COMPARATIVE STUDY ON APPLICATION OF DIFFERENTIAL EVOLUTION AND PARTICLE SWARM OPTIMIZATION FOR TUNING CONTROLLERS OF BALL AND BEAM SYSTEM AND ROBOT MANIPULATOR

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree
of

MASTER OF TECHNOLOGY
in
ELECTRONICS & COMMUNICATION ENGINEERING
(With Specialization in System Modeling and Control)

By

NARENDRA KUMAR DHAR

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)
JUNE, 2013

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being reported in this dissertation report, entitled **"Comparative study on application of Differential Evolution and Particle Swarm Optimization for tuning controllers of Ball and Beam system and Robot Manipulator"**, is being submitted in partial fulfillment of the requirements for the award of the degree of **Master of Technology** in **System Modeling and Control,** in the Department of Electronics and Communication Engineering, Indian Institute of Technology, Roorkee is an authentic record of my own work, carried out from June 2012 to June 2013, under guidance and supervision of **Dr. M. J. Nigam**, Professor, Department of Electronics and Communication Engineering, Indian Institute of Technology, Roorkee.

The results embodied in this dissertation have not submitted for the award of any other Degree or Diploma.

Date: 17. 06. 2013

Place: Roorkee

[illegible]

Narendra Kumar Dhar

CERTIFICATE

This is to certify that the statement made by the candidate is correct to best of my knowledge and belief.

[illegible]

Date: 17.06.2013

Place: Roorkee

Dr. M. J. Nigam

Professor

Department of E & CE

Indian Institute of Technology, Roorkee

ACKNOWLEDGEMENT

With profound respect and gratitude, I take the opportunity to convey my thanks to Indian Institute of Technology Roorkee for giving me opportunity to present my dissertation on the topic "Comparative study on application of Differential Evolution and Particle Swarm Optimization for tuning controllers of Ball and Beam system and Robot Manipulator".

I am extremely grateful to **Dr. M.J.Nigam,** Professor, Department of Electronics and Communication Engineering, for his co-operation and guidance. I shall always be indebted to him for his valuable guidance, support and motivation for this work and direction of my dissertation. I consider myself fortunate for getting an opportunity to learn and work under his able supervision over the entire period of my association with him.

My sincere thanks to the faculty members of System Modeling and Control for their constant encouragement and constructive suggestions towards the successful completion of this work. My sincere thanks to the laboratory staff to access the computers and other resources at will for completion of this work.

CONTENTS

LIST OF TABLES AND FIGURES

Page no.

CHAPTER - 1

INTRODUCTION

Soft Computing is a collection of techniques covering many areas that come under different classifications in computational intelligence. The computing techniques belong to the various fields such as computer science, machine learning and engineering fields, which undergoes study of model and their analysis of associated complex phenomena. Such benefits have not come forward through the conventional methods. They are far behind in yielding low cost complete solutions. Soft computing techniques use soft techniques contrasting it with classical artificial intelligence of its counterpart. The techniques are developed on the information processing in biological systems. The tasks like surrounding recognition, act according to the plans as per the ideas thought of in order to survive is the eloquent feature of the complex biological information system in humans.. The information processing involves both logical and intuitive processing. Logical processing is what conventional computers are good at, but they are far behind in capability for the later as compared to human beings. The three features are required for any computing system to have human like information processing capability: openness, robustness and real time processing[1]. Openness of a system is its capability to cope with circumstantial and random changes encountered in the real world and also allowing it to extend on its own. If a system has tolerance and also remains stable even if it meets with segregated, incomplete or imprecise information then it is said to be as robust. A system has real time processing characteristic if it reacts in a considerable amount of time if encountered with an event. Real world computing (RWC) systems are said to have these three features. A RWC system is therefore capable of representing the information in distributed manner, processing parallel in huge amount when required, adapting in order to organize itself and learning at the same time to achieve flexibility in information processing. Thus, RWC systems incorporate the soft computing techniques as key ingredient.

There are various techniques which come under the soft computation. Swarm intelligence, evolutionary computation, neural networks, fuzzy logic, probabilistic reasoning are some of the known components. All these have their own distinct nature of working and presenting solutions that suit the systems. The aspect which is common to all these components is that they are not hard bounded by any mathematical formulation.

Swarm intelligence and evolutionary computation are two of the powerful soft computing techniques. A lot of research has been done to identify the better of them. But no such generalization can be done as the algorithms pertaining to both these techniques behave differently depending on the system to which they are applied. A comparative analogy and differences are thus chalked out for the particular system in order to conclude the nature of the two techniques.

Particle Swarm Optimization (PSO) is a search heuristic technique [2]. It considers a set of particles which undergo movement in a particular area and corresponding dimensions. The area and dimensions are based on the parameters to be optimized. The algorithm reaches for both local as well as global searching ability. Identification of local and global best particle in every iteration smoothens the movement and thus leads to the optimized values of the parameters.

Differential evolution (DE) is a population based evolutionary algorithm which reaches out for the best candidate solution [3]. The algorithm pertains to population initialisation, recombination, mutation and selection, thereby obtaining the best of the individuals each and every time. The fitness value and the iterations lead to the optimised value of the parameters.

To draw out the analogies and differences between these two techniques two systems ball and beam and robotic manipulator are considered. The first system is simpler as compared to the second one. This has been done in order to observe how both the algorithms work and behave for the two systems.

The ball and beam control system is one of the most common and easy to handle system. It is a non linear system coupled with servomotors and gears but can be linearized under certain assumptions. Designing the feedback control law and to have a requisite output the controller seems necessary as it manipulates the required input signals to the system. The controller chosen for the purpose is PID. There are two of them and the controller gain parameters are optimized by the aforesaid two soft computing techniques.

The robotic manipulator is a highly non-linear system. It is due to number of joints and links and the degrees of mobility associated with them. The kinematical model coordinates one frame with the other involving transformation from one axis to another whereas the dynamic model beholds the control of end effectors movement due to the force, torque applied. The robotic manipulator used here is PUMA 560 which is a 6 DOF revolute jointed manipulator. The controller designed to regulate the torques and monitor the tracking of end effectors movement is computed torque controller (CTC). The controller considers all the inertial, coriolis, centrifugal forces and the gravity effects to regulate the torques.

1.1 Motivation

During the course work, I came to study about optimization in control systems. The optimization is utmost necessary for the systems to get good hold of output with the required convergence and accuracy. This fascinated me to learn some of the techniques and to apply them on some of the systems to know how they work.

In feedback control algorithm there exists relation between the error input and the output driven by the system. To have a good regulation controllers are used which in turn consider error input and provide regulated input to the given system. But for controllers to work fine the parameters associated to them should be tuned properly. Many times they are tuned manually. But this does not seem to be a good methodology for the complex and higher order systems where there is no direct relation between the controller and the system parameters. For such purposes there should be techniques which can optimize the parameter values and thereby can tune them properly. The soft computation techniques seem to be the robust one and their algorithms pursue optimization in accordance with the system. I chose to use Particle swarm optimization (PSO), a search based technique, and Differential evolution (DE), an evolutionary technique, for my systems. They both are robust in nature and have fast convergence rate.

1.2 Problem statement

The dissertation focuses on the application of soft computing techniques to the ball and beam system and robot manipulator (PUMA 560) in order to draw out the analogies and differences in the two techniques. The aims of this dissertation are:

- Study about the two soft computing techniques PSO and DE and their application.
- Obtain the tuned controllers for the ball and beam system and robot manipulator PUMA 560 using the two techniques.

- Draw out conclusion on the techniques based on the convergence and the accuracy.

1.3 Organization of thesis

The thesis organized in such a way that a clear and precise understanding of work is presented. It has been divided into several chapters. Chapter 1 bears the introduction to work , problem concerned and the motivation for the work. Chapter 2 introduces Differential Evolution (DE), its nature, algorithm and the variants. Another soft computing technique known as Particle Swarm Optimization (PSO) is presented in Chapter 3. It gives idea about its algorithm, the variants and nature of operation. Chapter 4 is about the dynamics of Ball and Beam system. The Robot manipulator kinematics and dynamics are introduced in Chapter 5. It also gives an insight to PUMA 560. Chapter 6 is about the results drawn out from the application of techniques on the two control systems. The conclusion and future work suggestion are put forward in Chapter 7.

CHAPTER – 2
DIFFERENTIAL EVOLUTION (DE)

2.1 Introduction

Differential Evolution, one of the evolutionary computations, first of its kind was developed in 1995 by Storn and Price[4]. DE is a genetic algorithm (GA) based approach which use particular operations on a population in order to minimize an objective function over the course of successive generations. As with other evolutionary algorithms, Alteration and selection form the basis of operation for optimization problem solution which helps in obtaining candidate solutions. DE uses floating-point instead of rigid method of bit manipulation of population member, and arithmetic operations instead of logical operations in mutation, in contrast to classical GAs.

The main concept of Differential evolution is using a perturbation of two members as the vector to add to the third member, which produces a new vector [5]. The new vector then is mixed with the predefined parameters in accordance with certain rules to produce test vectors. This operation is called crossover. The final choice of the operation must bring to bear all members of the population such that a correct selection of vectors is done in order to produce the same number of competitors in the next generation.

2.2 Basic DE steps

The basic DE steps are as follows :

a) Population Initialisation

The first step constitutes the decision of certain parameters for DE algorithm and creating arbitrary initial population in 'n' dimensional space [6].

$$x_{ij}(0) = \text{rand}(0,1)\,(x_{ij}^{U} - x_{ij}^{L}) + x_{ij}^{L} \tag{2.1}$$

where $i = 1,2,\ldots,m$; $j = 1,2,\ldots,n$;

'm' represents number of solution vector. x_{ij}^{U}, x_{ij}^{L} denote the upper and lower limits of the jth variable in the population respectively, rand(0,1) represents a uniformly distributed random value within (0,1).

b) Mutation

Selection of several solution vectors is done randomly, and acquiring the difference between the vectors to multiply scaling factor F and furthermore adding on target vector to assist it mutate in order to vary its characteristic. The target vector can be the best one or anyone among the total individuals depending on the choice made. Two of the traditional common mutation types are[7][8].

$$\text{DE/rand} : v_{i,g+1} = x_{ri,g} + F(x_{r2,g} - x_{r3,g}) \tag{2.2}$$

$$\text{DE/best} : v_{i,g+1} = x_{best} + F(x_{r2,g} - x_{r3,g}) \tag{2.3}$$

The first one considers the two different individuals r2 and r3 along with the arbitrary individual r3 in order to mutate and obtain the desired individual whereas the second one considers the best individual among the whole population to have the requisite characteristic in the resulted individual.

c) Crossover

The crossover operation is considered to increase the variety of the population. After crossover, a trial vector 'u' will be produced [7]. The following formulation thus helps in indentifying whether the i_{th} component is formed form the target vector or the donor vector. X_i is the target vector and V_i is the donor vector.

$$U_{j,i,g+1} = \quad v_{i,j,g+1} \quad \text{if rand} <= CR \tag{2.4}$$

$$x_{i,j,g} \quad \text{if rand} >= CR \tag{2.5}$$

Here, 'rand' is a random number that lies between 0 and 1. CR is the crossover rate and 'g' stands for the generation.

d) Selection

The selection is the final stage after the mutation and crossover operations are finished. The selection of the best solution depends on the fitness value of the individuals. For this every time the fitness value of the individuals are compared with that of the earlier fitness values. The individual having the best fitness value is considered for the selection. This is done for each and every iteration until the maximum numbers of iterations are completed. The best result is obtained when the iteration ends.

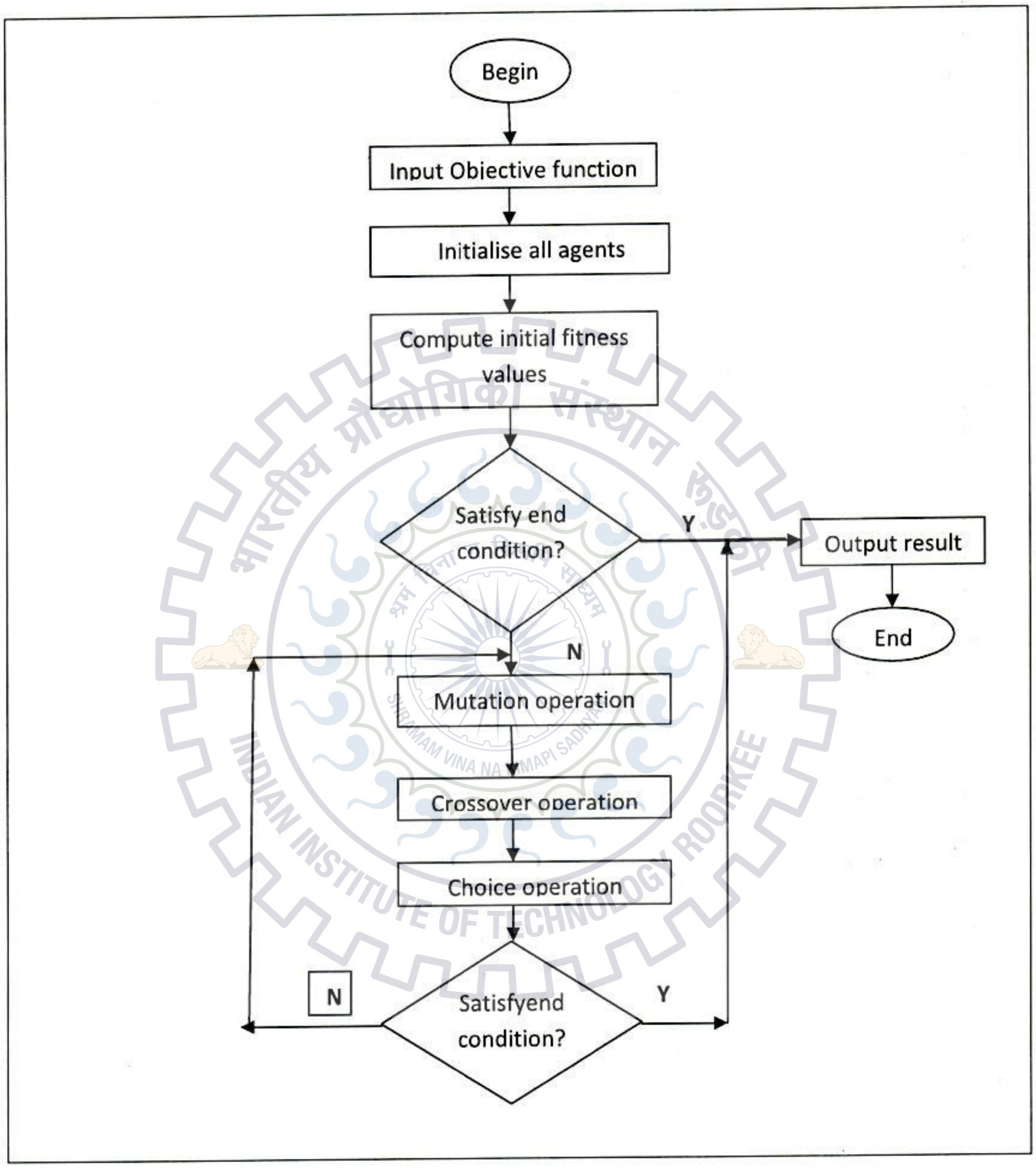


Fig.2.1 Flowchart of DE algorithm

2.3 DE Variants

Variants of DE have been developed depending on the mutation operator, crossover ratio, and mutation or scaling factor .

i) Mutation operator[3]

$$v_{g+1}^{\,i} = x_g^{\,best} + F\,(x_g^{\,r1} - x_g^{\,r2}) \tag{2.6}$$

$$v_{g+1}^{\,i} = x_g^{\,i} + F\,(x_g^{\,r1} - x_g^{\,r2}) \tag{2.7}$$

$$v_{g+1}^{\,i} = x_g^{\,i} + F\,(x_g^{\,best} - x_g^{\,i}) + F\,(x_g^{\,r1} - x_g^{\,r2}) \tag{2.8}$$

where, i is the current index , v_{g+1}^{i} is the mutant individual to be developed, r1 and r2 are random integers mutually different and not equal to the current index i , and F (>0) is a real parameter, called *mutation* or *scaling factor*.

ii) Modified Mutation Factor DE (MFDE)[9]

$$F = s * \sqrt{r(0,1)^2 * d} - b \tag{2.9}$$

Where, d is *linear decreasing factor*, r is a random variable, s is an acceleration factor, b is deceleration factor. The MFDE improves the balance between exploration and exploitation.

iii) *Crossover Ratio*

$$cr = cr_{max}+((\,cr_{max} - cr_{min}\,) * iter) / max\_iterations \tag{2.10}$$

$$cr = 0.5 * (\,1 + rand()\,) \tag{2.11}$$

where, cr_{max} : maximum crossover ratio value

cr_{min} : minimum crossover ratio value

$iter$: current iteration.

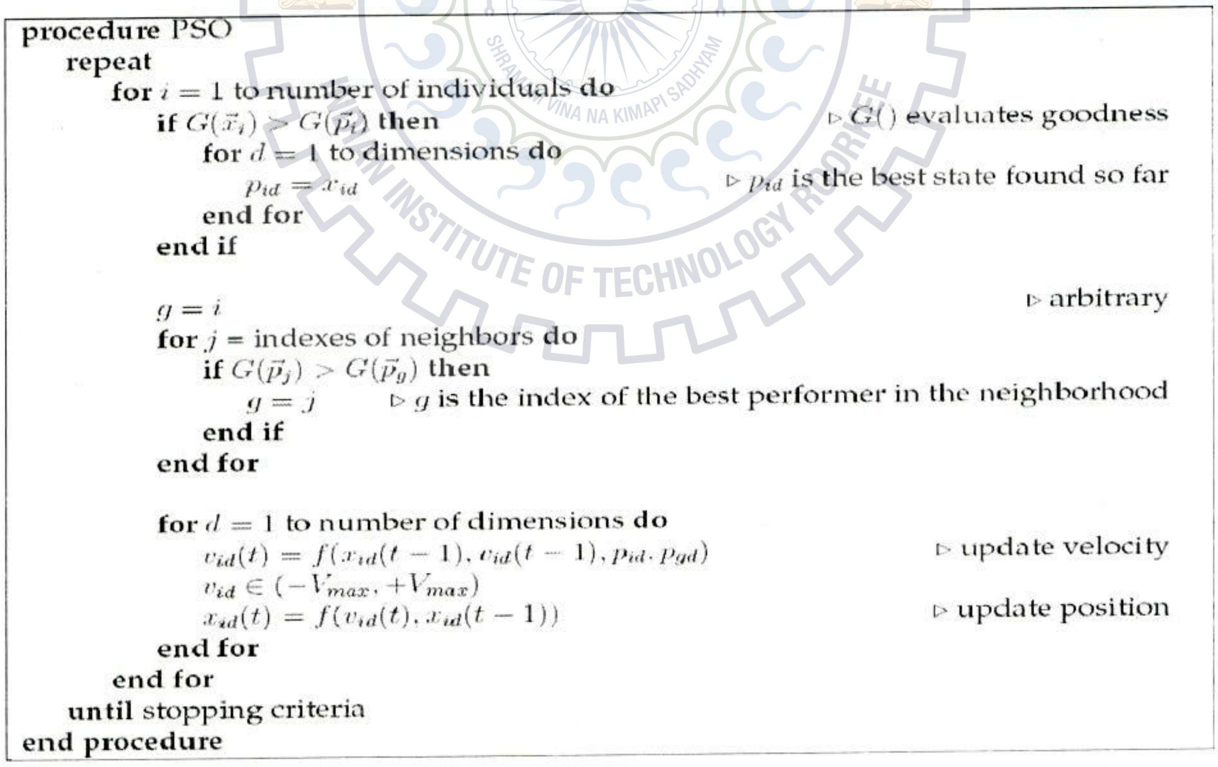
CHAPTER – 3

PARTICLE SWARM OPTIMIZATION

3.1 Introduction

The search methodology comprising population strikes out as a good choice when the search area is large and takes a lot of effort in search criteria. Particle Swarm Optimization (PSO) belongs to that kind of search technique. PSO, first introduced by Dr. Russell C.Eberhart and Dr. James Kenedy, is an algorithm that adapts itself depending on previous successful regions[10]. The population of individuals also known as particles follow a particular social cognitive pattern. The algorithm incorporates diversity as well as convergence.

