function =F(k)*IAE

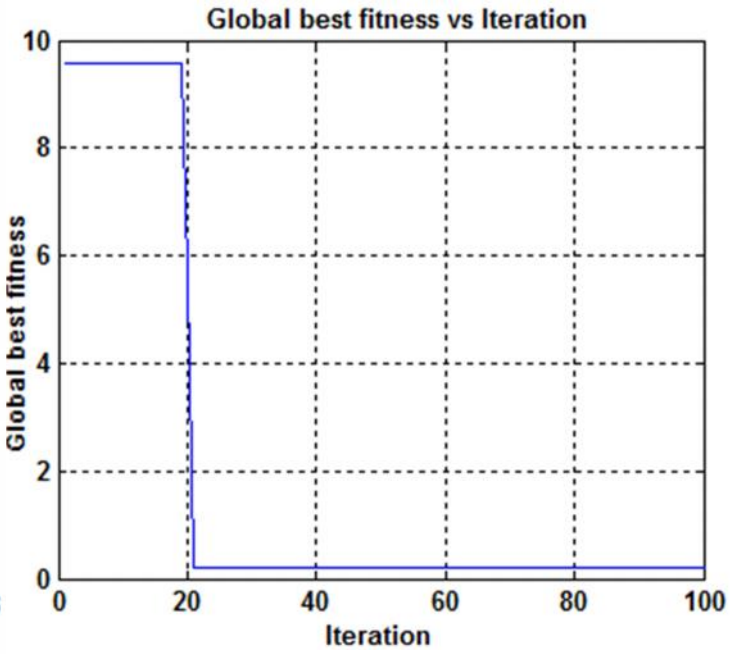

Figure 4.31: convergence tendency with Fitness function =F(k)*IAE

Mp = 5 % t_r= 0.17 s t_s= 0.49s **Best Fitness** = 0.1967

- **F(k)*ITAE**

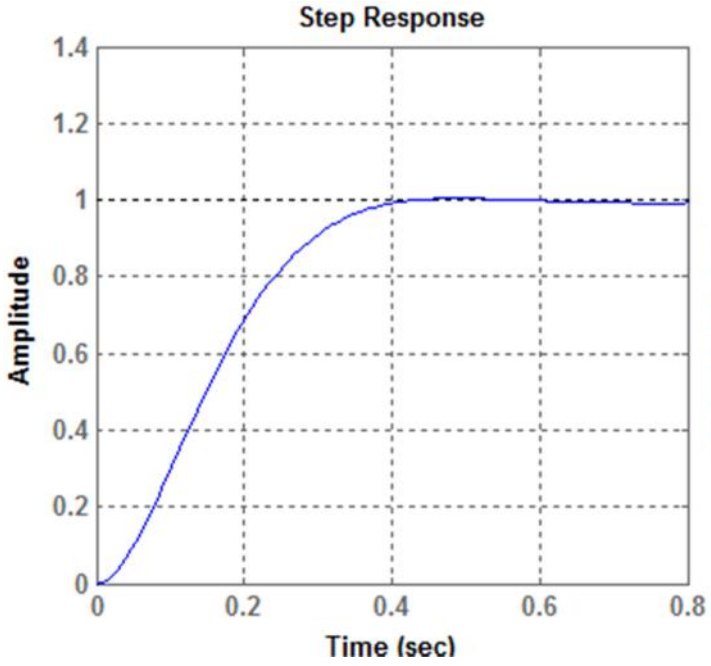

Figure 4.32: step response with Fitness function = F(k)*ITAE

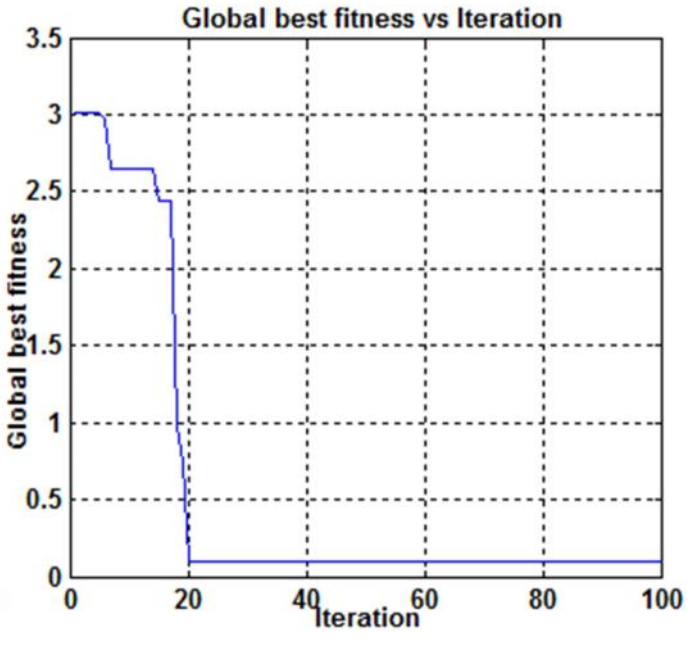

Figure 4.33: convergence tendency with Fitness function =F(k)*ITAE

