TUNING OF PID CONTROLLER BY BIOINSPIRED TECHNIQUES

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology in

Electronics and Instrumentation Engineering

By

BISWAJIT JENA(107EI002)

SAGAR KUMAR (107EI008)

Department of Electronics & Communication Engineering

National Institute of Technology ,Rourkela

2011

TUNING OF PID CONTROLLER BY BIOINSPIRED TECHNIQUES

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology in

Electronics and Instrumentation Engineering

Under the Guidance of

Prof. U.C.PATI

By

BISWAJIT JENA(107EI002)

SAGAR KUMAR (107EI008)

Department of Electronics & Communication Engineering

National Institute of Technology ,Rourkela

2011

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled “**TUNING OF PID CONTROLLER BY BIOINSPIRED TECHNIQUES**”, submitted by Mr BISWAJIT JENA(107EI002) and Mr SAGAR KUMAR (107EI008) in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in ‘ELECTRONICS & INSTRUMENTATION’ Engineering at the National Institute of Technology (NIT), Rourkela is an authentic work carried out by him under my supervision. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university / institute for the award of any Degree or Diploma.

Date: Prof. U.C.PATI

Department of Electronics and Communication Engg.

National Institute of Technology, Rourkela

Rourkela-769008

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude and sincere thanks to my respected supervisor Prof. U.C.PATI for the guidance, insight, and support that he has provided throughout the course of this work. The present work would never have been possible without his vital inputs and mentoring.

I would like to thank all my friends, faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T Rourkela for their extreme help throughout my course of study at this institute.

SAGAR KUMAR (107EI008)

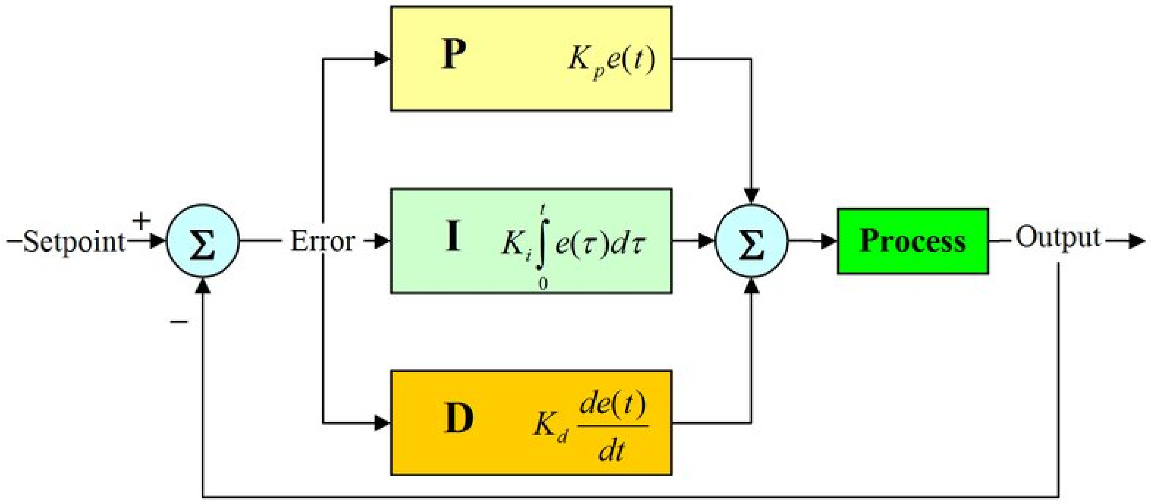
BISWAJIT JENA(107EI002)

CONTENTS

Introduction

PID

Pid is a feedback based controller which gets the error value and calculates the output based on the characteristics of th.e error.it is very widely used in plants as it is simple and gives good result.



Pid is used in aclosed loop .it has three elements P ,I ,D. Every parameter has gain by which we control the contribution.

PID Alogorithm:

$$\mathrm{u}(\mathrm{t}) = \mathrm{MV}(\mathrm{t}) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

where

P_{out}: Proportional term of output

K_p: Proportional gain, a tuning parameter

K_i: Integral gain, a tuning parameter

K_d: Derivative gain, a tuning parameter

e: Error = $SP - PV$

t: Time or instantaneous time (the present)

Proportional term

The proportional term makes a change to the output that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain.

The proportional term is given by:

$$P_{\text{out}} = K_p\, e(t)$$

Derivative term

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_d**.**

The derivative term is given by:

$$D_{\text{out}} = K_d \frac{d}{dt} e(t)$$

Integral term

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 (K_i) and added to the controller output.