The algorithm considers two basic and primary operators: Position alteration and Velocity alteration[2]. In each iteration every particle is impelled to move towards its best position in previous occurrence locally as well as the global best position for the whole population. Each particle's new velocity is determined based on its present velocity, and how distant it is locally as well as globally from the best position in the past.. The new velocity value monitors and thus articulates the particle's next position in the given dimensional search area. Repetition of this process is carried out for a certain number of times, or until the proximity to the target is achieved.



```
procedure PSO
    repeat
        for $i = 1$ to number of individuals do
            if $G(\vec{x}_i) > G(\vec{p}_i)$ then                    ▷ $G()$ evaluates goodness
                for $d = 1$ to dimensions do
                    $p_{id} = x_{id}$                               ▷ $p_{id}$ is the best state found so far
                end for
            end if

            $g = i$                                                 ▷ arbitrary
            for $j$ = indexes of neighbors do
                if $G(\vec{p}_j) > G(\vec{p}_g)$ then
                    $g = j$        ▷ $g$ is the index of the best performer in the neighborhood
                end if
            end for

            for $d = 1$ to number of dimensions do
                $v_{id}(t) = f(x_{id}(t-1), v_{id}(t-1), p_{id}, p_{gd})$        ▷ update velocity
                $v_{id} \in (-V_{max}, +V_{max})$
                $x_{id}(t) = f(v_{id}(t), x_{id}(t-1))$                   ▷ update position
            end for
        end for
    until stopping criteria
end procedure
```

Fig.3.1 Particle Swarm Optimization algorithm

2.5.1 Notations used in algorithm

t :the present time sample, $t-1$: the previous time sample

T_{max} : the total time sample for search

$x_{id}(t)$: the i_{th} particle's current position.

$v_{id}(t)$: the i_{th} particle's current velocity.

$[-V_{max}, +V_{max}]$: the extreme limits on velocity.

p_{id} : the i_{th} particle's best position.

p_{gd} : the global best position.

$c1$: social parameter, generally taken to be 2.0.

$c2$: social parameter, generally taken to be 2.0.

$\varphi 1$: a positive number between 0 and 1.

$\varphi 2$: a positive number between 0 and 1.

$w(t)$: the inertia weight.

w_{start} : the initial inertia weight.

w_{end} : the final inertia weight.

χ : the constriction coefficient.

3.2 Standard Particle Swarm Optimizer

The optimization problem treats every parameter of function as a point in search area. The particles movement occur in a heterogeneous space. This is determined by the fitness value. There are regions which prove to be important than others. The particles are evaluated on the basis of objective function. This leads to the identification of better regions of the search space.

$$x_{id}(t) = f(x_{id}(t-1), v_{id}(t-1), p_{id}, p_{gd}) \quad (3.1)$$

$$v_{id}(t) = v_{id}(t-1) + c_1\varphi_1(p_{id} - x_{id}(t-1)) + c_2\varphi_2(p_{gd} - x_{id}(t-1)) \quad (3.2)$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t) \quad (3.3)$$

The wider oscillations lead to explosion in basic PSO. In order to curb these surmounting oscillations binding the velocity to certain limit becomes necessary. The methodology to check this scenario is to consider a parameter V_{max} and thus monitor the velocity from exceeding the limits in any dimension. Generally an upper limit is set as V_{max}, the extremum for each particle.

$$\text{if } v_{id} > V_{max} \text{ then } v_{id} = V_{max} \quad (3.4)$$

$$\text{else if } v_{id} < -V_{max} \text{ then } v_{id} = -V_{max} \quad (3.5)$$

3.2.1 Inertia based Particle Swarm Optimizer

The inertia weight is associated with the previous velocity as a multiplying factor. The decrement in inertia value follows a linear pattern. Every iteration carries a non zero inertia weight value. This helps the particle to maintain the particular direction as it was following in the past occurrence. Initial inertia weight values help in exploratory (global search) to whereas it later shifts to convergence (local search) mode [2].

$$x_{id}(t) = f(w(t), x_{id}(t-1), v_{id}(t-1), p_{id}, p_{gd}) \quad (3.6)$$

$$v_{id}(t) = w(t) * v_{id}(t-1) + c_1\varphi_1(p_{id} - x_{id}(t-1)) + c_2\varphi_2(p_{gd} - x_{id}(t-1)) \quad (3.7)$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t) \quad (3.8)$$

In order to have smooth transition *w(t)* is linearly reduced in each iteration.

$$w(t) = \frac{(T_{max} - t) * (w_{start} - w_{end})}{T_{max}} + w_{end} \tag{3.9}$$

3.2.2 Particle Swarm Optimizer with constriction coefficients

PSO constriction coefficient developed by Clerc[11] results in particle convergence over time. The constriction coefficient allows damping of oscillation amplitude by the consideration of local previous best points. The algorithm leads the particle to converge to a point over time. The constriction coefficient prevents the particle from falling to any unwanted point judging the right constraints are in place. The particle will follow its trajectory around the averaged local(p_{id}) as well as global best position(p_{gd}). The nature of search varies as per the circumstance. The proximity of previous best position and the neighbourhood best position leads the particle to perform a local search whereas farther both the positions are the particle will perform a more spread out nature of search. The search changes the neighbourhood best position and best position in the past taking into account all constraints and values. The particle will shift from local search back to diversified search. The search criteria thus becomes well balanced between local and the global one taking into account the social conditions due to the inclusion of constriction coefficient methodology.

$$x_{id}(t) = f(\chi, x_{id,t-1}, v_{id,t-1}, p_{id}, p_{gd}) \tag{3.10}$$

$$\chi = \frac{2k}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}, \text{ where } \varphi = c_1 + c_2,\ \varphi > 4 \tag{3.11}$$

$$v_{id}(t) = \chi\,[\,v_{id}(t-1) + c_1\varphi_1(p_{id} - x_{id}(t-1)) + c_2\varphi_2(p_{gd} - x_{id}(t-1))] \tag{3.12}$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t) \tag{3.13}$$

Generally, k = 1; $c_1 = 2$;$c_2 = 2$, and $\varphi = 4.1$.

3.2.3 Neighbourhood Topologies

There are three main neighbourhood topologies used: wheel, circle and star. The determination of required individual to use for p_{gd} depends on choice of neighbourhood topology. The circle topology maintains a social connection to its k nearest neighbours (p_{gd} = best particle among k closest neighbours, k generally tends to 2). The wheel topology has an individual at its focus. It is the particle through which information is processed. It separates one particle from the other, p_{fd} (p_{gd} = most favourable among p_{fd} and p_{id}). The star topology is the most preferred topology. Every particle is well linked and associated to every other particle (p_{gd} = most favourable particle found in the swarm).

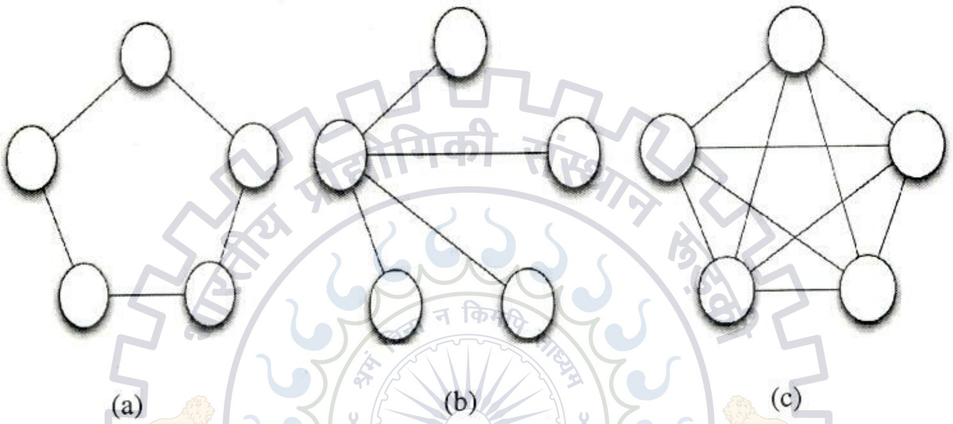


Fig. 3.2 Topology (a) Circle Topology, (b) Wheel Topology, (c) Star Topology

3.3 PSO variants

i) Basic PSO (BPSO)

$$x_{id}(t) = f\,(x_{id,t-1}\,,\, v_{id,t-1}\,,\, p_{id}\,,\, p_{gd}) \tag{3.14}$$

$$v_{id}(t) = v_{id}(t-1) + c_1\varphi_1(p_{id} - x_{id}(t-1)) + c_2\varphi_2(p_{gd} - x_{id}(t-1)) \tag{3.15}$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t) \tag{3.16}$$

The parameters acceleration coefficients (c_1, c_2), inertia weight (w) are constant parameters.

ii) Inertia Weight Approach (IWA)[10]

The inertia weight w(t) should be suitably selected. A well suited inertia weight improvises the balance between local and global search. This thus diminishes the number of iterations to find sufficiently optimal solution.

$$w(t) = w_{max} - \frac{Iter * (w_{max} - w_{min})}{Iter_{max}} \tag{3.17}$$

where, $Iter$ is 'Current Iteration', $Iter_{max}$ is 'Maximum Iterations', w_{max} is 'Maximum inertia weight', w_{min} is 'Minimum inertia weight'.

iii) Constriction factor Approach (CFA)

In CFA approach, the velocity is altered by constriction factor χ [12]. This leads to the enhancement in the performance of hybridized PSO. Selection of constriction factor is an important criteria. This helps in maintaining the velocities in set interval. The velocity values does not exceed the limits.

$$\chi = \frac{2k}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}, \text{ where } \varphi = c_1 + c_2 \text{, } \varphi \text{ greater than } 4 \tag{3.18}$$

Generally, $k = 1$; $c_1 = 2.05$; $c_2 = 2.05$, and $\varphi = 4.1$.

iv) Dynamic PSO

To optimize dynamic system by PSO, inertia weight w(t) is modified as,

$$w = (0.5 + r_3/2) \tag{3.19}$$

where r_3 is uniformly distributed random numbers in [0,1].

v) Fine Grained Inertia Weight Approach (FGIWA)

$$w_i(t+1) = w_i(t) - [\{w_i(t) - 0.4\} * \exp\{-\{|X_{gbest}(t) - X_{i,best}(t)| * (iter/max\_iter)\}\}] \qquad (3.20)$$

This approach combines both the non – linear and exponential characteristics. There is a regulated decrease in value by monitoring the particle's performance iteration wise. The inertia value decreases exponentially to a value approaching 0.4. This thus helps in obtaining global optimum by exploiting all the required searches.

vi) Time Varying Acceleration Coefficients PSO (TVACPSO) [13]

$$c_1(iter) = c_{1s} - (c_{1s} - c_{1e}) * (iter/(max\_iter-1)) \qquad (3.21)$$

$$c_2(iter) = c_{2s} - (c_{2s} - c_{2e}) * (iter/(max\_iter-1)) \qquad (3.22)$$

This particular approach provides a huge diversity in early iteration and convergence for later iterations. c_1(iter), c_2(iter) are the local best weight and the global best weight at particular iteration *iter* respectively. c_{1s}, c_{1e} are the initial and last iteration local best weight. c_{2s} , c_{2e} are the global best acceleration coefficient for initial and last iteration.

CHAPTER - 4

BALL AND BEAM SYSTEM DYNAMICS

4.1 Introduction

The setup used for ball and beam system is Ball and Beam GBB1004 by the courtesy of Googol Technology.

The dynamics of ball and beam system possesses non-linearity. The practical implementation involves additional non-linearity also. Deadband, backlash due to motor, sensing discrete position, and disturbed rolling on the surface are some of them.

The mechanical plant consists of a base, a beam, a ball, a lever arm, a gear, a support block, a motor and an embedded electrical power supply. The ball can roll freely along the whole length of the beam. At one side the beam is connected to a fixed support block and on the other to a movable lever arm. The motor through gear controls the motion of the lever arm. There is optical incremental encoder built inside along with the motor that gives the information about the current rotary position. The linear potentiometer along the beam gives the actual linear position of the ball along the beam. These two positions are fed back to have the organized closed loop control.

The angle of the beam is changed by alpha when the servo gear turns by the angle theta. Any movement away from the horizontal position, the gravity causes the ball to roll along the beam. The purpose is therefore to have a control algorithm which can stabilize the ball at a required position along the beam. The design is based on the non-linear Lagrangian equation of motion. The design is better as compared to expressing all the forces and the other geometric constraints.

The transfer function of the motor and the system needs to be presented separately in order to have accurate dynamics of setup.

4.2 Model for Motor

The transfer function of a DC motor with respect to position and the applied voltage is given by[14]:

$$G_m(s) = \frac{\theta(s)}{V(s)} = \frac{K}{s(\tau s+1)} \tag{4.1}$$

For the evaluation of system performance the values of motor parameters considered are:

K = 0.7/rev/sec/volts, $\tau = 0.014$sec.

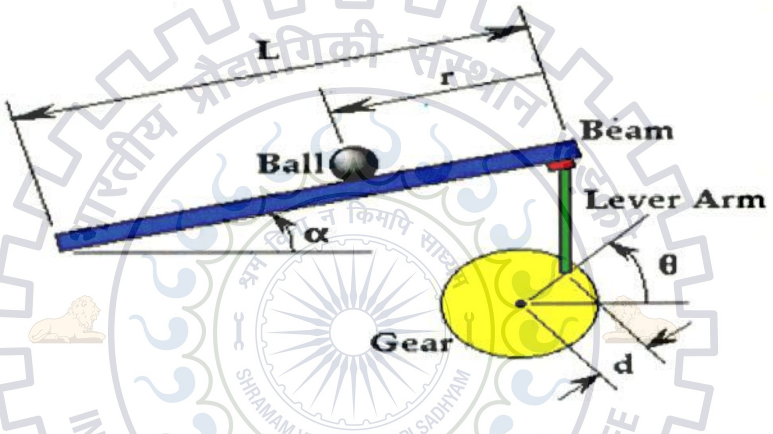


Fig 4.1 A Ball and Beam system schematic

4.3 Model for Ball and Beam system

Ball and Beam system is a Single Input Single Output(SISO) system. The beam angle bears a relationship with the rotation angle of the gear θ in the following manner :

$$\alpha = \frac{d}{L}\theta \tag{4.2}$$

The controller is thus designed in order to keep the ball at requisite position by manipulating the gear angle θ. The forces such as inertia, gravity and centrifugal one affect the dynamics of the ball. The ball linear acceleration is thus given by :

$$(\frac{J}{R^2} + m)\ddot{r} + mgsin\alpha \; - m\dot{r}\dot{\alpha}^2 = 0 \tag{4.3}$$

The system is linearized and the transfer function is obtained keeping the angle □ in the neighbourhood of 0.

Based on mechanical dynamics, the transfer function in open-loop mode for the system can be approximated by the double integrator[15][16][17] :

$$\begin{tikzcd}\theta(s) \arrow[r] & \boxed{W(s)} \arrow[r] & X(s)\end{tikzcd}$$

Fig 4.2 Transfer relation between input and output

$$W(s) = \frac{X(s)}{\theta(s)} = \frac{mgd}{L(\frac{J}{R^2}+m)s^2} \tag{4.4}$$

Where, ‘g’ is gravitational acceleration, ‘m’ is the ball’s mass, ‘L’ is the length of the beam, ‘J’ is the ball moment of inertia, ‘R’ is the radius of the ball, ‘d’ is the distance between the centre of the gear and the joint of the lever arm.

X(s) and Θ(s) are the Laplace transformation of the output (position of the ball) and input(beam angle) of the system.

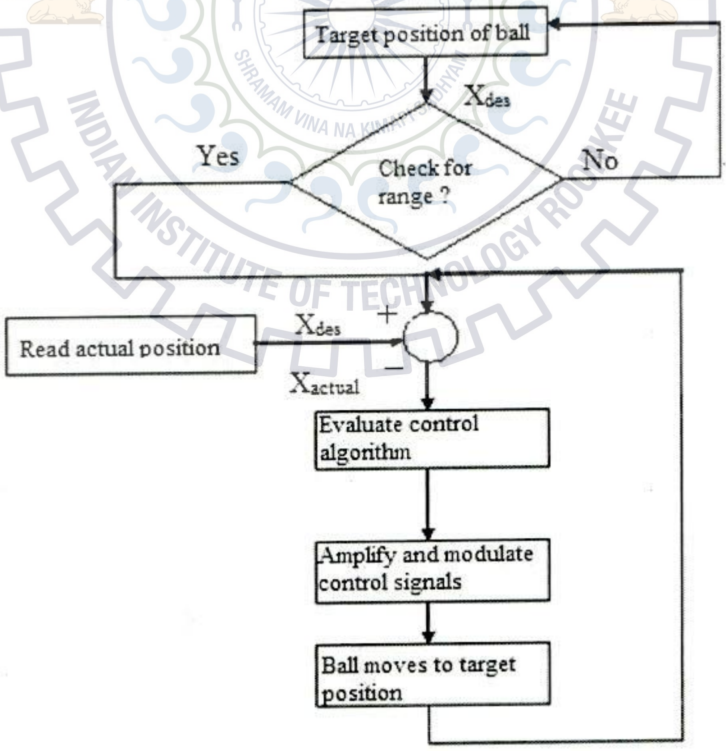


Fig4.3 Control algorithm flow-chart of Ball and Beam system

CHAPTER – 5

ROBOT MANIPULATOR DYNAMICS

5.1 Introduction

The Czech playwright, essayist and novelist Karel Capek was the first to use the word 'robot' in 1921 in his satirical drama entitled Rossum's Universal Robots. There is a Czech word 'robota' which literally means "forced labourer" or "slave labourer".

The robot is visualised as a machine that irrespective of exterior can operate in an environment modified by its own. The 'mechanical system' provides capacity for action which involves locomotion and manipulation apparatus. The 'sensory system' provides capacity for perception acquiring internal as well as external status for conditioning, processing and information retrieval. The 'control system' provides the necessary connection to perception which thereby decides action execution with respect to imposed constraints.

"A robot is a software controlled mechanical device that uses sensors to guide or move end-effectors through programmed motions in a workspace in order to manipulate physical objects."

5.2 Robot Manipulator structure

A manipulator has several joints with the help of which number of links are connected to each other[18]. The wrist of the manipulator is the final link and the base link is ground connected or to a foundation. The three dimensional space movement of the rigid body in a plane is due to manipulator's links and also the motion is constrained with respect to adjacent links. A link physically has at least two nodes so that it can join with the other links. The link having two nodes is the simplest case whereas there may be three and four nodes in parallel manipulator linkages. Binary, ternary, quaternary links can be found in manipulators.

Two or more links connected at their nodes and allowing potential motion between the connecting links is said to be as *joint*. A predetermined path for the motion (constrained motion) irrespective of the forces applied, the joint is said to form a *kinematic pair*. Therefore a joint's motion can be either free or controlled. A desired way movement of the links is due to actuators providing power in controlled joints. The position of other joints affects the status of freely moving joints. The kinematic pairs at each node forming the chain of links is called *kinematic chain*. The links and joints are assembled and interconnected in a way provides a controlled output motion due to supplied input motion. Kinematic chains vary being the open type in serial linkages and the table platforms having the closed types.

The chain of connected links through powered joints help in achieving the desired final motion of the end-effector. Therefore, a kinematic linkage being the constrained one fixed (grounded) at one of its links (or joints) is said to be as *mechanism*. The input power of the robot is controlled and a desired motion of the end-effector is provided by the mechanism. Thus 'n' links can form 'n' distinct mechanisms and one mechanism is said to be as the inversion of the other. Generally, in robotic manipulator mechanisms there is a physical distinction of links and joints. Different cross sections and sizes of links are used to provide effective manipulation in commercial manipulator linkage. Its normally the three joints responsible for the action of arm and body and two or three joints for actuation of wrist.

The mechanism should have proper distribution of *degrees of mobility* so that *degrees of freedom* exist for execution of given task.

5.3 Modeling of robot manipulator

The model of a system is basically how the real system is represented. There are different number of ways how a system can be modelled. The type, complexity level, and nature shows the purpose and aspect of the system[18]. The representation of system mathematically is generally what is called as modeling. The two models which are basically used are:

1. Kinematic model

2. Dynamic model

5.3.1 Kinematic model

Kinematic model shows mathematically how the joints and links are mechanically and geometrically arranged. The model associates the required joint parameter values and the position outcome of the links and end effectors. There exist dynamic laws of motion which can't be put through the kinematic model. The two ways of representing the kinematic model are direct and inverse kinematics.

Direct Kinematics

This particular approach has variables associated with joints fixed, and the solution is put in the form of orientation and position of the tool with respect to the particular frame connected to the base. The relationship between the base frames with that of the end effector frame is propounded by this kinematics by developing a transformation matrix.

The transformation matrix consists of twelve nontrivial elements which binds one frame with respect to another. The non reducible position portion of the matrix consists of the three elements. The reducible nine elements to three elements belonging to the orientation part is done by the Euler angles. Therefore, representation of one frame with respect to another involves at least three elements. This can be reduced to four only using Denavit-Hartenberg (DH) technique. However, this is done by restricting the selected link frame.