Mp = 0.41 % t_r= 0.24 s t_s= 0.37s **Best Fitness** = 0.098

- **F(k)*ISE**

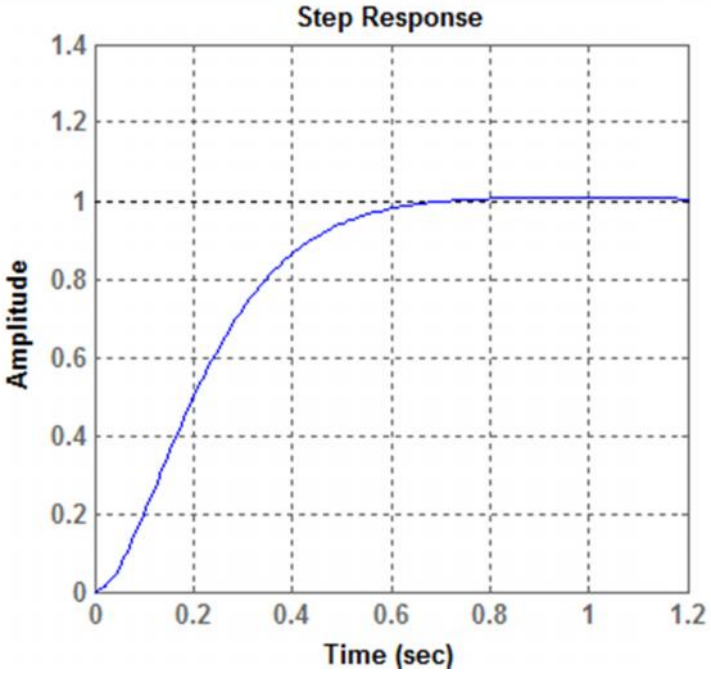

Figure 4.34: step response with Fitness function = F(k)*ISE

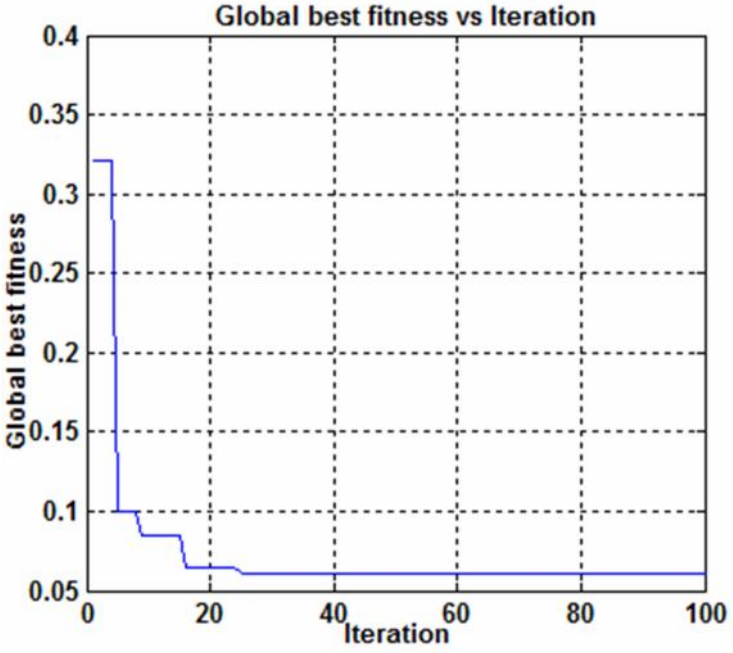

Figure 4.35: convergence tendency with Fitness function =F(k)*ISE

Mp = 0.7 % t_r= 0.37 s t_s= 0.60s **Best Fitness** = 0.061