The integral term is given by:

$$I_{\text{out}} = K_i \int_0^t e(\tau)\, d\tau$$

PID Tuning :

Performance of PID depends on the gain parameters. so we need to adjust them .Different methods are used

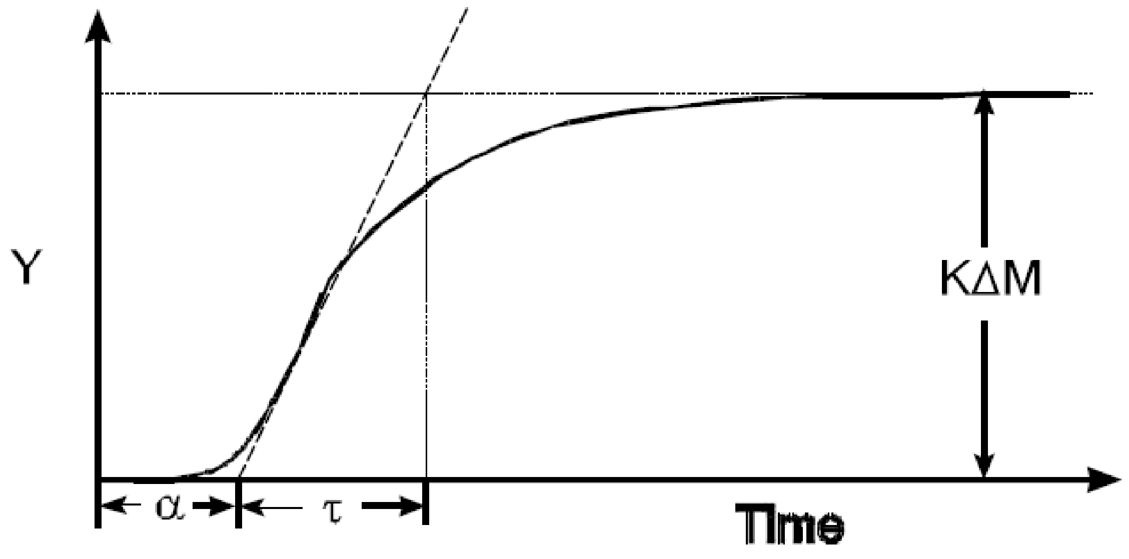
i)open loop method

ii)close loop method

Open Loop Method

Here we apply a step to the process and get the response like as shown in the graph and get the deadtime ,reaction rate and process gain..

- Put the controller in manual mode
- Wait until the process value (Y) is stable and not changing
- Step the output of the PID controller - The step must be big enough to see a significant change in the process value. A rule of thumb is the signal to noise ratio should be greater than 5.
- Collect data and plot as shown below.
- Repeat making the step in the opposite direction.
- K = the process gain=change in process value /change in manipulated value



Getting results:

Ziegler-Nichols			
Controller Type	K_c	τ_I	τ_D
P	$\frac{1}{K}\left(\frac{\tau}{\alpha}\right)$	___	___
PI	$\frac{.9}{K}\left(\frac{\tau}{\alpha}\right)$	3.33α	___
PID	$\frac{1.2}{K}\left(\frac{\tau}{\alpha}\right)$	2.0α	0.5α
Recommended Range of Applicability $1.0<(\alpha/\tau)<0.1$			

ii)close loop method

Ziegler–Nichols method

Another tuning method is formally known as the Ziegler Nichols method, by John G. Ziegler and Nathaniel B. Nichols in the 1944. As in the method above, the K_i and K_d gains are first set to zero. The *P* gain is increased until it reaches the ultimate gain, K_u, at which the output of the loop starts to oscillate. K_u and the oscillation period P_u are used to set the gains as shown:

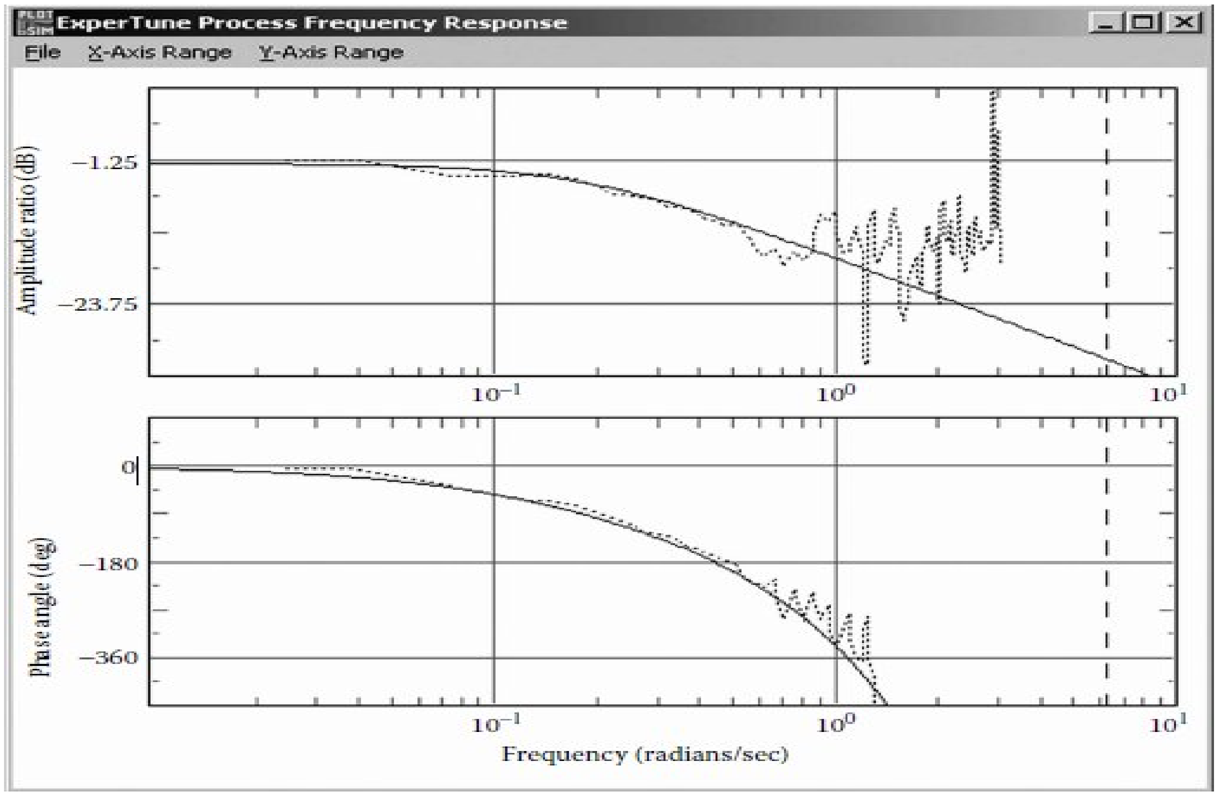
Control Type	K_p	K_i	K_d
P	$0.50K_u$	-	-
PI	$0.45K_u$	$1.2K_p / P_u$	-
PID	$0.60K_u$	$2K_p / P_u$	$K_pP_u / 8$

The main advantage of the closed-loop tuning method is that it considers the dynamics of all system components and therefore gives accurate results at the load where the test is performed. Another advantage is that the readings of *Ku* and *Pu* are easy to read and the period of oscillation can be accurately read even if the measurement is noisy.

The disadvantages of the closed-loop tuning method are that when tuning unknown processes, the amplitudes of undampened oscillations can become excessive (unsafe) and the test can take a long time to perform. One can see that when tuning a slow process (period of oscillation of over an hour),it can take a long time before a state of sustained, undampened oscillation is achieved through this trial-and-error technique. For these reasons, other tuning techniques have also been developed and some of them are described below.First, it is essentially trial-and-error methods,since several values of gain must be tested before the ultimate gain. Second, while one loop is being tested in this manner, its output may affect several other loops, thus possibly upsetting an entire unit.

PID Tuning Based on Frequency Response

First give the plant inputs with different frequency and get the frequency response In most processes, both the amplitude ratio and the phaseangle will decrease with increasing frequencies. Assumingthat the combined phase and amplitude ratio decreases withfrequency when the process and the controller frequencyresponses are combined, the following general stability ruleapplies: A control system will be unstable if the open-loop frequency response has an amplitude ratio that is larger than one when the phase lag is 180 degrees.To provide proper tuning, a margin of safety in the gainand phase is desired. Tuning constants are therefore adjusted to result in the highest gain at all frequencies and yet achieve a certain margin of safety or stability. This is best accomplished using computer software. A graph shows frequency response as :



Soft Computing under Artificial Intelligence:

Intelligence is the ability to acquire,understand and apply knowledge; or the ability to exercise thought and reason.It embodies all the knowledge both conscious and unconscious,which we acquire through study and experience,highly refined sight and sound perception, thought,imagination,ability to converse,read,write and recall facts,express and feel emotions,and much more.