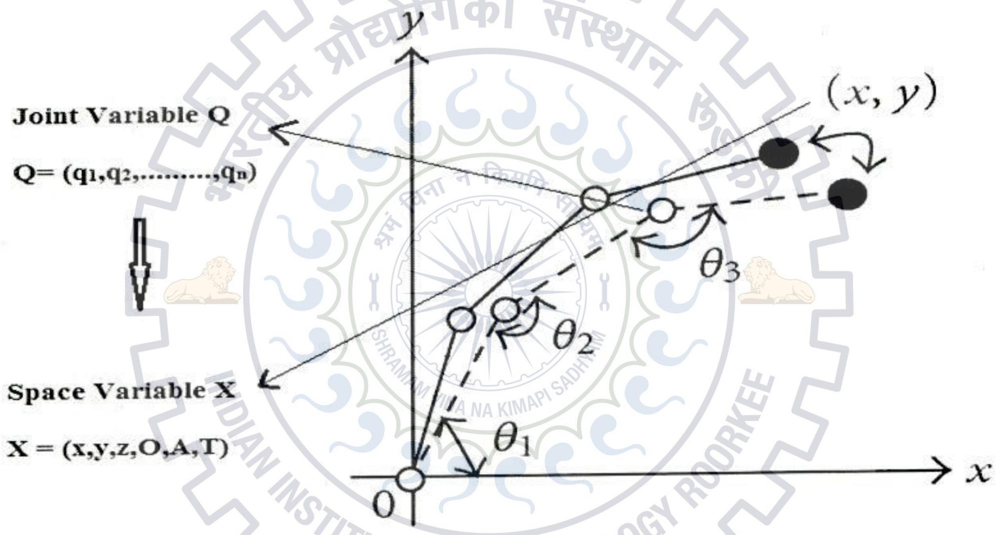


Fig.5.1 Direct kinematics schematic

Denavit-Hartenberg (DH) parameter:

The reference frame is associated with each frame from the base to end effector for the determination of the motion of manipulator. The DH parameter is thus required for such assignment. The DH technique thus reduces the number of elements from six to four only for representing the frame relationship F_i to F_{i-1} .

DH parameters terminology :

Link offset (d_i) : The distance to the point where common perpendicular to the axis Z_i and Z_{i-1} is present.

Link length (a_i) : The common perpendicular's length to Z_i and Z_{i-1} axis.

Link angle (Θ_i) : The angle made by the common perpendicular to the vector X_{i-1} and around the Z_{i-1} axis.

Link twist ($\square_i$) : The angle made by the vector Z_i with Z_{i-1} .

Assignment of link frame :

The assignment of link frame involves few basic rules :

1. The link frame's Z_i vector always lies on the joint axis leaving the end effector frame.
2. The common perpendicular oriented from axes Z_{i-1} to Z_i has vector X_i along it.
3. The intersection of Z_i joint axes and the common perpendicular to axes Z_i and Z_{i-1} has link frame F_i's origin.

Inverse Kinematics:

The inverse kinematics aims at determining joint parameters of the manipulator in order to achieve the desired position of the end effectors. As compared to the forward kinematics problem which has DH parameter approach, it has no specific approach to achieve the required solution. The solution to the inverse kinematics problem is much more important and useful than the forward one. It maps the space configurations of the manipulator to the joint configurations thereby proving an important approach to obtain the desired position of the end effectors.

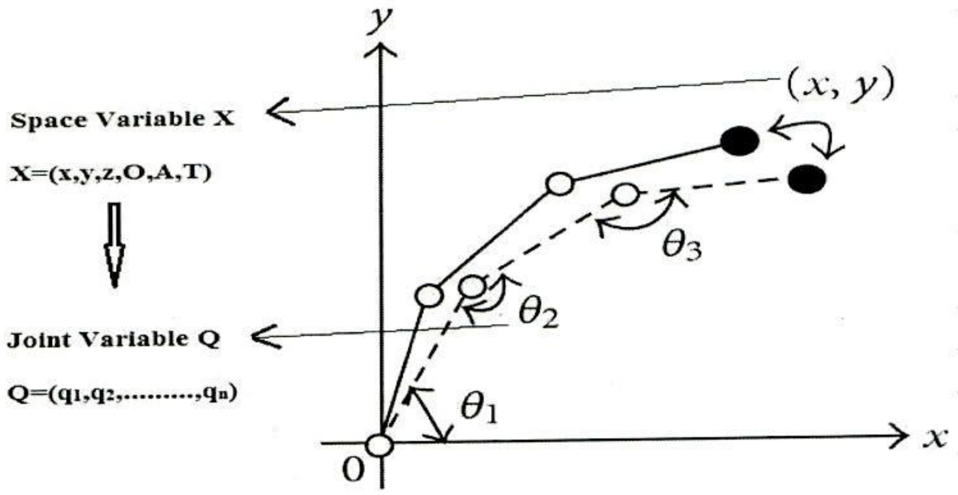


Fig.5.2 Inverse Kinematics schematic

5.3.2 Dynamic model

The manipulator is a highly coupled mechanical device consisting of several joints and links[19]. Being of mechanical nature, its dynamics is governed by dynamic laws. These dynamic laws consider the applied forces and torques and thereby involve differential equations.

Robot manipulator is n-DOF system connected by different joints. The system's kinetic energy K(v,$\dot{v}$) is thus given by,

$$K(v,\dot{v}) = \frac{1}{2}\,\dot{v}^{T}\,M(v)\,\dot{v} \tag{5.1}$$

Where M(v) is nxn dimension mass inertia matrix. M(v) is positive definite symmetric matrix for all 'v'. The kinetic energy depends on the velocity and thus have a specific representation. The potential energy U(v) whereas has no particular form.

The Lagrangian L(v,$\dot{v}$) of n-DOF manipulator is the relation between its kinetic energy K and potential energy U.

$$L(v,\dot{v}) = \frac{1}{2}\,\dot{v}^{T}\,M(v)\,\dot{v} - U\,(v) \tag{5.2}$$

The Lagrange – Euler equation of motion is :

$$\frac{d}{dt}\left[\frac{\delta}{\delta\dot{v}}\left[\frac{1}{2}\,\dot{v}^{T}M(v)\,\dot{v}\right]\right] - \frac{\delta}{\delta\dot{v}}\left[\frac{1}{2}\,\dot{v}^{T}\,M(v)\,\dot{v}\right] + \frac{\delta U(v)}{\delta v} = \tau \tag{5.3}$$

$$\frac{\delta}{\delta\dot{v}}\left[\frac{1}{2}\,\dot{v}^{T}\,M(v)\,\dot{v}\right] = M(v)\dot{v} \tag{5.4}$$

$$\frac{d}{dt}\left[\frac{\delta}{\delta\dot{v}}\left[\frac{1}{2}\,\dot{v}^{T}M(v)\,\dot{v}\right]\right] = M(v)\ddot{v} + \dot{M}(v)\dot{v} \tag{5.5}$$

The equation of motion becomes,

$$M(v)\ddot{v} + \dot{M}(v)\dot{v} - \frac{\delta}{\delta\dot{v}}\left[\frac{1}{2}\,\dot{v}^{T}\,M(v)\,\dot{v}\right] + \frac{\delta U(v)}{\delta v} = \tau \tag{5.6}$$

The equation can also be written in form of :

$$M(v)\ddot{v} + C(v,\dot{v})\dot{v} + G(v) = \tau \tag{5.7}$$

where,

$$C(v,\dot{v})\dot{v} = \dot{M}(v)\dot{v} - \frac{\delta}{\delta\dot{v}}\left[\frac{1}{2}\,\dot{v}^{T}\,M(v)\,\dot{v}\right] \tag{5.8}$$

$$G(v) = \frac{\delta U(v)}{\delta v} \tag{5.9}$$

$C(v,\dot{v})\dot{v}$ is n dimensional vector of Centrifugal and Coriolis forces. *G(v)* is a n dimensional vector of gravitational forces. τ is n dimension vector of torques or forces applied by joint actuators.

5.4 Robot Manipulator Control

The problem to be pursued is planning a robot trajectory which follows the nominal trajectory based on the commands issued to the joint actuators that causes the manipulator to track or follow the desired trajectory. This is said to be as the 'Control problem' and for achieving solution numerous techniques have been proposed.

5.4.1 Control methods

There are two methods to control the manipulator :

- Control of task space
- Control of joint space

Control by task space refers to the determination of position of the end effector corresponding to the coordinate frame when the joint space output is available. This is done by forward kinematics. The error if any is found in terms of position. Joint space problem is not considered.

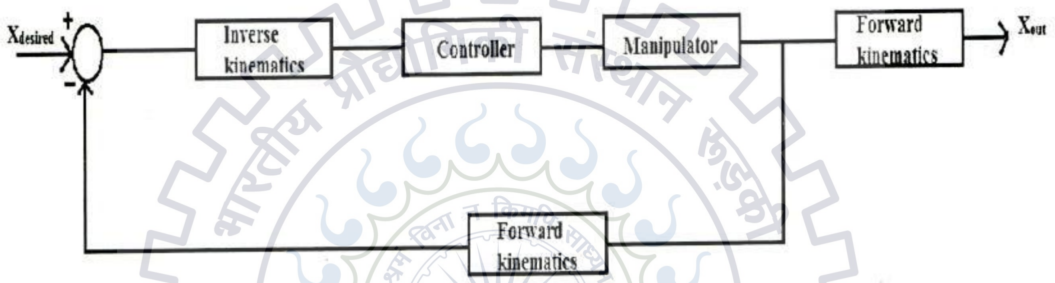


Fig.5.3 Task space control schematic

Control by joint space aims at controlling the joint parameters in terms of angle for revolute joints and length for the prismatic joints. The inverse kinematics is used to determine the joint parameters. The error if occurs in the joint angles are considered for the problem. If the parameters are available in task space then they are converted to the joint space using transformation matrix.

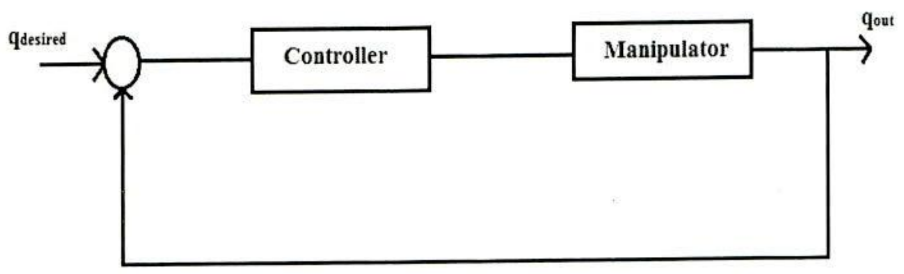


Fig.5.4 Joint space control schematic

The Computed Torque Control approach regulates the joint torque[20]. The technique develops directly on the dynamic model. The approach cancels the gravity effects and also the coriolis and centrifugal force as well as the manipulator inertia tensor. The technique relies on the estimated values gravity factor, coriolis and centrifugal force and the inertia tensor. This is a powerful nonlinear controller. The principle lies in the feedback linearization and thereby computes the required arm torques. The manipulator's desired trajectory and error corresponding to it is given by,

$$e(t) = q_d(t) - q_o(t) \tag{5.10}$$

e(t) : error in the trajectory

$q_d(t)$: desired trajectory

$q_o(t)$: actual output trajectory

The principle being the nonlinear feedback one provides a method for tracking desired trajectory. It involves a proportional-plus-derivative (PD) feedback for computing torque.

$$\square = M(q)\,(\ddot{q}_d + K_v.\,\dot{e} + K_p.e\,) + N(q,\,\ddot{q}) \tag{5.11}$$

the governed error is given by

$$(\ddot{q}_d + K_v.\,\dot{e} + K_p.e\,) = 0 \tag{5.12}$$

K_v and K_p are the controller gains.

As per theory, the error in tracking converges to zero. The CTC design involves two feedback loops, the inner and outer ones. The inner loop being the compensate one and the other one tracks the error.

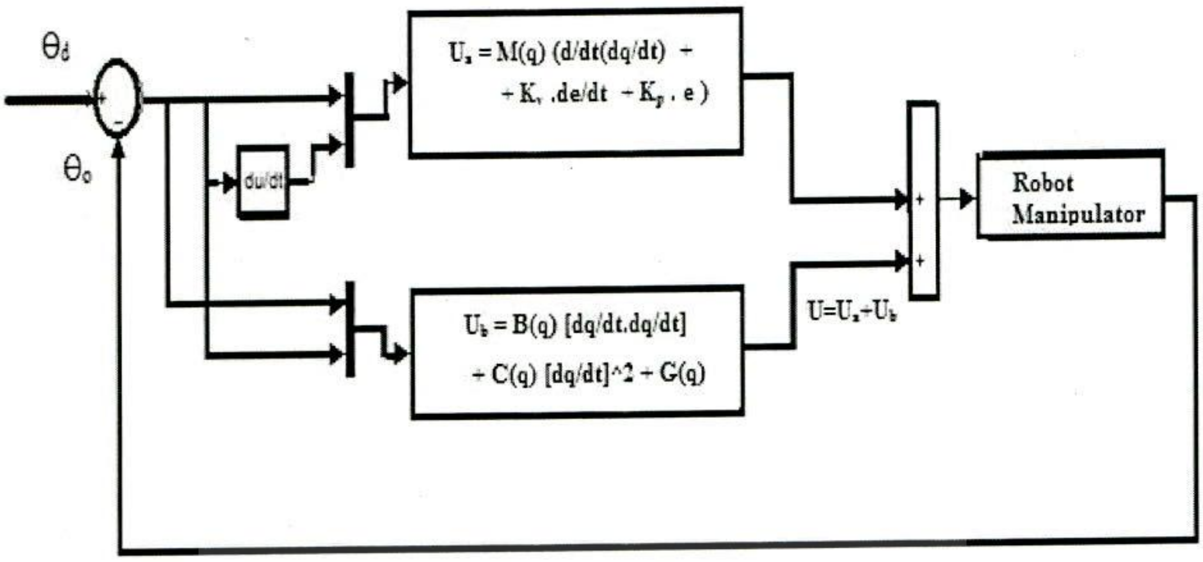

θd
θo
du/dt
Ua = M(q) (d/dt(dq/dt) + + Kv .de/dt + Kp . e)
Ub = B(q) [dq/dt.dq/dt] + C(q) [dq/dt]^2 + G(q)
U=Ua+Ub
Robot Manipulator

Fig.5.5 Computed Torque Controller schematic

5.5 PUMA 560

The PUMA 560 is considered to be one of the most popular industrial robot from the operation point of view. In order to illustrate computational developments, various research issues and concepts it has been particularly used. It bears a wrist partition and has six degrees of freedom. It strikes similarities with the human arm and its rotational aspect bears waist, shoulder and elbow rotation. The orientation of gripper within the workspace is smoothly allowed due to the three degree of freedom wrist[21]. The jointed spherical manipulator has six revolute joints. The spherical wrist is so formed as a single point intersection of the last three joint axes occur. RPY (spherical) wrist is formed out of the last three revolute joints. The identification of joints once being done, the manipulator configuration along with the coordination frames with the links assigned can easily be drawn. Its a straightforward task of assigning frame for each joint link.

CHAPTER – 6

SIMULATION RESULTS

Simulation environment

The simulation software used is MATLAB2008b and Simulink.

6.1 Ball and Beam system

The ball and beam system model is controlled by two PID controllers[22]. One of them regulates the system while the other one monitors the input to the servomotor considering the error in the system.

Both the PID controller parameters are in turn optimized by the soft computing technique. Simulation has been carried out using two of the techniques.

- Differential Evolution
- Particle Swarm Optimization

To obtain the desired position of the ball on the beam step input with the final magnitude of 1 has been considered. The fitness function used to obtain the optimized values of the PID parameters is Integral Square of error.

$$ISE = \int_0^{tf} e^2 \, dt \tag{6.1}$$

e stands out for the error in the position of the ball.

tf is the total time of the simulation.

Transfer functions used for the motor and the ball and beam system are :

Motor : $$G_m(s) = \frac{\theta(s)}{V(s)} = \frac{K}{s(\tau s+1)} \tag{6.2}$$

System : $$W(s) = \frac{X(s)}{\theta(s)} = \frac{mgd}{L(\frac{J}{R^2}+m)s^2} \tag{6.3}$$

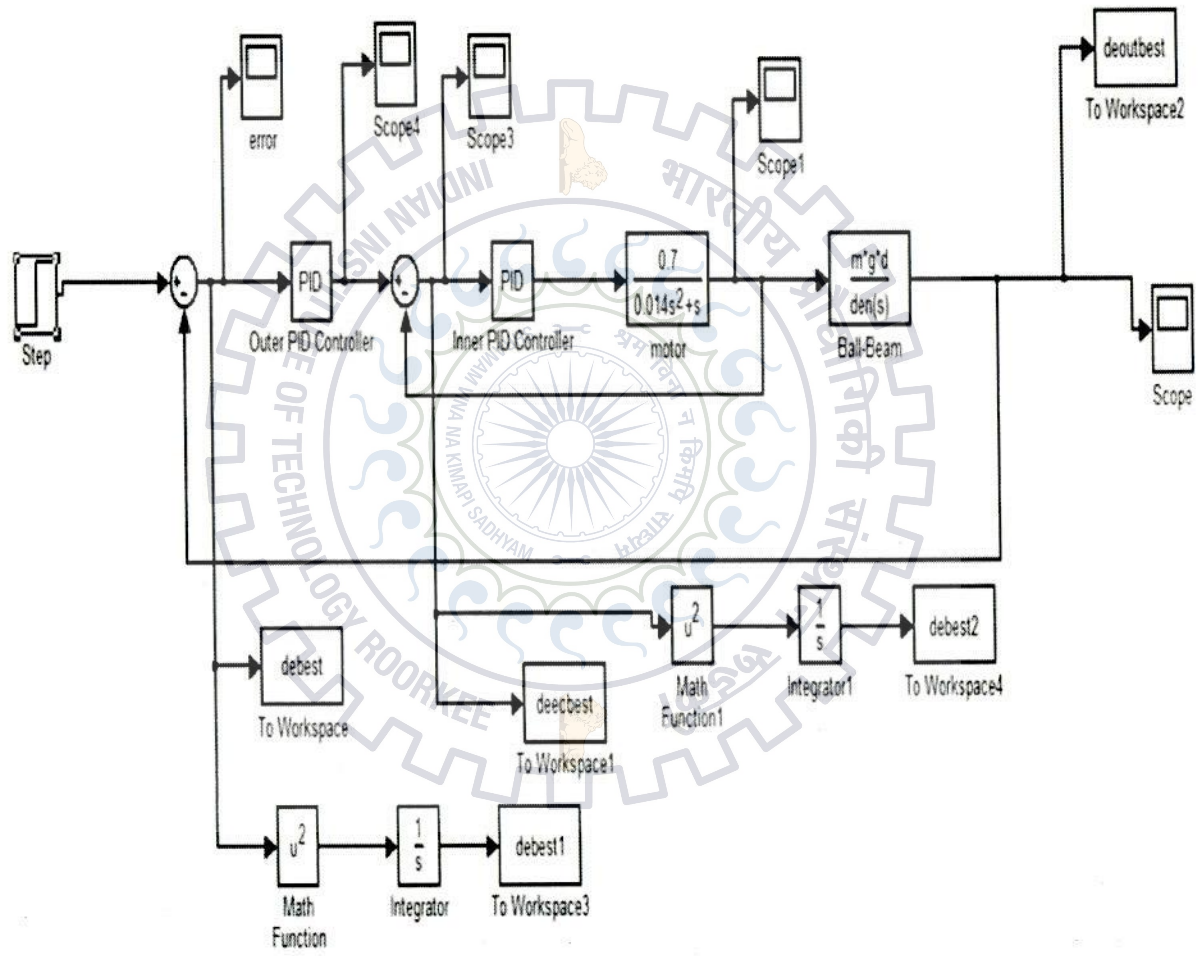


Fig.6.1 Simulink diagram for ball and beam

The first technique used is PSO for optimizing the parameters of PID controllers. The results show the convergence of the system to the desired position. The error profile has also been drawn out in order to understand the convergence to the steady state. Different number of iterations has been carried out to monitor the behaviour of the system.