- **F(k)*ITSE**

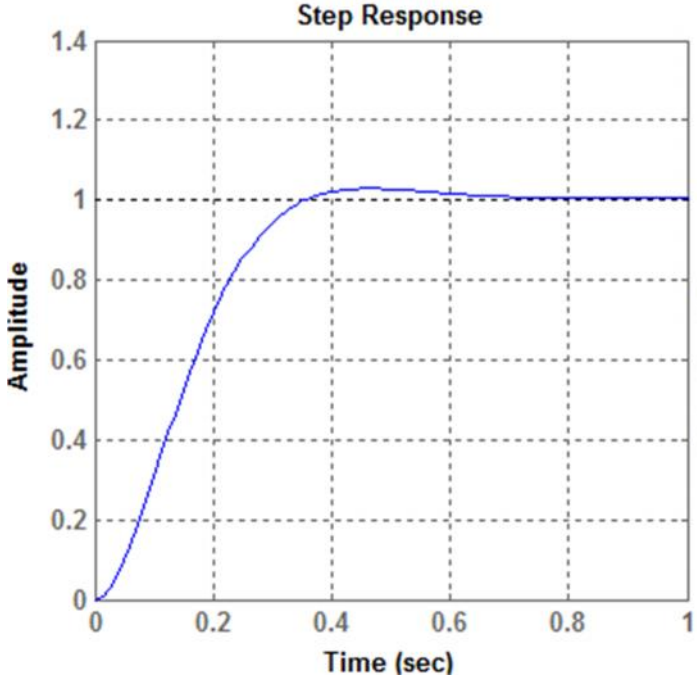

Figure 4.36: step response with Fitness function = F(k)*ITSE

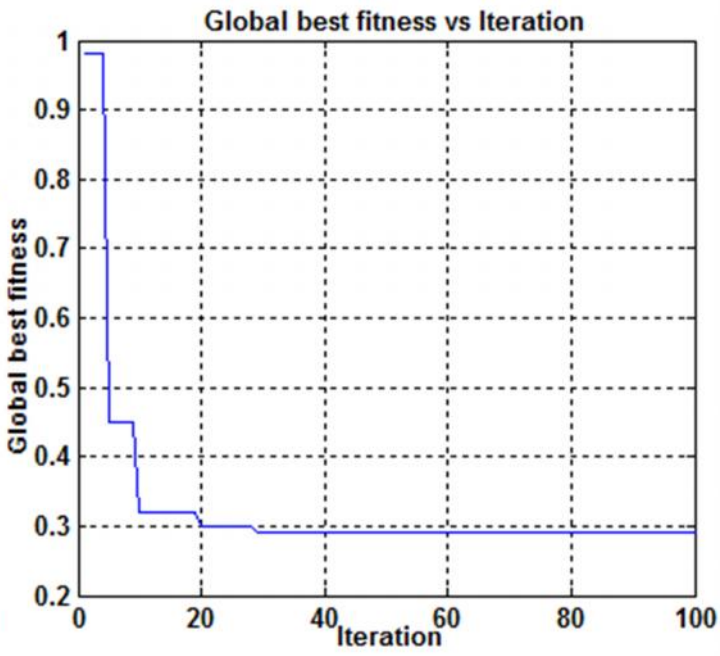

Figure 4.37: convergence tendency with Fitness function =F(k)*ITSE

Mp = 2 % t_r= 0.22 s t_s= 0.57s **Best Fitness** = 0.29

The best results of closed-loop step response for PSO PID tuning method are summarized in Table 4.4