Artificial Intelligence deals with the study and creation of computer systems that exhibit some form of intelligence: systems that can learn new concepts and tasks,systems that can reason and draw conclusions about the world around us,systems that can understand a natural language or perceive and comprehend a visual scene,and systems that perform other feats that require human intelligence.

The motivation of AI technology is to make computers behave more like humans in solving problems.AI is fundamentally different from general programming.Soft computing is a tool of artificial intelligencewhich differs from hard computing in that,unlike hard computing,it is tolerant of imprecision,uncertainty, partial truth and approximation.In effect,the role model of soft computing is the human mind.

The project work is based on exploiting the two efficient swarm intelligence based evolutionary soft computational technique viz. Particle Swarm Optimization (PSO) and Bacterial Foraging Optimization (BFO) to design a PID controller for a low damping plant.

Matlab Basics for the Implementation of Project

LTI VIEWER :-

LTI Viewer is a software package inbuilt in matlab which can produce following information regarding a transfer function:-

-Step Response.

-Impulse Response.

-Bode Plot.

-Nyquist Plot.

-Nichols Chart.

-Pole-Zero Plot.

As the project is objected with transient and steady state response of a low damping plant, we have focused mainly upon the step response through LTI viewer. Steps to use LTI Viewer for any transfer function :-

-We use “ltiview” command to open the LTI Viewer in a program.

-After running the program we have to use the import function from the file window.

-We select the transfer function whose plot we have to trace from the work file and we choose the “stepinfo”.

-We get the plot and then we can point peakovershoot,settling time,rise time etc. from the characteristic menu of the graph.

Transfer Function Basics in Matlab:-

To represent a system transfer function and play with its different parameters they have to be coded in a suitable format in matlab. A function ‘tf’ is used for that. For example:-

A transfer function $T_1(s)=(S^2 +3*S +5)/(S^5 +5*S^4 +3.75*S^3 +21*S^2 +3*S +1)$ can be represented as T1 = tf([1 3 5],[1 5 3.75 21 3 1])

Problem Statement of the Project

The project is objected to design a PID controller for a low damping plant.The low damping plants are the higher order plants which exhibits sluggish behaviour.This means that the plant has large settling time,large peak overshoot which are undesirable for better performance. Here we have selected a model transfer function of a low damping raw plant as follows:-

$T(s) = (25.2*S^2 +21.2*S +3)/(S^5 + 16.58*S^4 + 25.41*S^3 +17.18*S^2+11.70*S+1)$

For the plant model the transfer function is as follows:-

T1=([25.2 21.2 3],[1 16.58 25.41 17.18 11.70 1])

The parameters can be obtained as follows:-

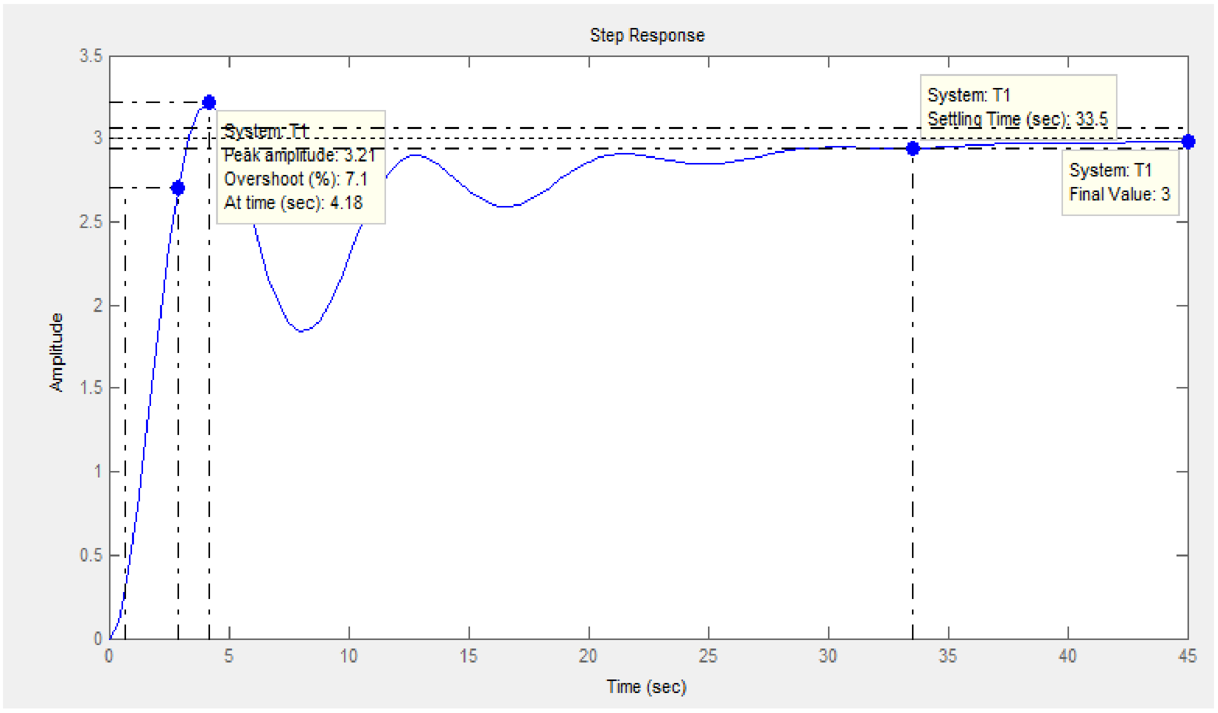
S=Stepinfo{T1,'RiseTimelimits',[0.1, 0.9]}