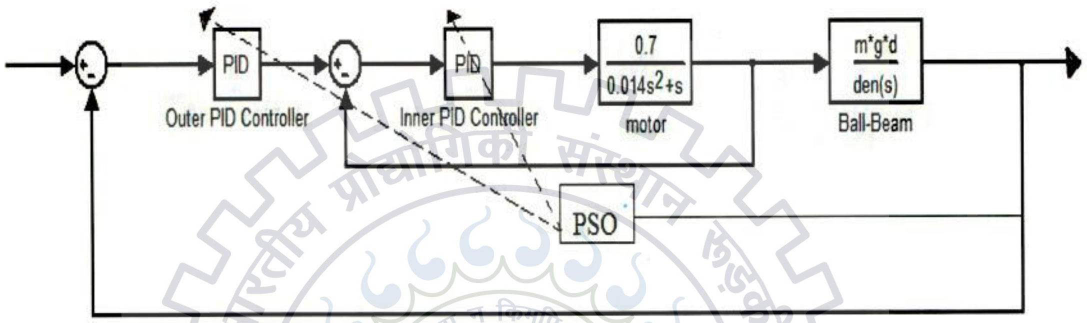


Fig.6.2 PSO optimized PID controllers

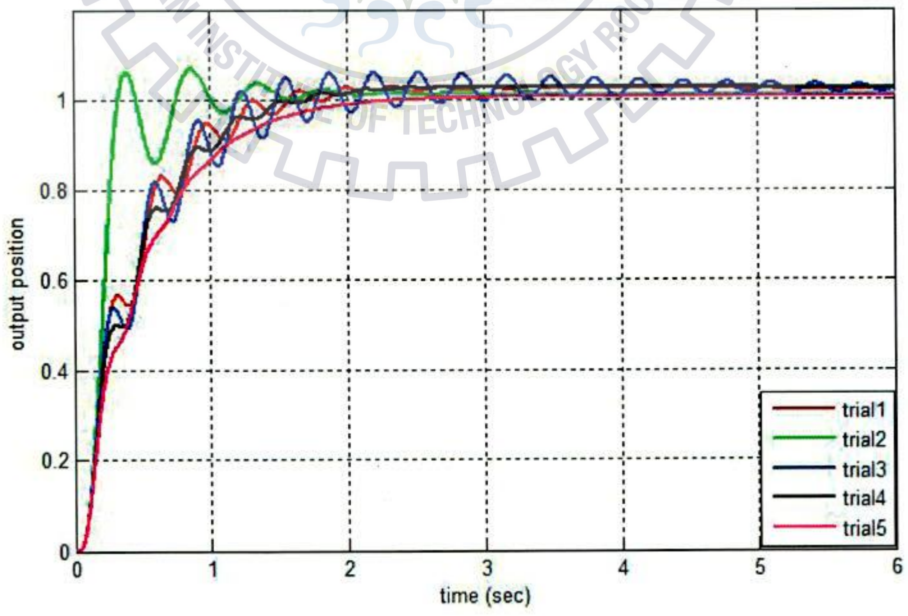


Fig.6.3 PSO trials' Step Response

Table 6.1. PID parameters for PSO trials

	Outer PID			Inner PID		
	K_P	K_D	K_I	K_P	K_D	K_I
Trial 1	11.7867	5.3054	0.7264	12.8185	0.0197	0.7193
Trial 2	12.6990	2.7124	0.7197	14.2826	0.3342	0.8275
Trial 3	14.2539	6.6456	0.9191	11.5639	0.0162	1.1963
Trial 4	11.6541	5.5851	0.7912	14.6514	0.0369	1.5199
Trial 5	14.2676	7.1934	0.1828	12.9867	0.1676	0.2701

Table 6.2. Objective function values

	Trial1	Trial2	Trial3	Trial4	Trial5
ISE value	0.2445	0. 1681	0.2526	0.2569	0.2766

Number of trials considering swarm size of 50 and iterations ranging up to 2000 were simulated on the system. The response of the system is shown for five of the best trials. Of all the trials of PSO conducted on the system trial 2 seems the best for given number of iterations. The objective function value for the trial 2 is 0.1681 which is the least among all the objective function values for the 5 trials. The PID parameters for the trial 2 are given in table 6.2. The controller tuned with the mentioned gain parameters gives good steady-state response, least rise time as well as least settling time.

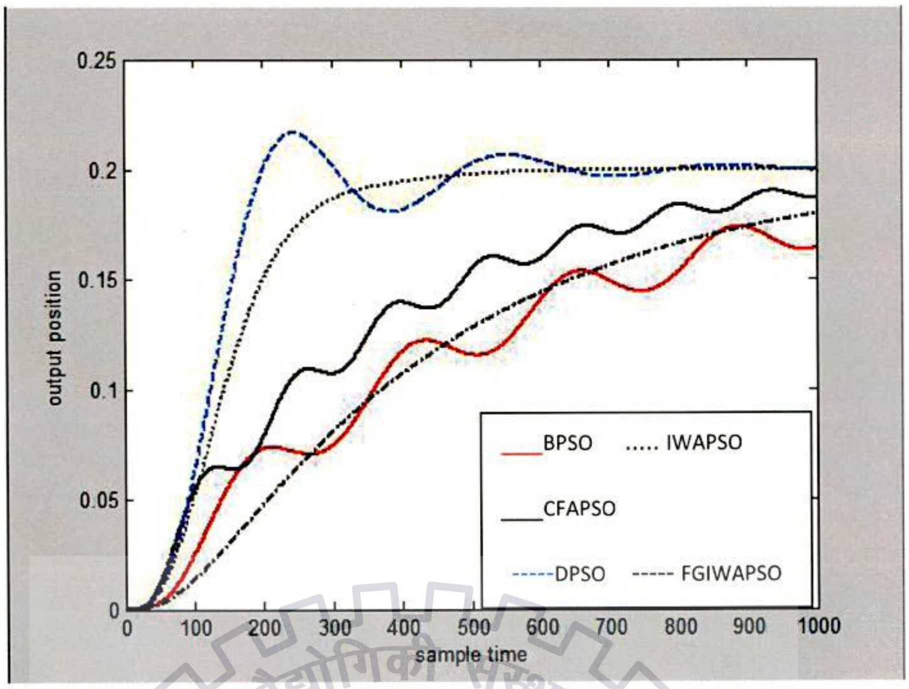


Fig.6.4 Output position convergence after 5 iterations

The above figure 6.4 shows the output position of the ball after 5 iterations. The five variants of PSO have been used. The convergence of IWAPSO to the steady state is fastest among all the variants. Also it does not show any overshoot and the graph shows the smooth nature.

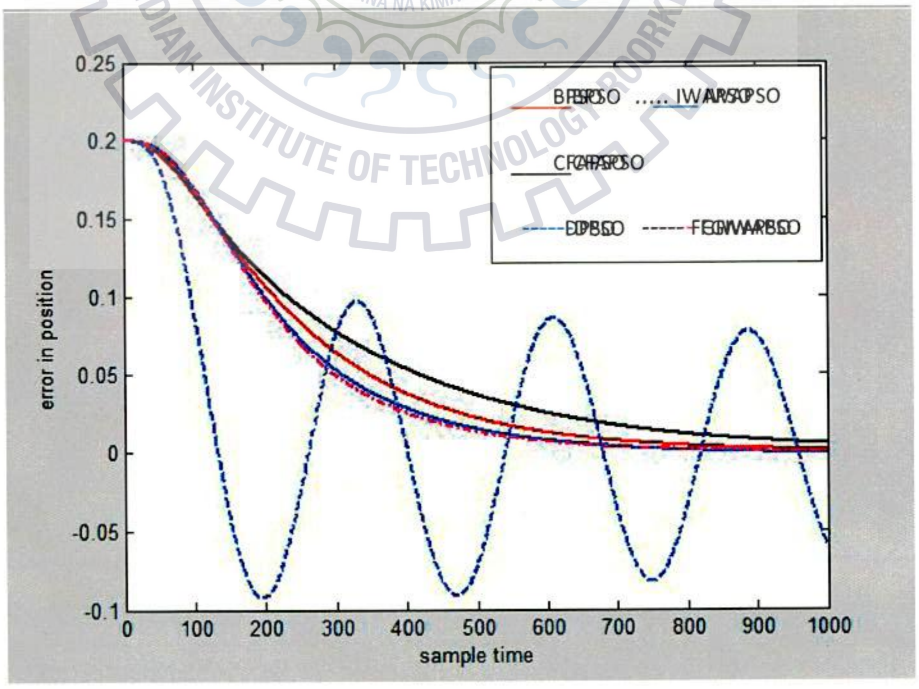


Fig.6.5 Error convergence after 5 iterations

The figure 6.5 shows the error profile for the PSO variants. The DPSO (Dynamic PSO) variant shows the maximum error in the position whereas FGIWAPSO and IWAPSO show minimum error. Their steady state error converges to zero.

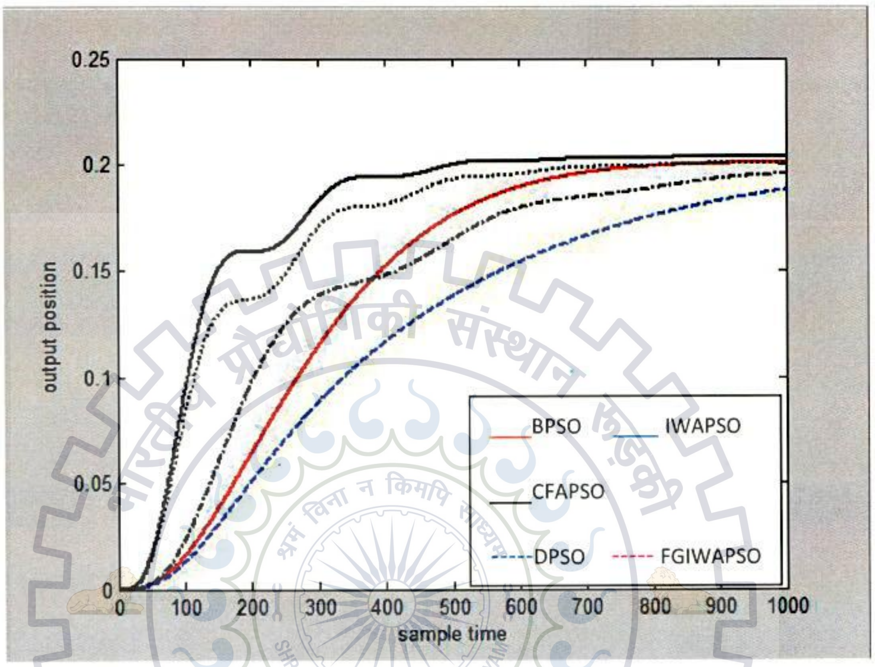


Fig.6.6 Output position convergence after 20 iterations

With the increase in number of iterations the variants behave differently. The CFAPSO variant leads the ball to reach the target position very quickly but its transient nature is not smooth. There tends to be some overshoots and undershoots in this case i.e. the ball does not settle easily at the required position. The BPSO takes a little more time but the ball reaches its target position and settles down smoothly.

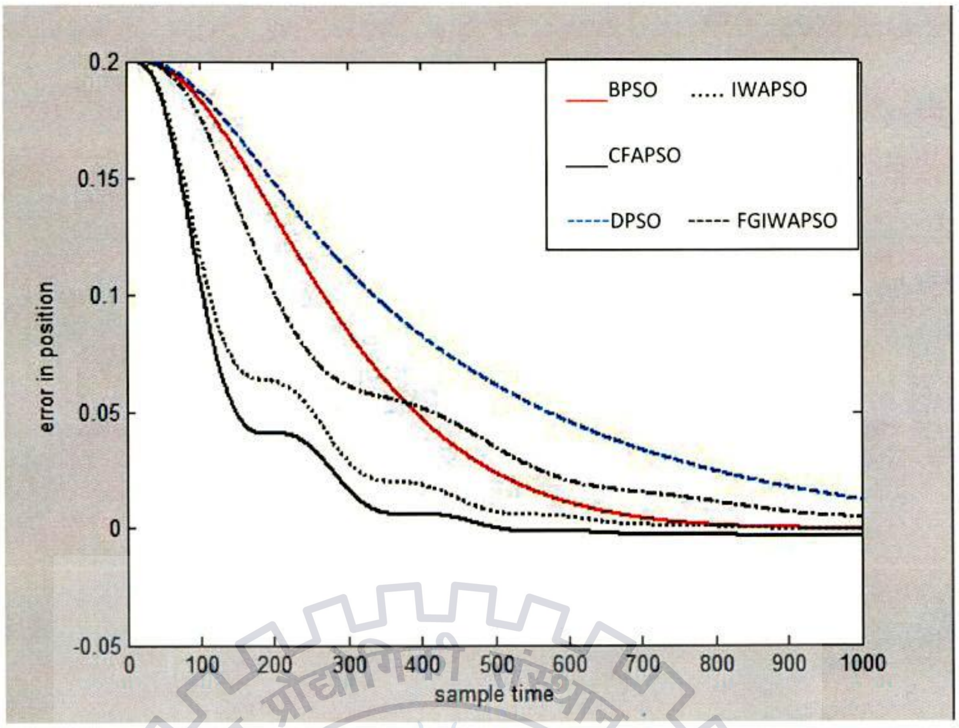


Fig.6.7 Error convergence after 20 iterations

The figure 6.7 represents error convergence for the PSO variants obtained after 20 iterations. CFAPSO variant tends to reduce the error in position very quickly. The DPSO has the maximum error. BPSO takes some time but the steady state error converges to zero. The other two variants IWAPSO and FGIWAPSO have intermediate error profile.

The second technique employed to obtain the parameter values is Differential Evolution. The evolutionary algorithm goes for the selection of the better individuals from the lot and ultimately comes out with the best one. The simulation has been done similarly as was done in case of PSO i.e. both the PID controllers are monitored with the help of DE for a given number of iterations[23]. Five variants have been used for this particular purpose.

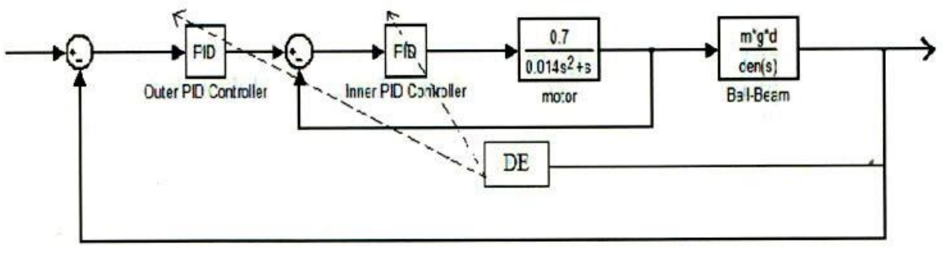


Fig.6.8 DE optimized PID controllers

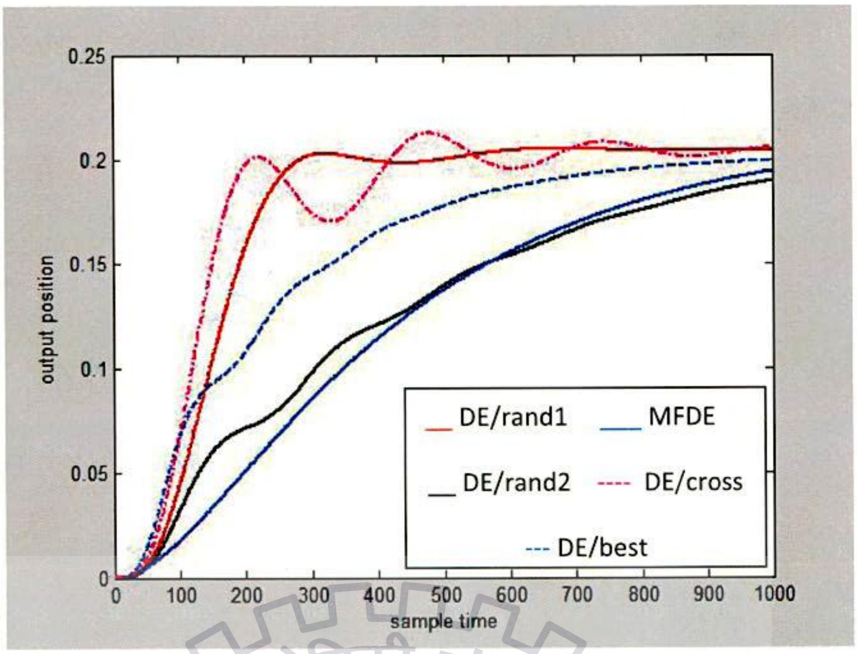


Fig.6.9 Output position convergence after 20 iterations

Figure 6.9 shows for 20 iterations DE/rand1 and DE/cross variants tend to follow closely. Out of the two DE/rand1 shows the smooth characteristic whereas DE/cross /has overshoots and undershoots in order to reach for the target position. DE/best shows the intermediate response. The other two MFDE and DE/rand2 show poor responses.

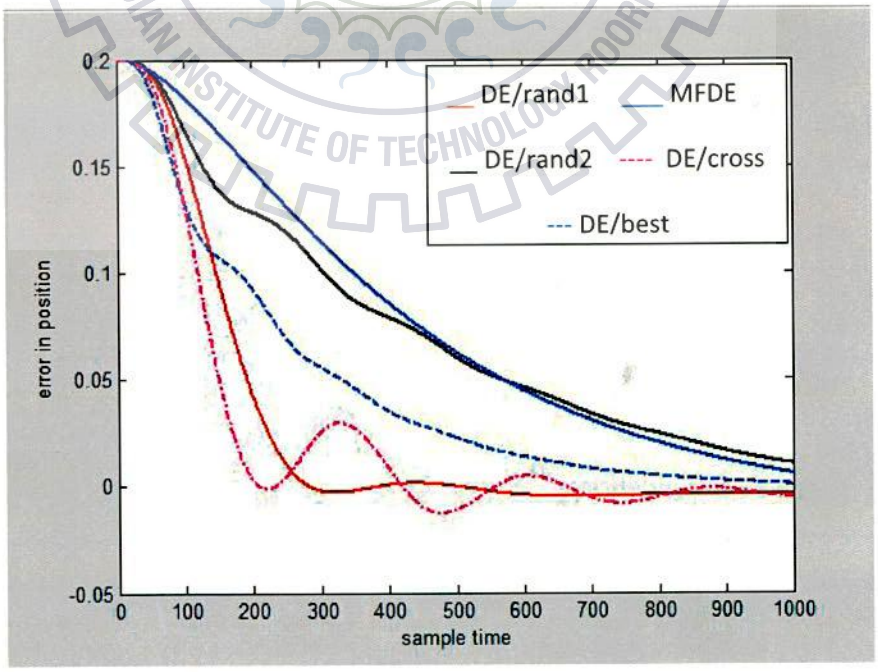


Fig.6.10 Error convergence after 20 iterations

As per the error profile presented in figure 6.10 the DE/rand1 and DE/cross show lesser error whereas MFDE and DE/rand2 show large error. Hence, the parameter values optimized by the DE variant DE/rand1 should be used.

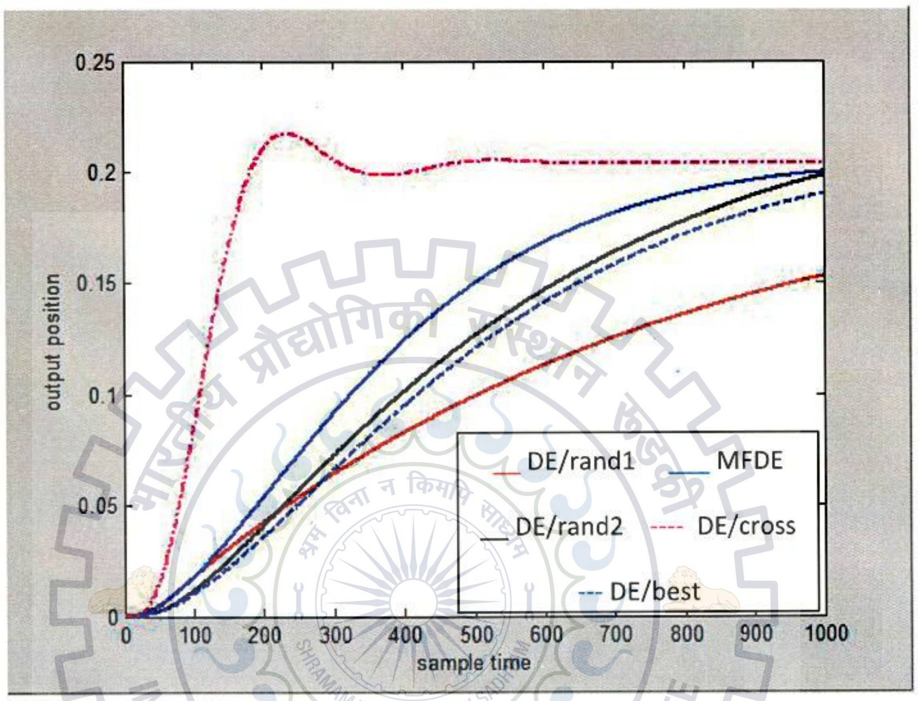


Fig.6.11 Output position convergence after 5 iterations

The number of iterations is reduced to 5. Figure 6.11 shows the output position of the ball for different DE variants. For lesser number of iterations DE/cross stands out amongst the others. It helps the system to reach the desired position very quickly. It also shows a good transient and steady state performance. DE/rand1 takes large time to settle at the required position.