Table 4.4: Step response performance for PSO PID controllers

Fitness function	Rise time (s)	Settling time (s)	Over shoo (%)	Kp	Ki	Kd	Best fitness
IAE	0.06	0.48	30	9.84	11.60	20	1.00
ITAE	0.068	0.582	31.9	19.8	16.3	16.48	0.01
ISE	0.06	0.46	29	3.51	18.9	19.6	1.00
ITSE	0.061	0.46	30	5.81	14.40	20	0.2
F(k)	0.27	0.41	1.3	2.36	1.44	3.40	0.1967
F(k)*IAE	0.17	0.49	5	3.36	3.33	5.37	0.1957
F(k)*ITAE	0.24	0.37	0.41	2.05	2.71	3.95	0.098
F(k)*ISE	0.37	0.60	0.7	2.02	1.29	2.55	0.061
F(k)*ITSE	0.22	0.57	2	2.83	2.62	4.11	0.29

4.5 Discussion

The convergence tendency graphs depict the changes of the value of the best fitness evaluated with respect to iterations. It can be seen that these graphs start by a sudden drop, which explain that the first positions that is randomly generated to each individual are bad and far from the best fitness value but after the execution of the first step of PSO or DE these latter generate a much better fitness value. After performing a certain number of iterations, the fitness continue converging but slowly until it reaches a low point, and remains steady for the rest of the iterations, this point indicates the global best fitness value.

The results for closed-loop step response for DE PID tuning method gave us better performance, the responses of the first set of fitness functions (IAE,ITAE,ISE,ITSE) are almost indistinguishable, they gave us a good transient performance rise time **tr** = 0.102s and settling time **ts** = 0.557s with small overshoot **MP**= 15%. Furthermore, the steady-state error (**Ess**) is now equal to zero. For the second set of fitness functions (F(k), F(k)*IAE, F(k)*ITAE, F(k)*ISE, F(k)*ITSE), it gives us good response compared to the first set when considering the overshoot , in this set the obtained overshoot **MP** vary between 0% and 0.04%.

The obtained step response for PSO PID tuning method also gave us a good performance, for the first set of fitness functions (IAE,ITAE,ISE,ITSE), the value of rise time **tr** can vary from 0.060s to 0.068s and settling time **ts** vary between 0.46s and 0.582s with little high overshoot **MP** about 30%. For the second set of fitness functions (F(k), F(k)*IAE, F(k)*ITAE, F(k)*ISE,

F(k)*ITSE) , it reduces the overshoot to vary between 0.41s to 5% and the values of rise time and settling time are still approximately the same. However the choice of the fitness function is highly relevant, we have seen that the second set of fitness functions performed by multiplying each performance index of the first set by F(k) gives better results, its convergence tendency graphs show more trends which explain that these fitness functions helps to provide thorough exploration of the search space.

4.6 PID controller parameters tuned by Ziegler-Nichols method

In this part, we have implemented the PID controller using classical Ziegler-Nichols closed-loop method. From the root locus plot we determined the ultimate gain ku = 6.66, and its ultimate period of oscillation *t*u = 1.892s, from these values and according to table 4.5 we have calculated the best parameters KP = 3.99, KI = 4.33 and KD =0.91.

Table 4.5: Ziegler-Nichols PID Tuning Values

Controller	KP	Ti	Td
P	0.5KPU	∞	0
PI	0.45KPU	$\frac{P_u}{1,2}$	0
PID	0.6KPU	$\frac{P_u}{2}$	$\frac{P_u}{8}$

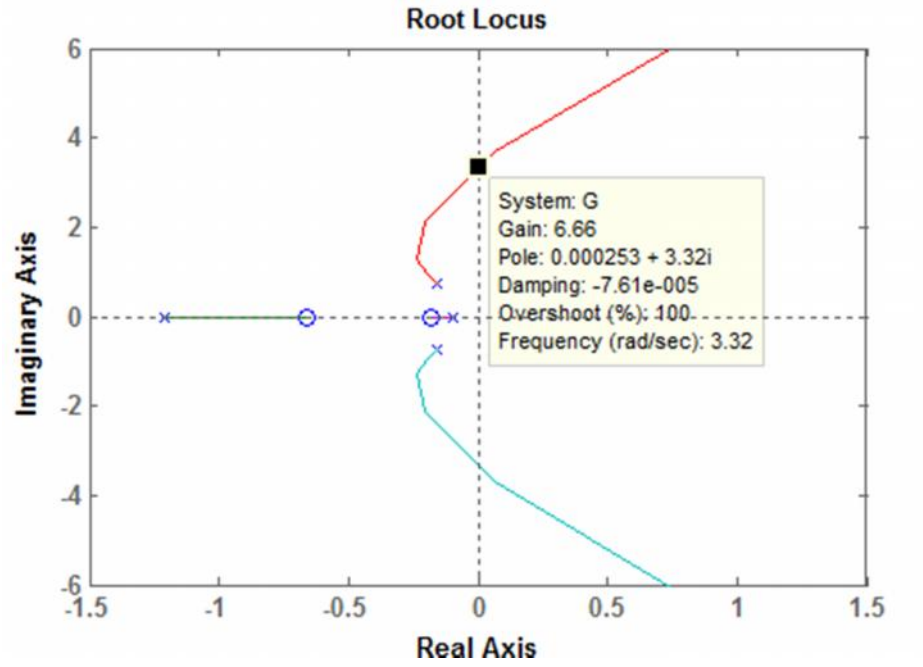

Figure 4.38: Root locus plot for the system.

The step response of our system tuned using Z-N closed-loop method is shown in Figure 4.39.

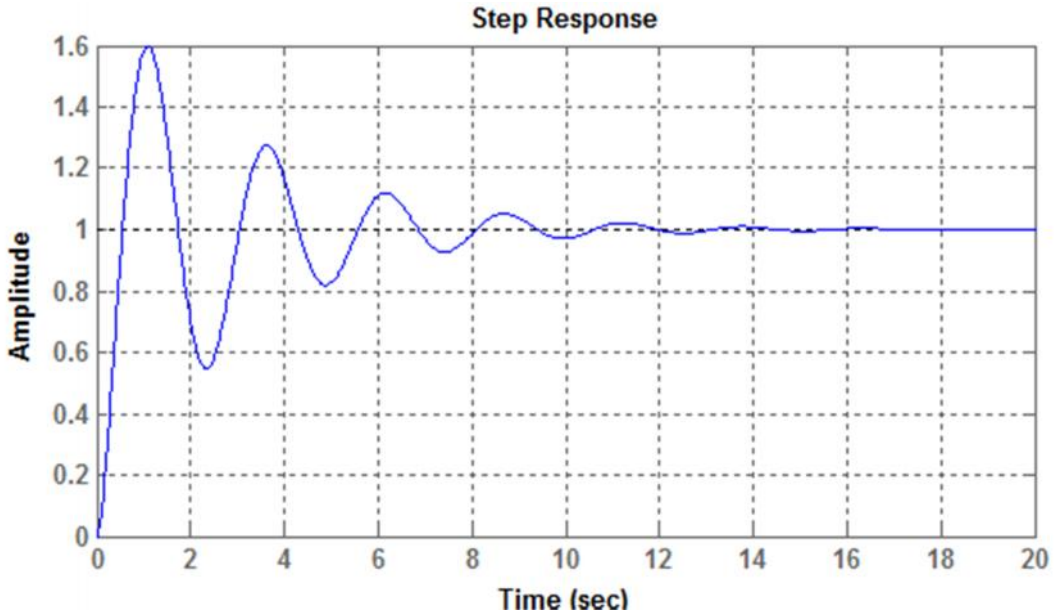

Figure 4.39: step response using Z-N method.