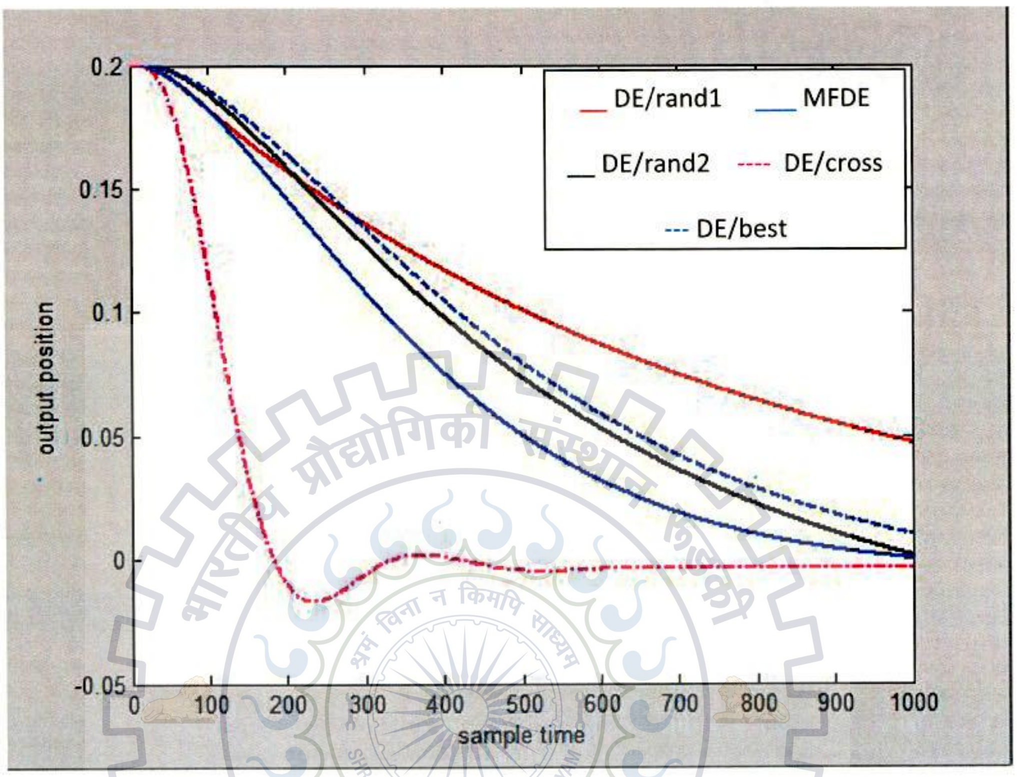


Fig.6.12 Error convergence after 5 iterations

The error profile for different DE variants for less number of iterations i.e.5 is shown in figure 6.12. As per the figure DE/cross has the least error and takes very little time to reduce the error near to zero. DE/rand1 takes the largest time for steady state error convergence. The other three have intermediate error profile.

Thus, for the same evolutionary algorithm the variants behave differently. It can be thereby inferred that for lesser iterations DE/cross suited the best whereas for the larger iterations it was DE/rand1.

Different DE trials are considered to optimize the parameters of inner as well as outer PID controllers. The number of iterations considered here are 2000. Results of five best trials have been put here and their corresponding responses are shown in figure 6.13.

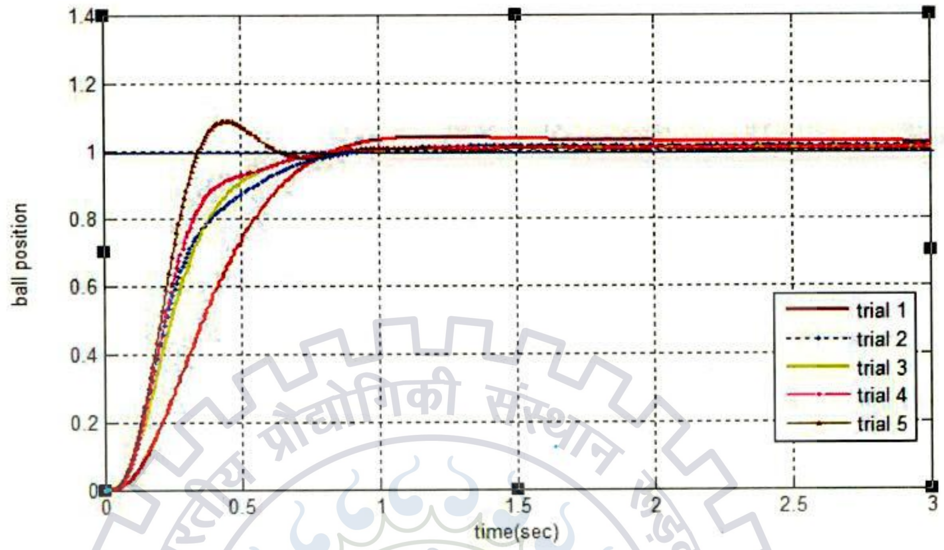


Fig.6.13 Step response of DE trials

The overshoots and rise time of different DE trials given in figure 6.14 and 6.15 are for the above step responses.

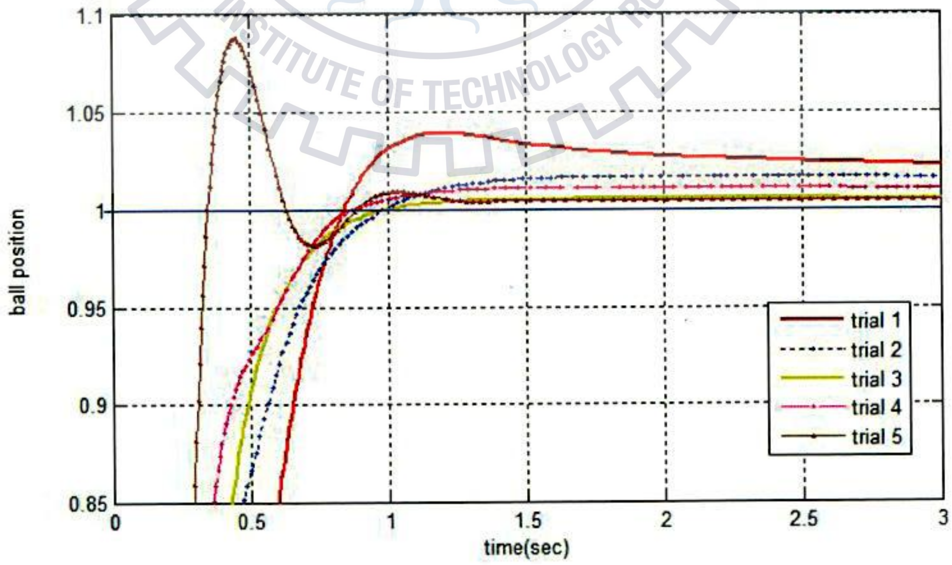


Fig.6.14 Overshoots of DE trials

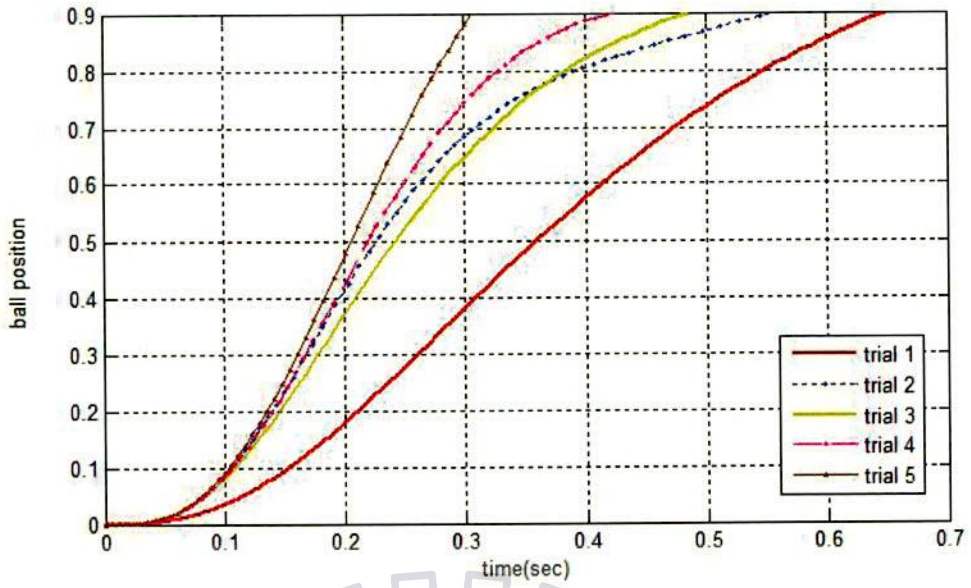


Fig.6.15 Rise time of DE trials

The trial 3 gives the most optimized solution for our system as per the response specifications comparison in Table IV and the graph in figure 6.15. It has the least peak overshoot and the rise time and settling time are pretty small enough. Though the ISE is larger than other trials and steady-state error value being the moderate one, is still very small.

Table 6.3. PID parameters for DE trials

	Outer PID			**Inner PID**		
	K_P	K_D	K_I	K_P	K_D	K_I
Trial 1	14.4759	2.8703	0.4334	13.5289	0.1847	0.9047
Trial 2	14.1719	3.7150	0.7572	12.5373	0.3804	1.1356
Trial 3	14.9568	3.4404	0.1143	13.3117	0.6596	0.8455
Trial 4	14.0607	3.1671	0.4209	12.2073	0.4230	1.5602
Trial 5	14.4135	2.1280	0.1042	14.0914	0.6751	1.0221

Table 6.4. DE trials specifications

	trial 1	**trial 2**	**trial 3**	**trial 4**	**trial 5**
Rise time	0.4934	0.5396	0.3594	0.3132	0.2066
Maximum overshoot(%)	4	1.6	0.4	1.2	8.8
Steady-state error	2.8197e-04	5.4334e-05	8.0875e-04	3.2310e-04	5.0299e-04
ISE	0.1491	0.1955	0.2001	0.1881	0.1817
Settling time	0.7198	0.6726	0.5924	0.6000	0.3264

The DE variants are considered here for optimizing the parameters for 2000 iterations. The four of the variants used are DE/rand1, DE/MFDE, DE/CR, DE/best. This is shown in figure 6.16.

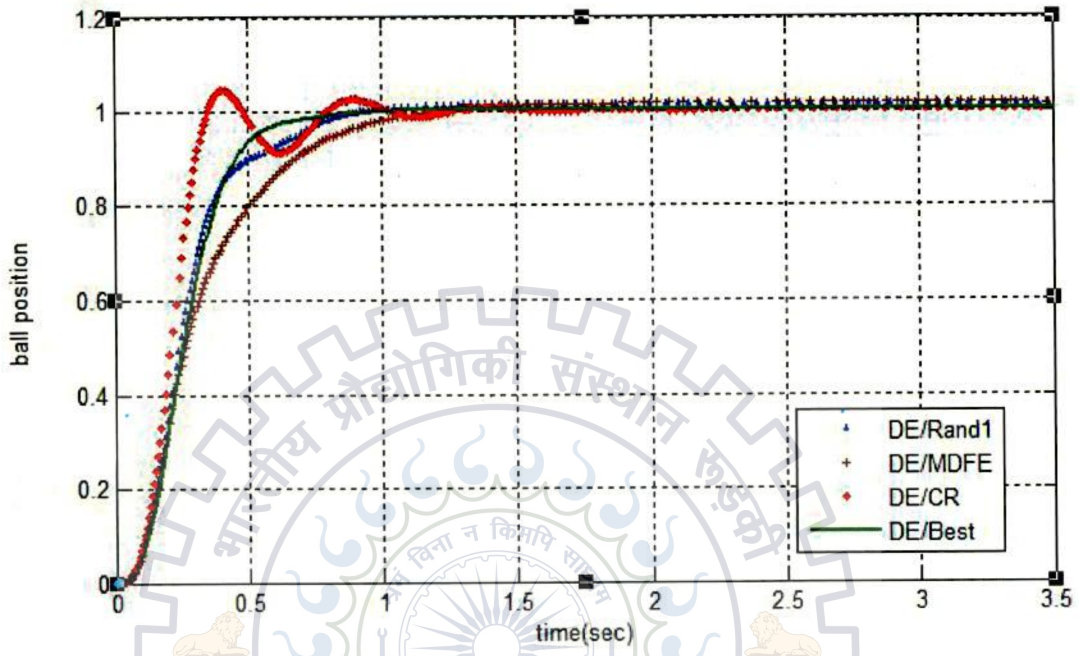


Fig.6.16 Step response of DE variants

Table 6.5. PID parameters for DE variants

	Outer PID			Inner PID		
	K_P	K_D	K_I	K_P	K_D	K_I
DE/Rand1	13.1522	2.6097	0.9081	14.4458	0.7187	1.6445
DE/MFDE	14.5089	4.6638	0.0600	13.6675	0.6312	0.7101
DE/Best	14.9568	3.4404	0.1143	13.3117	0.6596	0.8455
DE/Cross	10.5571	2.7636	0.1341	8.7904	0.0671	0.7206

Table 6.6. Comparison of DE variants

	DE/Rand1	DE/MFDE	DE/Best	DE/Cross
Rise time	0.38	0.56	0.35	0.19
Peak overshoot(%)	2.1	2.0	0.4	4.5
Steady-state error	6.2401e-06	4.1673e-04	8.0875e-04	5.1788e-05
ISE value	0.2102	0.2341	0.2001	0.1869
Settling time	0.68	0.82	0.59	0.33

Of the four DE variants DE/Best has the least peak overshoot and comparatively lesser rise and settling time as per the performance specifications in Table VI, thus stands out better considering the performance of the system.

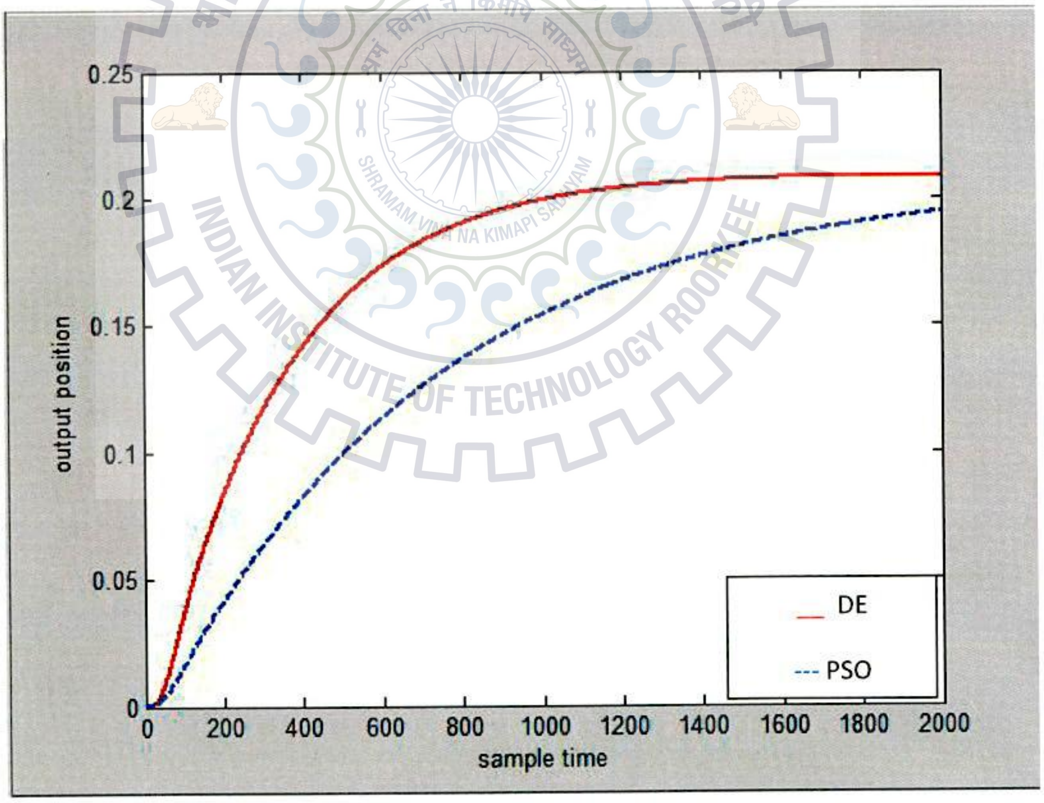


Fig.6.17 Output position convergence after 20 iterations

To obtain a comparative study on performance of DE and PSO, the variant used for DE is DE/rand1 whereas for PSO it is BPSO. The output position curves for the two techniques show that the convergence rate for DE is higher as compared to that of PSO. The rise time as well as settling time is far smaller for the DE as compared to PSO. The DE variant helps the system to come to the required position very quickly in contrast to PSO variant.

Thus an inference can be drawn out from the above results based on the simulation performed. It shows that for ball and beam control system DE works better than PSO.

6.2 Robot Manipulator

The model has been developed following the Armstrong et.al parameters[24]. The block diagram of manipulator PUMA 560 is shown in Appendix. The model has a computed torque controller (CTC). The input to the system is provided in terms of joint angles. The resulting joint angles are found with the help of 6-DOF manipulator kinematics. The three joint angles are provided as input whereas the other three joint angles are restricted to zero[25]. The computed torque controller regulates the torque considering the error in joint angles, their derivatives and double derivatives. Computation involves the inertial quantities, the coriolis and centrifugal forces and also the gravity effects. The controller has two gain parameters K_P and K_D. These controller gains are generally tuned by hit and trial in order to obtain the desired joint angles. Here the tuning is being done by using PSO and DE. The simulation undergoes certain iterations to follow up with best gain parameter values [27].

Input at joint 1 = 1.sin(0.0628t) degree

Input at joint 2 = 2.sin(0.0628t) degree

Input at joint 3 = 3.sin(0.0628t) degree

The fitness function considered is Integral Square of error (ISE).

$$\text{ISE} = \int_0^{t_f} e_1^2 + e_2^2 + e_3^2 \, dt \tag{6.4}$$

e_1, e_2, e_3 : the errors of joint angles 1, 2 and 3 respectively.

t_f : the time required for the simulation.

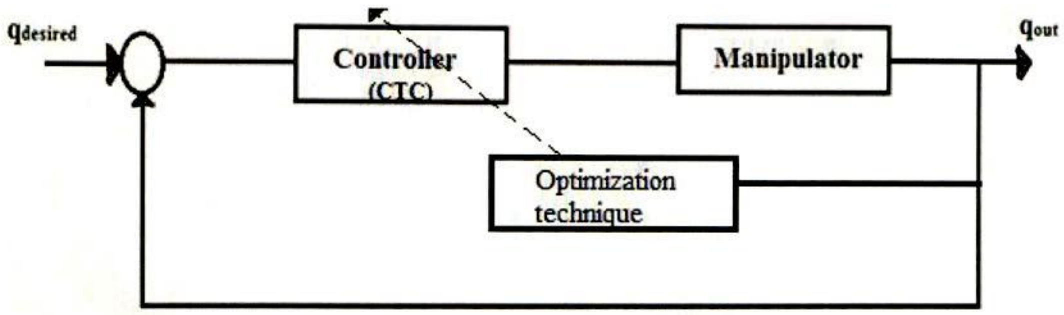


Fig.6.18 Optimized Controller Block diagram

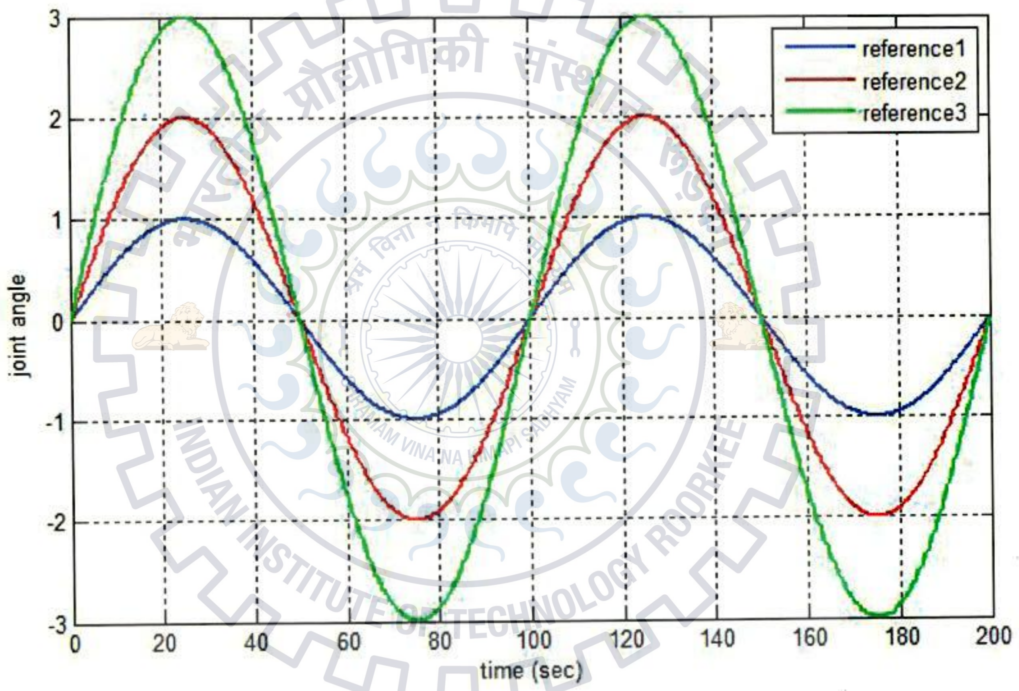


Fig.6.19 Reference signals

The figure 6.19 represents the reference signals for all the three joint angles. These input signals are considered for all the simulation. The signals are sinusoidal in nature and they vary in magnitude but have no phase difference.