Mp = 60% **tr** = 0.38s **ts** = 11.4s

As shown in Figure 4.39, it can be found easily that classical Ziegler-Nichols did not gave us a good result, since the step response has small oscillation before it reaches the steady state, with slightly large overshoot **MP**=60% and an acceptable settling time **ts**= 11.4s

4.7 Comparison between PSO and DE with Z-N tuning methods:

To show the effectiveness of the proposed methods, a comparison is made between these heuristic optimization techniques (PSO and DE) and classical Z-N methods for tuning PID parameters. To clearly show the difference between the three methods, we plot their best responses obtained by PSO and DE with the one obtained by Z-N in the same one graph, which is shown in Figure 4.40. A comparison of time domain specifications peak overshoot, rise time, settling time and steady state error is done and the results are tabulated in Table 4.6, it can be deduced that the both tuning methods gave better performance compared to Ziegler-Nichols method especially when the system becomes higher than 2nd order. As seen from the Table 4.6, Ziegler-Nichols gives poor rise-time, settling time and highest overshoot. In addition, comparing between the two methods we noticed that DE gives much better result than PSO.

Table 4.6: Comparison Performance between PSO and DE and Z-N.

Tuning method	KP	KI	KD	Rise Time(s)	Settling time (s)	Overshoot (%)	ESS (%)
PSO	2.05	2.71	3.95	0.24	0.37	0.41	0
DE	1.08	5.51	4.50	0.22	0.34	0	0
Z-N	3.99	4.33	0.91	0.38	11.4	60	0