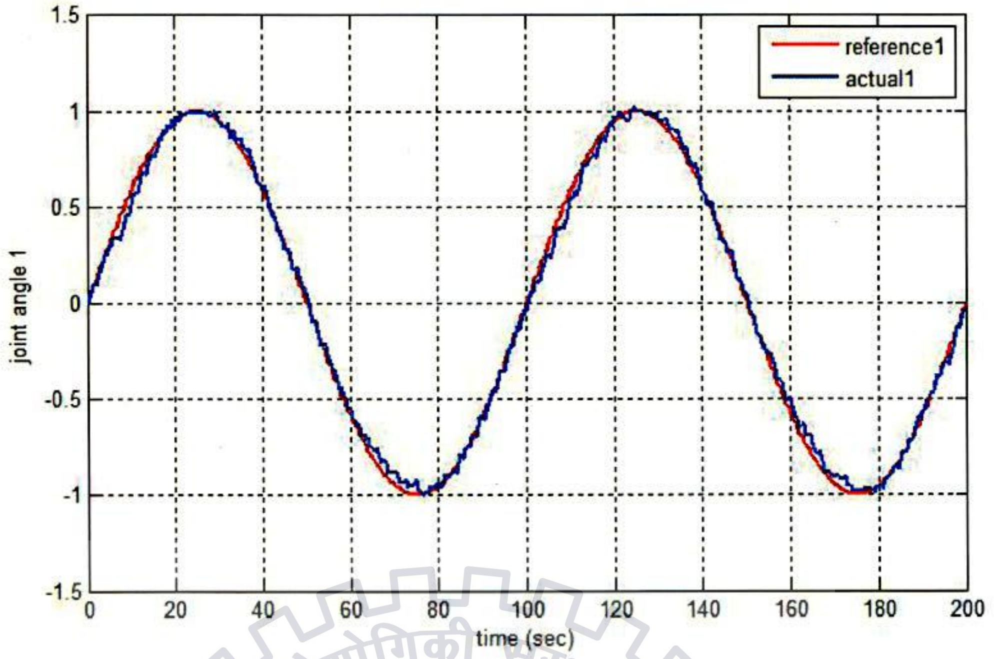


Fig6.20. Trajectory of joint angle 1

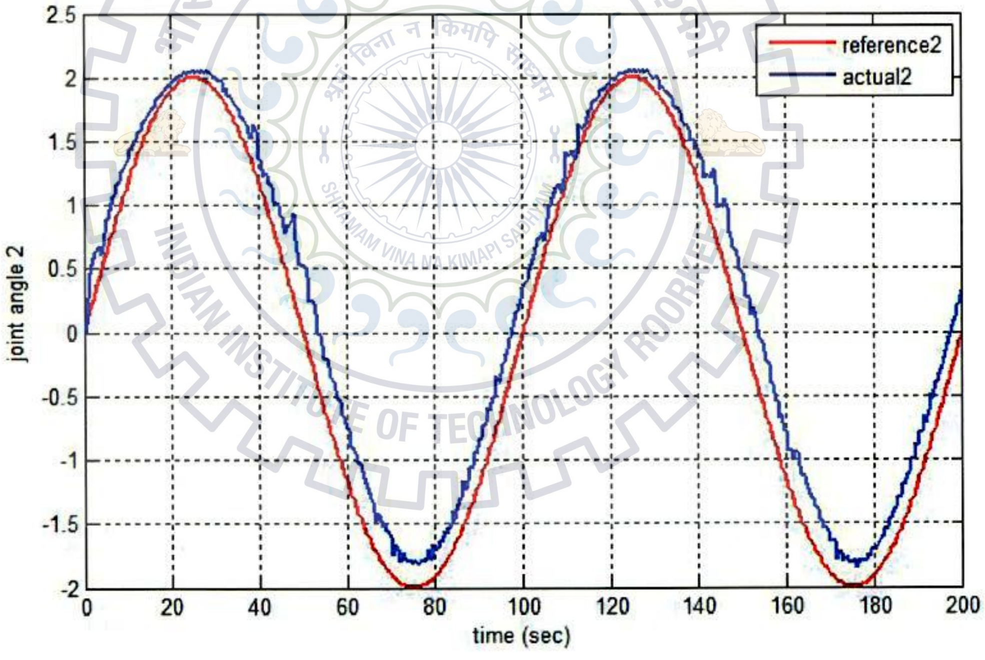


Fig.6.21 Trajectory of joint angle 2

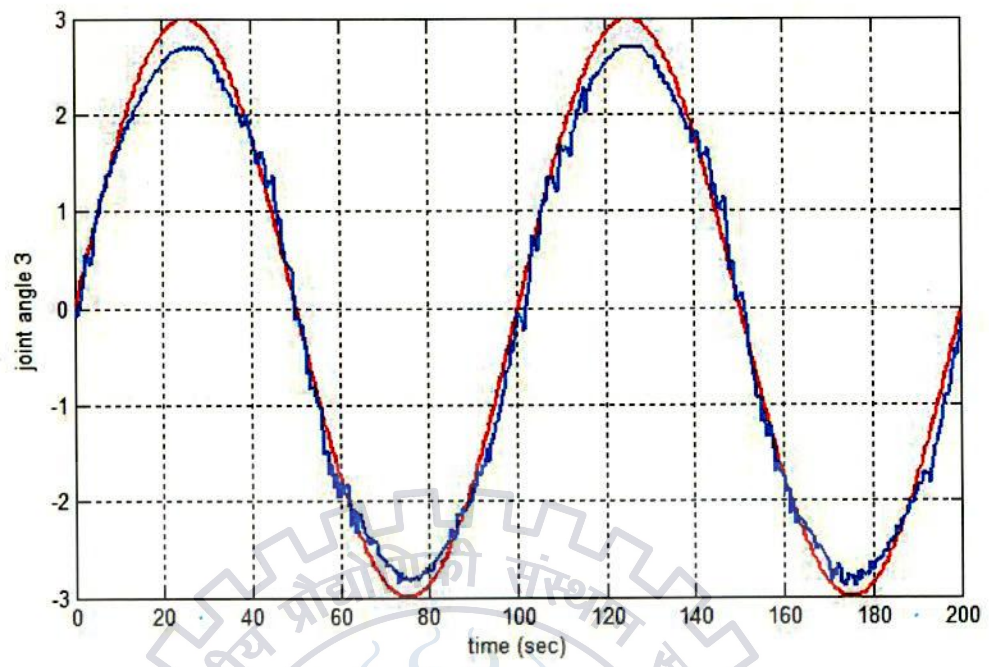


Fig.6.22 Trajectory of joint angle 3

The trajectory tracking by the joint angles has been obtained by deploying DE algorithm. The population vector numbers 50 and the maximum number of generations considered is 500. The simulation has been carried out number of times. The best of all the trials has been used as the tuned values for controller parameters. The best trials are judged on the basis of objective function values. The best one has been considered based on the minimum objective function value. The figures 6.21, 6.22 and 6.23 show the trajectory followed by each of the joint angles. The trajectory of the joint 1 angle has been followed very closely whereas for the other two joint angles they have drifted a little bit away.

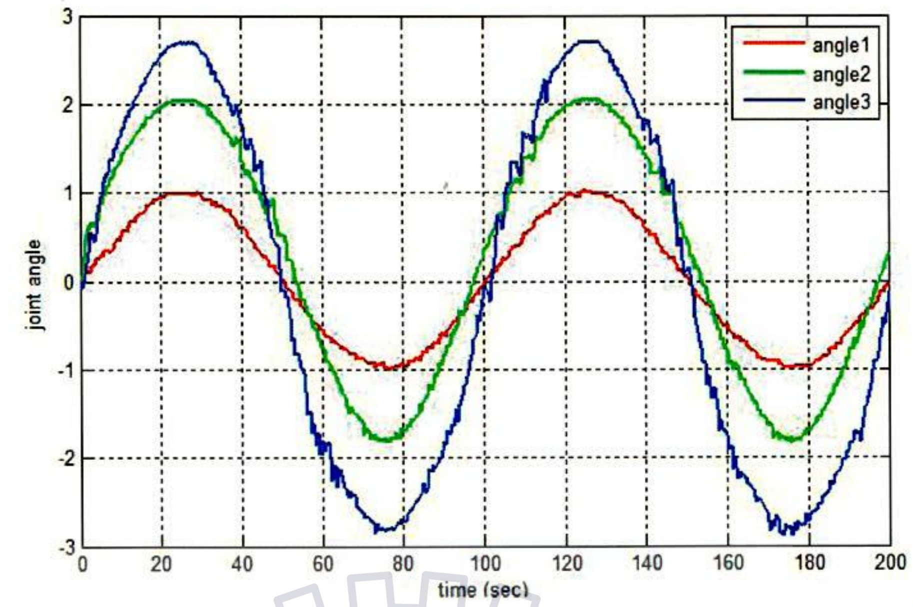


Fig.6.23 Trajectory of all joint angles

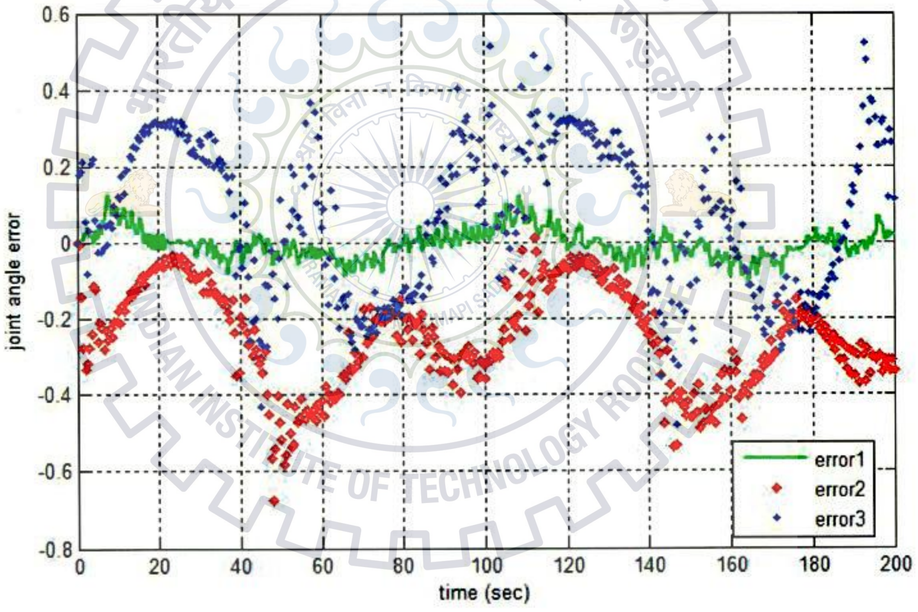


Fig.6.24 Error profile of all joint angles

The error profile for all the joint angles has been shown in figure 6.24. The error in angles is very small for all the joints. They lie in the range from -0.5 to 0.5. Of the three joints the joint 1 has the least error and it is approximately equal to zero all the time. The joint 2 shows the maximum error whereas the joint 3 has error in the intermediate range.

PSO trials :

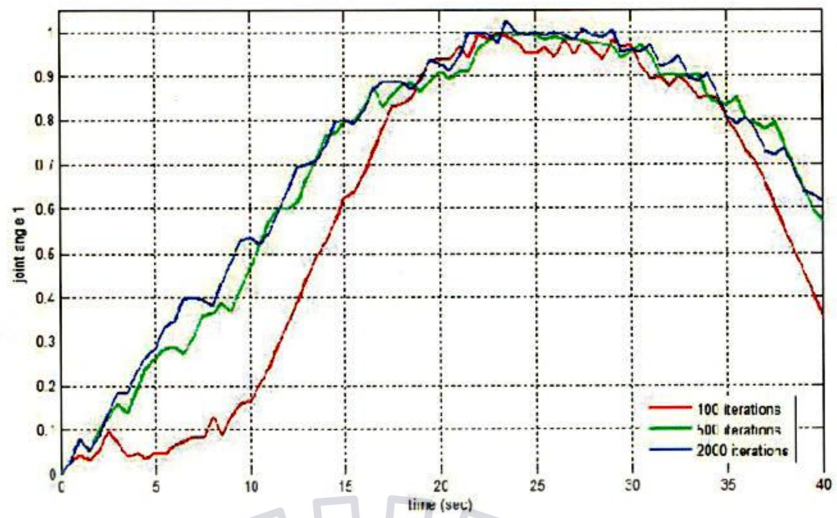


Fig.6.25 Tracking of joint angle for different iterations

Figure 6.25 is based on the iterations carried out using PSO. The profile clearly indicates the difference in the nature of tracking of the joint angle 1 depending on the number of iterations. Greater the iterations better is the tracking of joint angle. The largest number of iterations used is 2000 which gives a good result in the form of tracking.

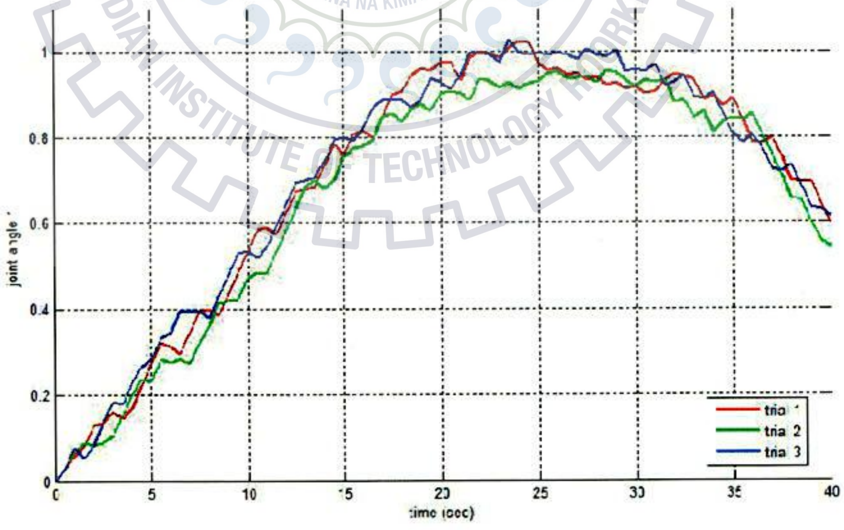


Fig.6.26 Joint angle 1 output of PSO trials

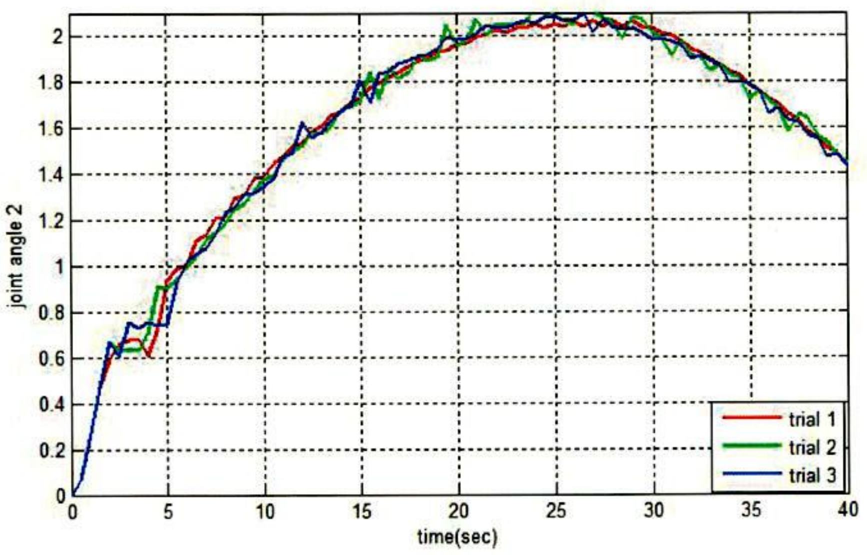


Fig.6.27 Joint angle 2 output of PSO trials

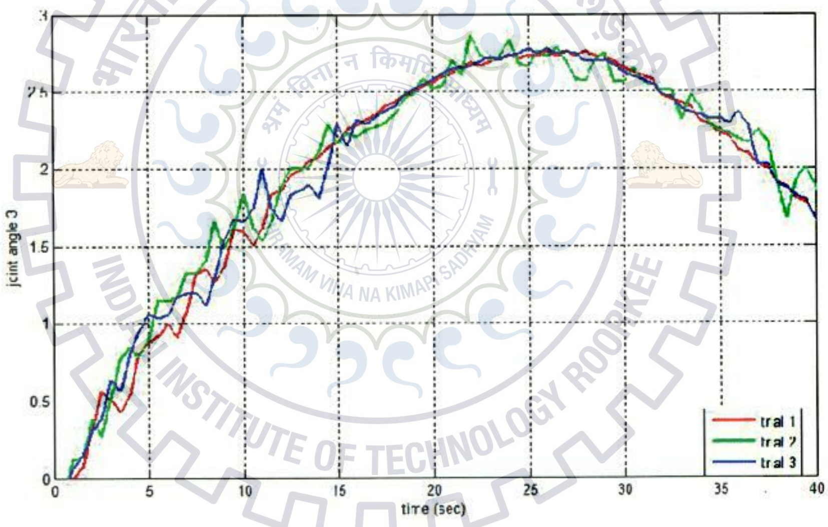


Fig.6.28 Joint angle 3 output of PSO trials

Table 6.7. PID parameters for PSO trials

	K_{P1}	K_{V1}	K_{P2}	K_{V2}	K_{P3}	K_{V3}
trial 1	53.7327	6.9512	22.7958	8.3448	24.1572	3.9336
trial 2	24.9951	8.4353	22.9546	2.5698	22.7492	0.2575
trial 3	50.6630	0.4545	22.7308	3.5669	24.1105	2.7908

Table 6.8. Comparison of PSO trials

	trial 1	trial 2	trial 3
ISE value	0.2706	0.2765	0.2422

The above three figures 6.26, 6.27 and 6.28 are plotted considering three best trials out of different number of trials conducted using PSO for the three joint angles 1, 2 and 3 respectively. The trail 3 is the best among all in terms of tracking of joint angles. The objective function value for the trial 3 is lesser than the other 2 best trials considered.

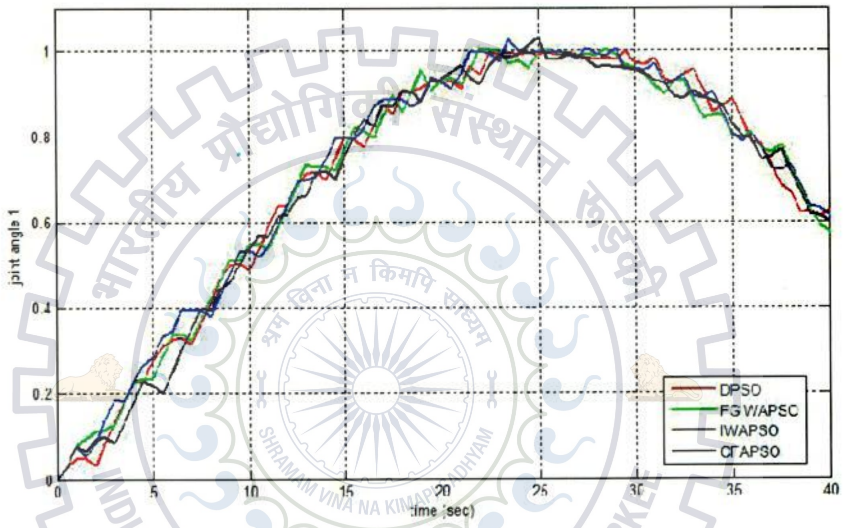


Fig.6.29 Joint angle 1 output of PSO variants

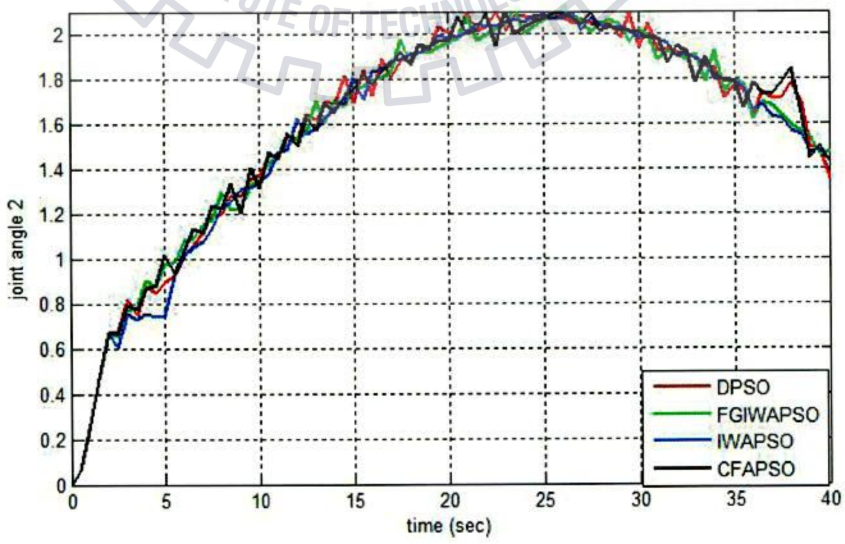


Fig.6.30 Joint angle 2 output of PSO variants

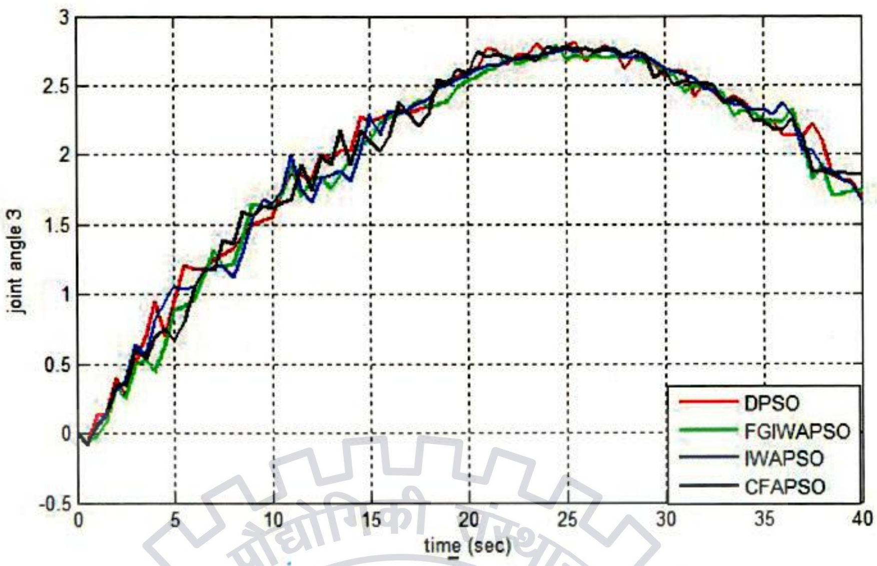


Fig.6.31 Joint angle 3 output of PSO variants

Table 6.9. Comparison of PSO variants

	DPSO	FGIWAPSO	IWAPSO	CFAPSO
ISE value	0.2732	0.2422	0.0667	0.3104

The three figures 6.29, 6.30 and 6.31 are plotted considering the variants of PSO for the three joint angles. The variants are DPSO, FGIWAPSO, IWAPSO and CFAPSO. Considering the CTC parameters as well as objective function values, obtained after certain number of iterations, IWAPSO turns out to be best among them.

DE trials :

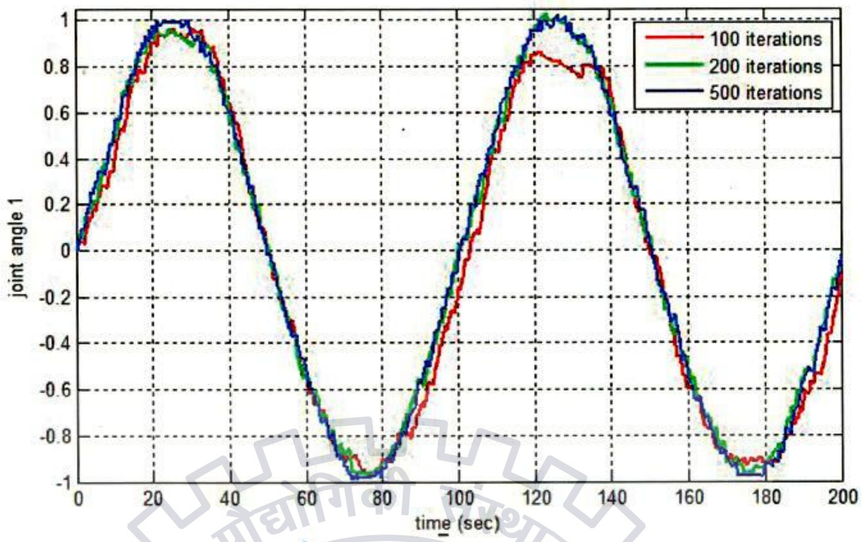


Fig.6.32 Tracking of joint angle for different iterations

Different numbers of iterations are run on the system using the DE algorithm. The tracking profile obtained after different iterations are presented in figure 6.32. It gives a clear picture that greater the number of iterations better is the tracking of joint angles.

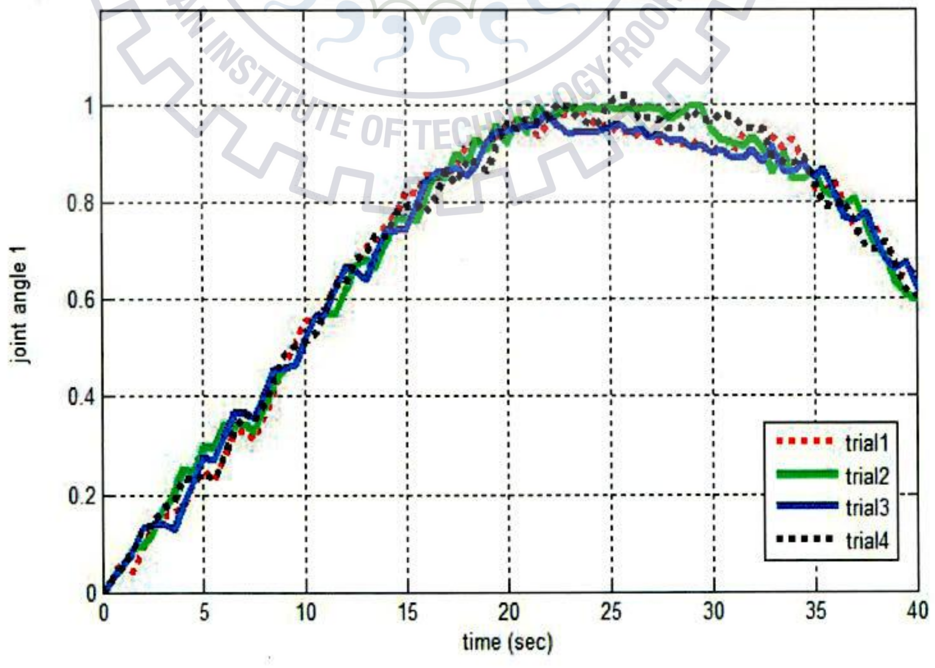


Fig.6.33 Joint angle 1 output of DE trials

The figures 6.25, 6.26 and 6.27 give the joint angle output for the joints 1, 2 and 3 respectively for 4 different DE trials.

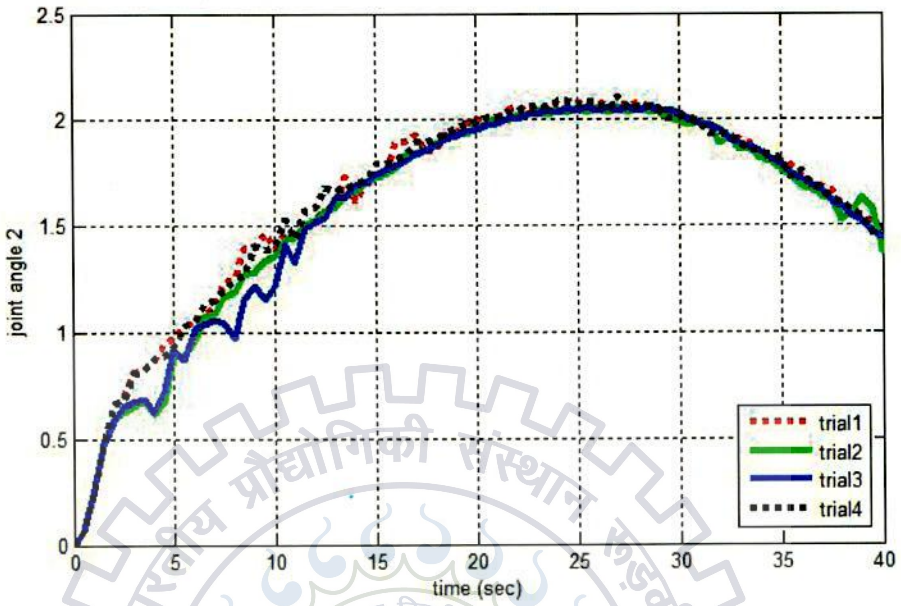


Fig.6.34 Joint angle 2 output DE trials

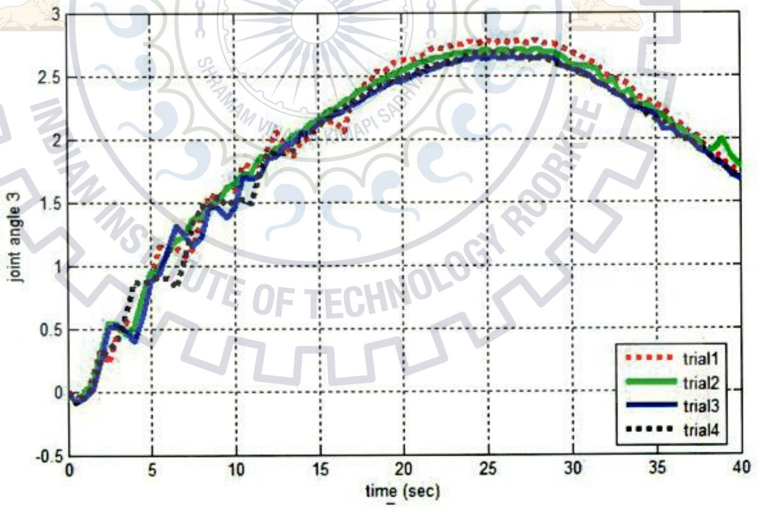


Fig.6.35 Joint angle 3 output of DE trials

Table 6.10 PID parameters for DE trials

	K_{P1}	K_{V1}	K_{P2}	K_{V2}	K_{P3}	K_{V3}
trial 1	40.6204	6.9977	18.4839	5.6677	28.5997	4.0104
trial 2	49.6508	8.6024	25.4249	5.2468	20.8589	3.8529

trial 3	49.4844	7.8203	23.2725	7.9673	15.4985	3.7496
trial 4	49.4701	7.6938	19.0880	5.0659	16.9926	3.5620

ISE values :

Trial 1 : 0.0667

Trial 2 : 0.0569

Trial 3 : 0.0729

Trial 4 : 0.1141

The trial 2 seems the most optimum solution for the controller gains of CTC based on the comparison of the objective function values..The objective function (ISE) value for the trial 2 is 0.0569 which is least among all the trials. Table 6.7 gives the required controller parameters of trial 2.

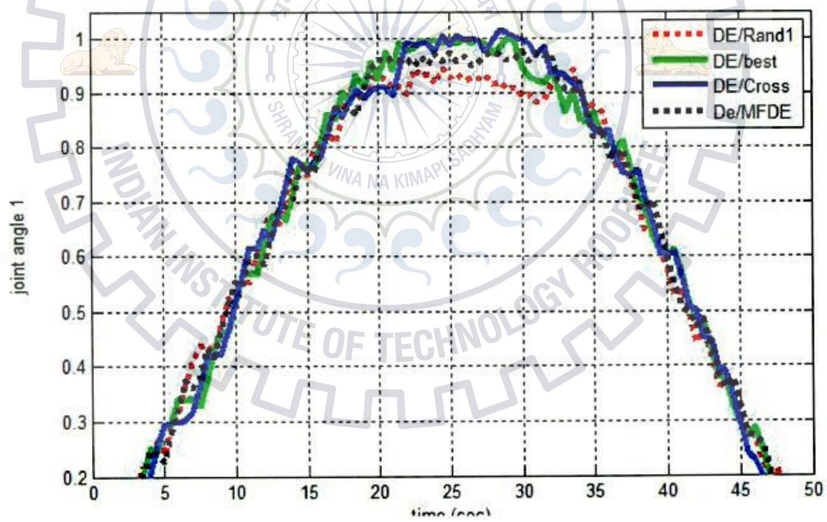


Fig.6.36 Joint angle 1 output of DE variants

Table 6.11 PID parameters for DE variants

	K_{P1}	K_{V1}	K_{P2}	K_{V2}	K_{P3}	K_{V3}
DE/Rand1	44.4070	8.0802	49.2815	6.6430	31.3098	3.1743
DE/MFDE	51.8096	9.6980	48.0670	4.0461	33.3116	3.5525
DE/Best	49.6508	8.6024	25.4249	5.2468	20.8589	3.8529

DE/Cross	52.8304	6.4656	49.1411	4.8271	32.9930	4.3382

Table 6.12 Comparison of DE variants

	DE/Rand1	DE/MFDE	DE/Best	DE/Cross
ISE value	0.1314	0.1374	0.0667	0.1293

Of the four DE variants DE/Rand1, DE/MFDE, DE/Best and DE/Cross, DE/Best has the least objective function value as per the performance specifications (ISE value) in Table 6.9. It thereby stands out best considering the performance of the robot manipulator. The parameters for the joint angle controllers can be used from the Table 6.8.

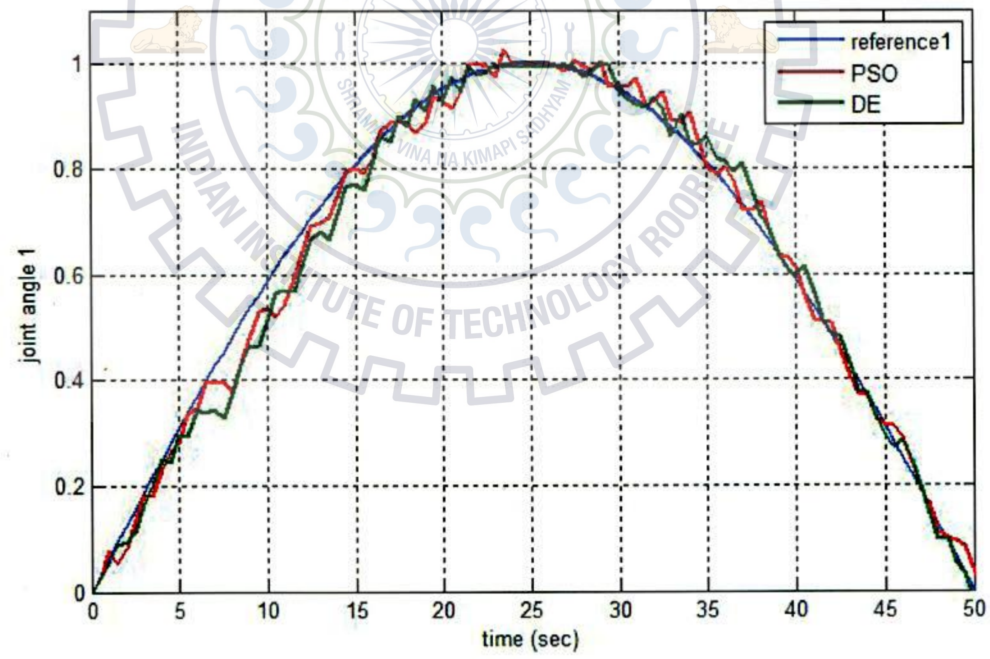


Fig.6.37 Joint angle 1trajectory

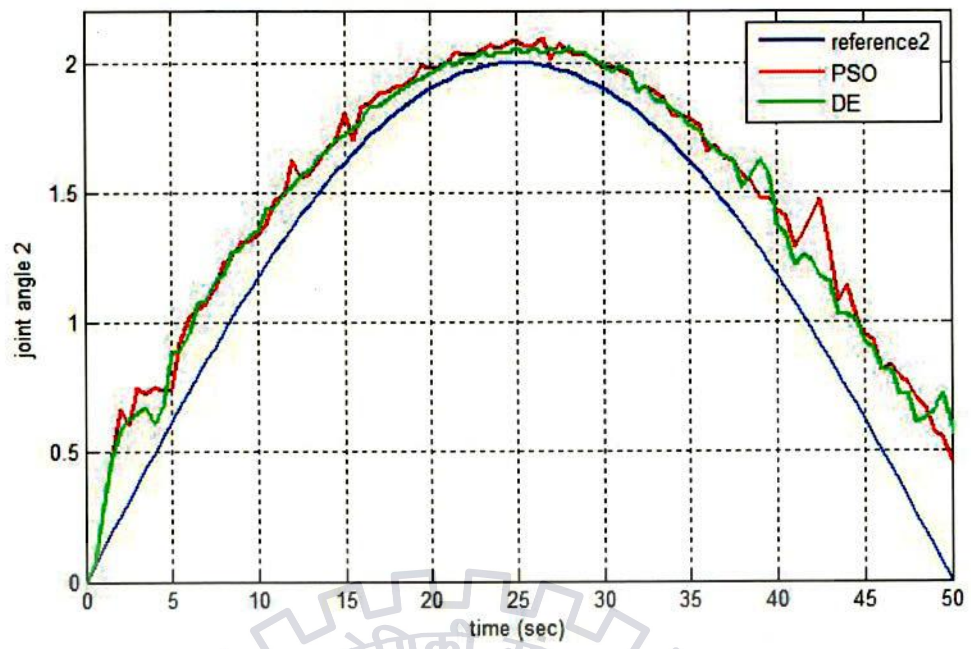


Fig.6.38 Joint angle 2 trajectory

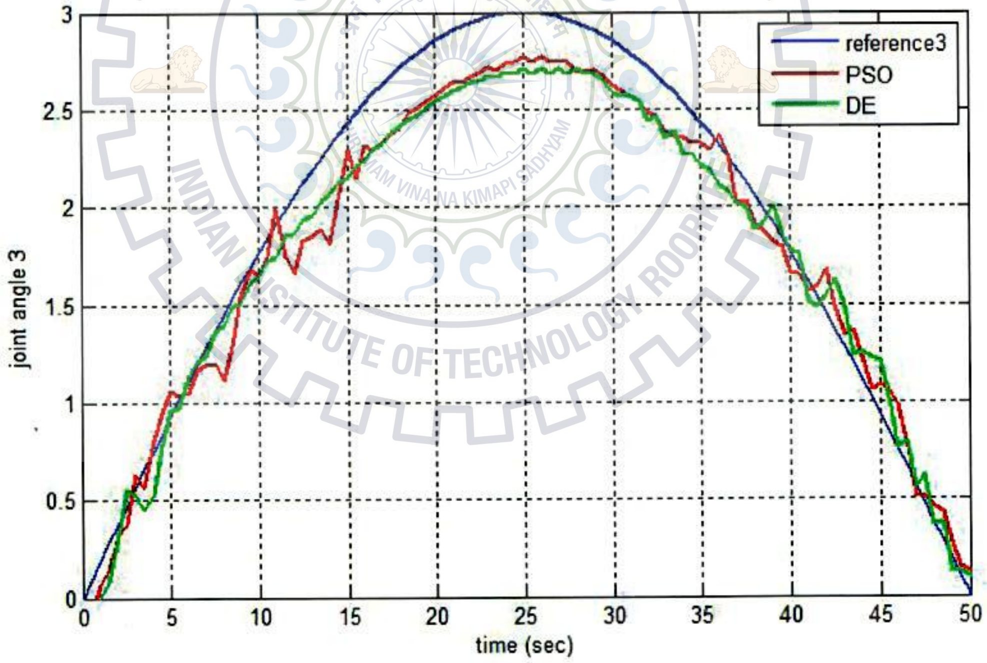


Fig.6.39 Joint angle 3 trajectory

Table 6.13 CTC parameters for PSO and DE

	K_{P1}	K_{V1}	K_{P2}	K_{V2}	K_{P3}	K_{V3}
PSO	50.6630	6.4545	22.7303	3.5669	24.1105	2.7098
DE	49.6508	8.6024	25.4249	5.2468	20.8589	3.8529

Table 6.14 Comparison of PSO and DE

	PSO	**DE**
ISE value	0.2429	0.0667

The figures 6.29, 6.30 and 6.31 are obtained after simulation to draw out the convergence of both the techniques. As earlier, the trajectory followed in case of joint 1 is equal to that of the reference angle whereas same is not the case for joint 2 and 3.The trajectory followed deviates from the reference for both the techniques. Still the DE shows better follow up of trajectory as compared to that of PSO.