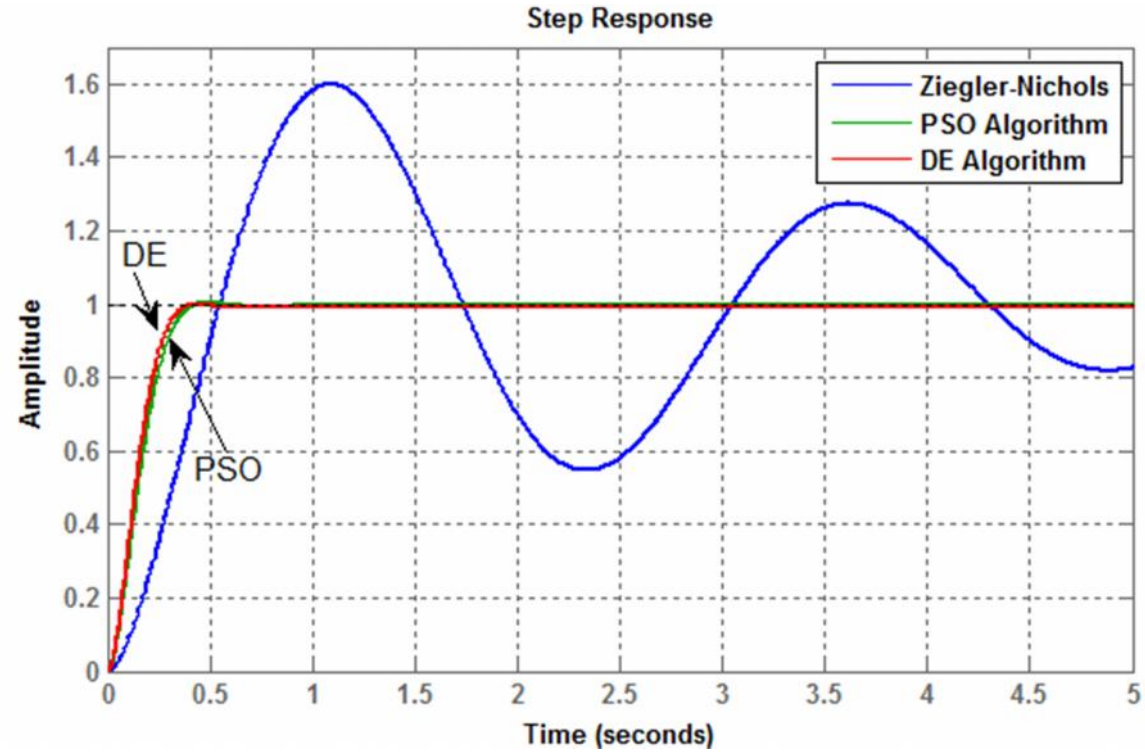

Figure 4.40: step response of PSO and DE and Z-N based PID controller

4.8 Conclusion:

In this chapter, the PID controller has been designed and optimized by Particle Swarm Optimization (PSO) and differential evolution (DE) algorithms. The proposed methods are tested on high order system in comparison with classical Ziegler-Nichols method in order to demonstrate its effectiveness and robustness for solution of the desired optimization problem. The simulation results demonstrate that these heuristic methods (PSO and DE) can improve the control system performance in terms of time domain specifications quickly and accurately compared with classical Ziegler-Nichols method.

Our project has been carried out to get an optimal PID tuning by using particle swarm optimization (PSO) and Differential Evolution (DE) algorithms on high order system. The simulation results allows us to conclude that PSO and DE can perform an efficient search for the optimal PID controller parameters for the system with high order and subsequently prove their viability to solve practical engineering optimization problems.

According to the obtained results, it was shown by comparison of step responses, that these heuristic techniques have out-performed the classical Z-N method in terms of control performances: overshoot, settling time and rising time. Therefore PSO and DE algorithm shows more accuracy and can solve very complex optimization problems. Regarding to the comparison between the two techniques, we have noticed that DE enhances the control performances better than PSO, furthermore, DE is able to undertake local search with a fast convergence rate, To this, as a future work we can suggest to investigate these methods in real time systems.

References:

[1] Michael A. Johnson,Mohammad H.Moradi *"PID Control New Identification and Design Methods"*.

[2] Bechhoefer, John*."Feedback for Physicists: A Tutorial Essay On Control"*. Reviews of Modern Physics (APS Physics).

[3] Araki M " *PID Control*" Control systems,ROBOTICS and automation –Vol II.

[4] Spyros J. G. Ziegler and N. B. Nichols *"Optimum Settings for Automatic Controllers"* Trans. ASME, Vol. 64, 1942, s. 759-768.

[5] S.M.GirirajKumar, Deepak Jayaraj, Anoop.R.Kishan "*PSO based Tuning of a PID Controller for a High Performance Drilling Machine*" International Journal of Computer Applications. 2010

[6] Jakob Vesterstrøm, Ren´ e Thomsen "*A Comparative Study of Differential Evolution,*

Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems"

[7] Momin Aftab,Momin Shadab, Panigrahi Santosh Parwaiz Sadaf, Prof.Javed Taili Faculty "*Optimal Control of Multi-Axis Robotic System using Particle Swarm Optimization*" Annual IEEE India Conference (INDICON). 2013

[8] M. S. Abou Omar, T. Y. Khedr and B. A. Abou Zalam "*Particle Swarm Optimization of Fuzzy Supervisory Controller for Nonlinear Position Control System*" IEEE.2013

[9] Maria-Iuliana DASCĂLU "*Application of Particle Swarm Optimization to Formative E-Assessment in Project Management "* Informatica Economică vol. 15, no.2011

[10] Gowrishankar Kasilingam "*Particle Swarm Optimization Based PID Power System Stabilizer for Synchronous Machine*" International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering.2014

[11] K.E. PARSOPOULOS, M.N. VRAHATIS "*Particle Swarm Optimizer in Noisy and*

Continuously Changing Environments"