Another striking difference is the number of iterations used for the convergence. The DE uses lesser number of iterations to optimize the parameters and hence for the convergence whereas PSO uses larger iterations. Also the objective function values show difference in both the techniques. The minimised objective function value is very small for DE but for PSO it is larger in magnitude.

CHAPTER – 7

CONCLUSION AND FUTURE SUGGESTIONS

The dissertation aimed at application of soft computing techniques to two control systems. Both control systems are highly coupled non-linear device [23] involving complex time variable dynamics being. Two soft computing techniques are used for the fine tuning of controller parameters. One of them is a search based heuristic technique i.e. Particle Swarm Optimization (PSO) and the other is a population based generation technique known as Differential Evolution (DE). Both the techniques have different nature and pertain to different algorithm.

The systems are simulated with the tuned parameters and the outputs are thus well regulated. Both the soft computing techniques give the desired output. The two techniques are similar to the point that they are iteration based and consider a population of likely individuals for the solution. The difference lies in their operation based on respective algorithms. It is found that Differential Evolution (DE) is better than Particle Swarm Optimization (PSO) in terms of convergence to the optimal values. The DE converges rapidly as compared to PSO. The number of iterations considered for DE is far smaller than PSO. In DE the number of iterations required was 500 whereas it ranged to 2000 for PSO. The objective function values are also fairly minimized by DE. Hence, DE suits better than PSO for systems modelled in this dissertation.

The soft computing techniques are not restricted by any mathematical formulation. They are flexible by nature. Different variants have been developed to make them much more adaptive towards the systems. There exist different controllers and techniques which can be applied on these systems. A better convergence, accuracy and tolerance can be looked out for using different techniques and methodologies apart from the above mentioned ones. Thus, future holds fair research opportunities in this area.

REFERENCES

[1] Yasuhiko Dote and Seppo J. Ovaska, "Industrial Applications of Soft Computing: A Review", *Proc.of the IEEE,* vol. 89, no. 9, pp.1243-1254, 2001.

[2] Matthew Settles, "Introduction to Particle Swarm Optimization", *Proc. IEEE Int. Conf. Evol. Comput., 2005.*

[3] David Ardia, Kris Boudt, Peter Carl, Katherine M. Mullen and Brian G. Peterson, "Differential evolution with DEoptim – An Application to Non-Convex Portfolio Optimization" *The R Journal,* vol. 3/11, June2011, pp.27-34.

[4] Storn, R., Price, K., "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization* ,vol. 11, 1997, pp.341-359.

[5] Ji-Pyng Chiou, Feng-Sheng Wang, "A Hybrid Method of Differential Evolution with Application to Optimal Control Problems of a Bioprocess System," *IEEE International Conference on Evolutionary Computation Proceedings, 1998,* pp.627-632.

[6] LI Gao-yang, LIU Ming-guang, "The summary of Differential Evolution Algorithm and its Improvements," *3rd International Conference on Advanced Computer Theory and Engineering(ICACTE) IEEE* ,vol. 3, 2010, pp.153-156.

[7] Ching-Wei Chien, Zhan-Rong Hsu,Wei-Ping Lee,"Improving the performance of Differential Evolution algorithm with modified mutation factor," *International Conference on Machine Learning and Computing IPCSIT,* vol.3,2011, pp.64-69.

[8] R.Storn,"Differential Evolution research- trends and open questions,"*Advances in Differential Evolution,* vol.143,Heidelberg: Springer Berlin,2008,pp. 1-31.

[9] Jakob Vesterstrom, Rene Thomsen, "A Comparative study of Differential evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems," *Danish Research Council.*

[10] R.C.Eberhart and Y.Shi,"Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization," *Proc.IEEE Congress on Evolutionary Computation,* vol.1,2000, pp. 84-88.

[11] Clerc M., "Towards a deterministic and adaptive particle swarm optimization", *Cong. On Evolutionary Computation(CEC99)* ,pp. 1951-1959, 1999.

[12] Zhi-Hui Zhan, Jun Zhang, Yun Li and Henry Shu-Hung Chung, "Adaptive Particle Swarm Optimization", *IEEE Trans. On Systems, Man, and Cybernatics-Part B:Cybernatics,* vol. 39,no. 6, 2009

[13] Thomas Schoene,Simone A.Ludwig and Raymond J.Spiteri,"Step-Optimized Particle Swarm Optimization,"*WCCI IEEE World Congress on Computational Intelligence,*2012 .

[14] Cockburn J.C. and Savakis A., "DC Motor Transfer Function,"*http://www.ce.rit.edu/~cockburn/courses/ce553_su02/labs/lab3_r1.pdf.*

[15] Rosales E. A., "A Ball-on-Beam Project Kit"*ADepartment of Mechanical Engineering, Massachusetts Institute of Technology,* June 2004.

[16] Hirsch R, "Mechatronic Instructional Systems Ball on Beam System",*dShandor Motion Systems, http://www.ro.feri.uni_mb.si/predmeti/skup_sem/projekt1/shandor.pdf,* 1999.

[17] Wellstead P., "Ball and Beam I: Basics"*Ahttp://www.controlsystemsprinciples.co.uk/whitepapers/ball-and-beam.*

[18] Rachid Manseur, "Robot Modeling and Kinematics",*ADa Vnchi Engineering Press,*First Edition,2006.

[19] Nevzat Ozturk, Behruz Fardanesh, „"Controller Design for a Manipulator in the presence of Time Delay," *IEEE,*1990.

[20] Farzin Piltan, Mohammad Hossein, Mohammad Shamsodini, Ebrahim Mazlomian & Ali Hosainpur, „"PUMA 560 Robot Manipulator Position Computed Torque Control Methods Using MATLAB/SIMULINK," *International Journal of Robotics and Automation,(IJRA),*vol.3,issue.3,2012.

[21] Alireza Izadbakhsh,."Closed Form Dynamic Model of PUMA 560 Robot arm," *Proceedings of the 4th International Conference on Autonomous Robots and Agents,* Wellington, New Zealand, pp.675-680,2009.

[22] Muhammad Asif Rana, Zubair Usman, Zeeshan Shareef, "Automatic Control of Ball and Beam System using Particle SwarmOptimization,"*s12th IEEE International Symposium on Computation Intelligence and Informatics,* 2011,Budapest,Hungary,pp. 529-534.

[23] Epitropakis Michael G.,Vassilis P. Plagianakos, and Michael N. Vrahatis," Finding multiple global optima exploiting differential evolution's niching capability,"*IEEE Symposium on Differential Evolution (SDE)*,2013,pp.1-8.

[24] Brian Armstrong, Oussama Khatib and Joel Burdick,."The Explicit Dynamic Model and Inertial Parameters of PUMA 560 Arm,"*Stanford University, Artificial Intelligence Laboratory,* IEEE,pp.510-518,1986.

[25] T.C.Hsia and G.Z.Lu.,"On Simplification of Robot Arm Dynamic Equation," *IEEE,* 1986.

[26] Mahmoodabadi M.J., A.Bagheri, N.Nariman-Zadeh, A.Jamali, and R.Abedzadeh Maafi,"Pareto Design of Decoupled Sliding-Mode Controllers for Nonlinar Systems Based on a Multiobjective Genetic Algorithm," *Journal of Applied Mathematics,*2012.

APPENDIX

Appendix A : DE and PSO parameters

DE parameters

Number of vector(NP)	50
Dimension(n)	6
Maximum generations	500
Mutation factor(F)	0.5
Crossover Rate(CR)	0.9

PSO parameters

Number of iterations	1000/2000
Dimension(n)	6
Swarm size	50
Inertia_{min}	0.4
Inertia_{max}	0.9
Local Correction factor(c1)	2.0
Global Correction factor(c2)	2.0

Appendix B: PUMA 560 dynamic parameters and Ball and Beam system parameters

B.1 PUMA 560 dynamic parameters

$$\tau = M(\Theta).\ddot{\theta} + V(\Theta,\dot{\theta}).\,\dot{\theta} + G(\theta) \qquad \text{(B.1)}$$

where,

Θ_{nx1} : angular position vector

M_{nxn} : manipulator inertia matix

V_{nx1} : centrifugal and coriolis vector

τ_{nx1} : torque vectors

The above equation is the dynamic form of modelling for the robot manipulator. The term $V(\Theta,\dot{\theta})$ can be decomposed into centrifugal and coriolis terms.

$$\tau = M(\theta).\ddot{\theta} + B(\theta).[\,\dot{\theta}.\dot{\theta}] + C(\theta).[\dot{\theta}^2] + G(\theta) \qquad \text{(B.2)}$$

where,

$B_{nxn(n-1)/2}$: coriolis torque matrix

C_{nxn} : centrifugal torque matrix

$[\dot{\theta}.\,\dot{\theta}]_{n(n-1)/2x1}$: joint angular velocity products vector

$[\dot{\theta}^2]_{nx1}$: joint angular velocity square vector

Only three links of PUMA 560 has been considered. Therefore, the other three links have $\theta_4 = \theta_5 = \theta_6 = 0$.

M is a 6x6 symmetric matrix :

$$M(\theta) = \begin{bmatrix} m_{11} & m_{12} & m_{13} & 0 & 0 & 0 \\ m_{21} & m_{22} & m_{23} & 0 & 0 & 0 \\ m_{31} & m_{32} & m_{33} & 0 & m_{35} & 0 \\ 0 & 0 & 0 & m_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & m_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & m_{66} \end{bmatrix} \tag{B.3}$$

Matrix B is :

$$B(\theta) = \begin{bmatrix} b_{112} & b_{113} & 0 & b_{115} & 0 & b_{123} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b_{214} & 0 & 0 & b_{223} & 0 & b_{225} & 0 & 0 & b_{235} & 0 & 0 & 0 & 0 \\ 0 & 0 & b_{314} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_{412} & b_{413} & 0 & b_{415} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b_{514} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.4}$$

Matrix C is :

$$C(\theta) = \begin{bmatrix} 0 & c_{12} & c_{13} & 0 & 0 & 0 \\ c_{21} & 0 & c_{23} & 0 & 0 & 0 \\ c_{31} & c_{32} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ c_{51} & c_{52} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.5}$$

Matrix G is :

$$G(\theta) = \begin{bmatrix} 0 \\ g_2 \\ g_3 \\ 0 \\ g_4 \\ 0 \end{bmatrix} \tag{B.6}$$

B.2 Ball and Beam system parameters

Mass of the ball, m= 0.028 kg

Radius of the ball, R = 0.01 m

Acceleration due to gravity, g = 9.8 m/s^2

Length of the beam, L = 0.4 m

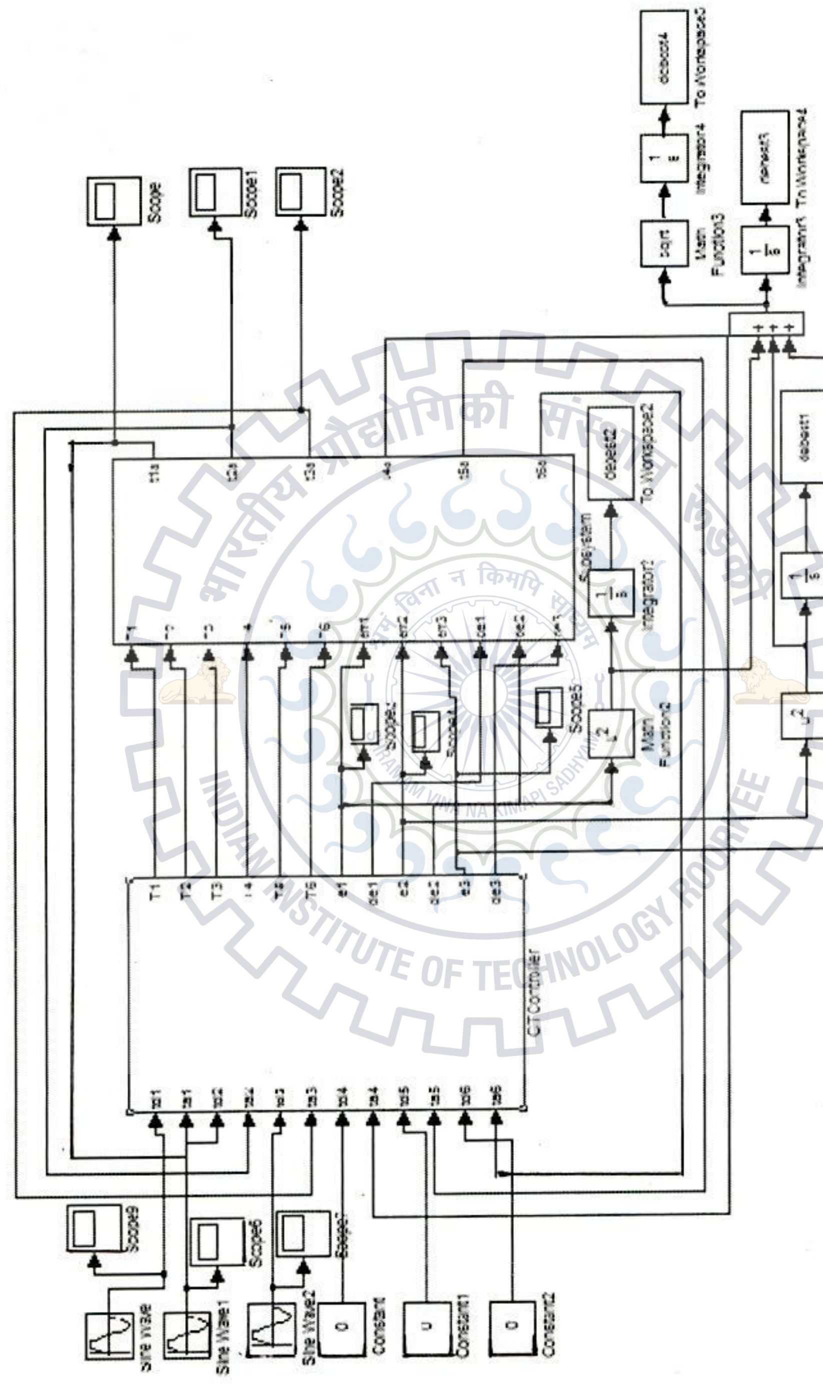
Distance between center of gear and joint of lever arm, d = 0.04 m

The bounds on both the PID parameter values while tuning them are :

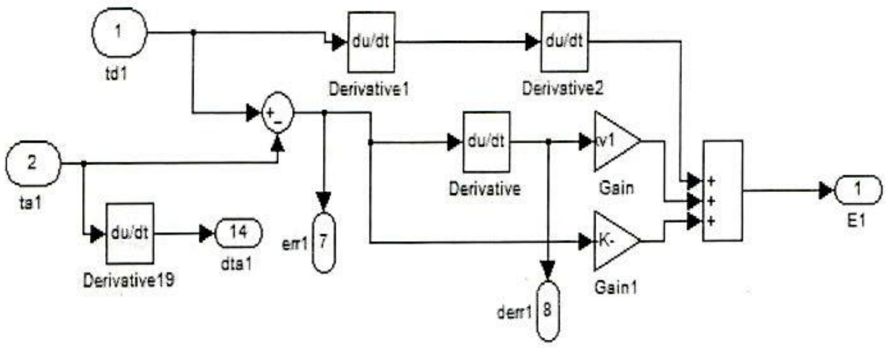
	Outer PID			Inner PID		
	K_P	K_D	K_I	K_P	K_D	K_I
Lower Bound	6	3	0.2	6	1	0.2
Upper Bound	17	9	1.5	17	2	2.5

Appendix C: Simulink model

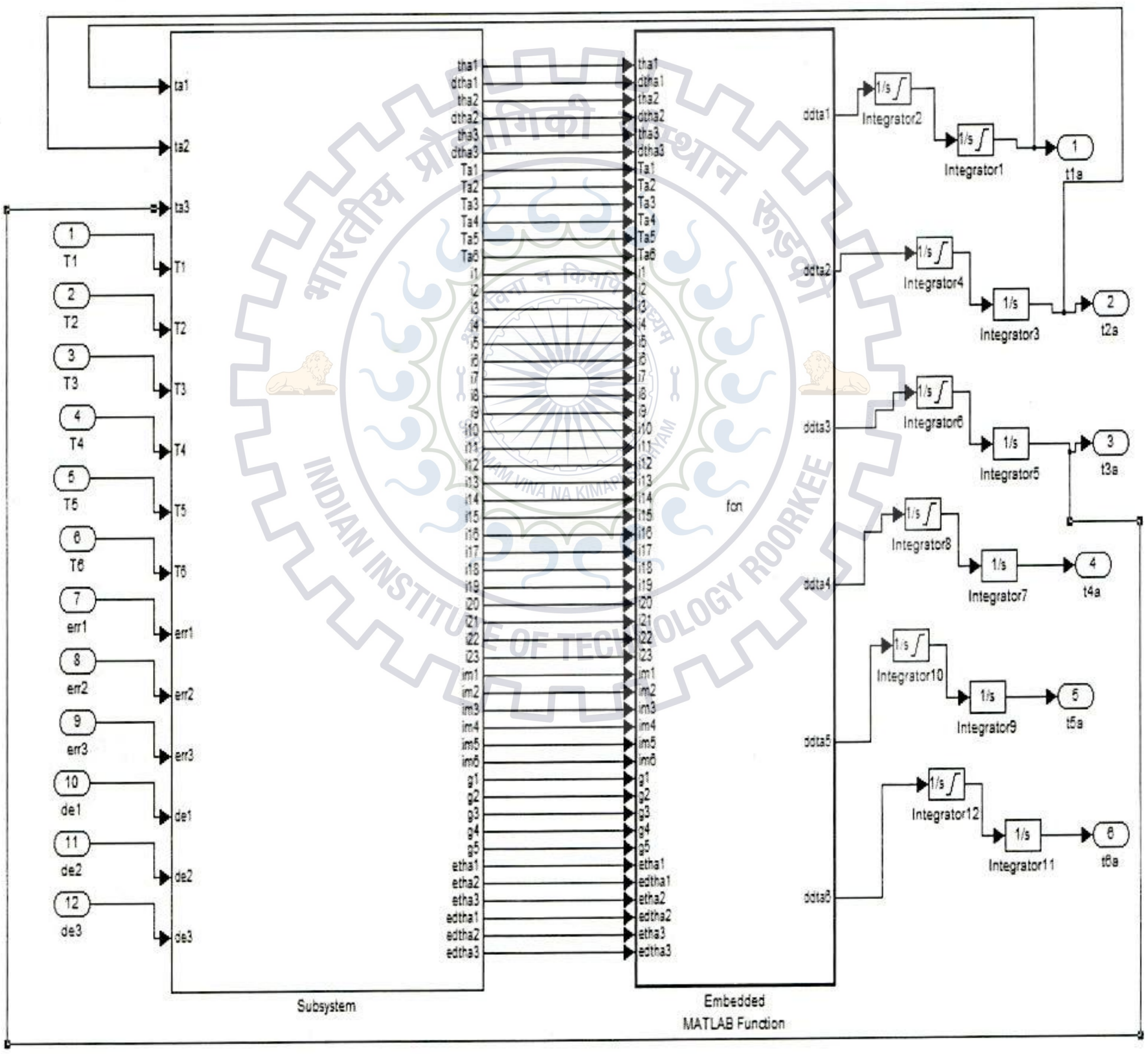
Puma 560 Robot manipulator



A.1 Robot Manipulator with Computed Torque Controller



A.2 PD – CTC for a joint



A.3 Robot Manipulator

Papers published and communicated

1. Narendra Kumar Dhar and M.J.Nigam,“ Controller Parameter tuning using Differential Evolution for automatic control of Ball and Beam system,”*International Conference on Electrical,Electronics and Computer Engineering(ICEECE),*Institute of Research and Journals(IRAJ),2013.

2 Narendra Kumar Dhar and M.J.Nigam,(accepted and to be published)“ Differential Evolution tuned Computed torque controller parameter for trajectory tracking of Robot manipulator,” *SARC-International Conference on Industrial Electronics and Electrical Engineering (SARC-ICIEEE-2013)* ,2013.