[12] Satyobroto Talukder "*Mathematical Modeling and Applications of Particle Swarm Optimization*" Thesis.2008

[13] Chih-Cheng Kao "*Application of Particle Swarm Optimization in Mechanical Design*" 2009

[14] LI Gao-yang, LIU Ming-guang "*The Summary of Differential Evolution Algorithm and its Improvements*" 3rd international Conference on Advanced Computer Theory and Engineering.2010

[15] Wu Lianghong, Wang Yaonan, Zhou Shaowu, Tan Wen "*Design of PID controller with incomplete derivation based on differential evolution algorithm*" Journal of Systems Engineering and Electronics. 2008

[16] Youxin LUO, Xiaoyi CHE "*Tuning PID Control Parameters on Hydraulic Servo Control System Based on Differential Evolution Algorithm*" IEEE. 2010.

[17] Efren Mezura-Montes, Carlos A. Coello Coello, Edy I.T un-Morales "*Simple Feasibility Rules and Differential Evolution for Constrained Optimization"*.

[18] Trias Andromeda, Azli Yahya, Syahrullail Samion, Ameruddin Baharom, Nor Liyana Hashim "*Differential Evolution for Optimization of PID Gain Electrical Discharge Machining Control System"* Transactions of the Canadian Society for Mechanical Engineering. 2013.

[19] Aki Sorsa, Anssi Koskenniemi and Kauko Leiviskä "*Evolutionary algorithms in nonlinear model identification"* Report A No 44, September 2010.

[20] Itishree Ghatuari, Nandan Mishra, Binod Kumar Sahu "*Performance analysis of automatic generation control of a two area interconnected thermal system with non-linear governor using PSO and DE algorithm"* IEEE.2013.

[21] Xiao-lei DONG, Sui-qing LIU, Tao TAO, Shu-ping LI, Kun-lun XIN "*A comparative study of differential evolution and genetic algorithms for optimizing the design ofwater distribution systems"* Journal of Zhejiang University-SCIENCE A (Applied Physics & Engineering).2012.

[22] MOHD S. SAAD, HISHAMUDDIN JAMALUDDIN, INTAN Z. M. DARUS "*PID Controller Tuning Using Evolutionary Algorithms"* E-ISSN: 2224-2856. October 2012

[23] Website: WWW.researchgate.net

[24] Amit R. Khaparde, M. M. Raghuwanshi, L. G. Malik "*Industrial Application of Differential Evolution Algorithm"* International Journal of Soft Computing and Engineering. 2016.

[25] Dr.S.M.Girirajkumar, Atal.A.Kumar, Dr.N.Anantharaman "*Tuning of a PID Controller for a Real Time Industrial Process using Particle Swarm Optimization"* IJCA.2010.

[26] R. A. Krohling and J. P. Rey, "*Design of optimal disturbance rejection PID controllers using genetic algorithm,"* IEEE. 2001.

[27] K.Vijaya Lakshmi, P.Srinivas "*Optimal Tuning of PID Controller using Particle Swarm Optimization"* International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO) – 2015.

[28] Anil Kumar, Dr. Rajeev Gupta "*Compare the results of Tuning of PID controller by using PSO and GA Technique for AVR system"* International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 6, June 2013.

[29] Ibrahem S. Fatah *"PSO-Based Tuning of PID Controller for Speed Control of DC Motor"* Diyala Journal of Engineering Sciences. ISSN 1999-8716.

[30] Trias Andromeda, Azli yahya, Syahrullail Samion, Ameruddin Baharom, Nor Liyana Hashim *"PID Controller Tuning by Differential Evolution Algorithm on EDM Servo Control System"* Applied Mechanics and Materials Vols. 284-287.2013.

[31] Mohd Sazli Saad,Hishamuddin Jamaluddin,Intan Zaurah Mat Darus *"Implementation of PID controller Tuning Using Differential Evolution and Genetic Algorithms"* ICIC International. 2012.