The above command returns:-

```
S=

  RiseTime : 2.1972 sec

  Settling Time : 33.513 sec

  Overshoot : 7.1023

  Peak : 3.2131

  Peak Time : 4.1789 sec
```

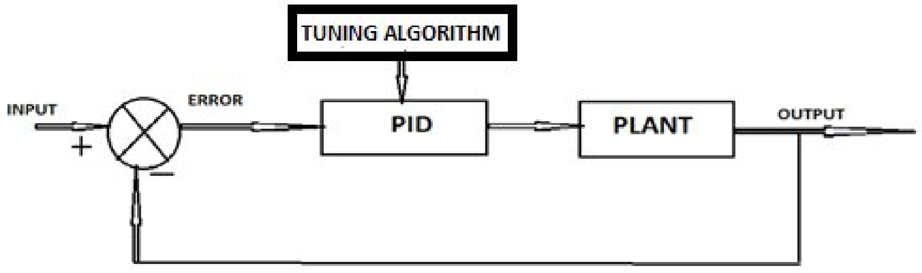
The step response of the raw plant obtained using the LTI Viewer is as shown in the attached graph

Step Response of the Raw Plant Using LTI Viewer



Derivation of Closed Loop Transfer Function for the Plant Model Tuned with the PID Controller

The plant model can be figured as :-



The open loop transfer function of the model :-

$T(s) = (25.2*S^2 + 21.2*S + 3)/(S^5+16.58*S^4 + 25.41*S^3 + 17.18*S^2 + 11.70*S+1)$

Contribution of PID:-

$PID(S) = (k_D*S^2 + k_I + k_p*S)/S$

So, the overall transfer function of the controlled model:-

$C(S)/R(S) = PID(S)*T(S)/(1+PID(S)*T(S)) =$

$$\frac{(25.2*k_D*S^4) + (21.2*k_D+21.5*k_P)*S^3 + (25.2*k_I+21.2*k_p+3*k_D)*S^2 + (21.2*k_I + 3*k_p)*S + 3*K_I}{S^6 + 16.5*S^5 + (25.41 + 25.2*k_D)*S^4 + (17.18 + 21.2*k_D + 25.2*k_P)*S^3 + (11.70 + 25.2*k_I + 21.2*k_P + 3*k_D)*S^2 + (21.2*k_I + 3*k_p + 1)*S + 3*k_I}$$

Concept of Fittness Function for the Design

For our case of design,we had to tune all the three parameters of PID such that it gives the best output results or in other words we have to optimize all the parameters of the PID for best results.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 [K_P K_I K_D] 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.This fitness function consists of several component functions which are the performance index of the design.The point in the search space is the best point for which the fitness function attains an optimal value.

For the case of our design,we have taken four component functions to define fittness function.The fittness function is a function of steady state error, peak overshoot, rise time and settling time.However the contribution of these component functions towards the original fittness function is determined by a scale factor that depends upon the choice of the designer.For this design the best point is the point where the fitness function has the minimal value.

The choosen fitness function is:-

$$F = (1-\exp(-\beta))\,(M_P + E_{SS}) + (\exp(-\beta))(T_S - T_r)$$

Where F:- Fittness function

M_P :- Peak Overshoot

T_S :- Settling Time

T_r :- Rise Time

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

For our case of design we have taken the scaling factor $\beta = 1$.

In the matlab library we have defined a fitness function which has PID parameters as input values and it returns the fitness value of the PID based controlled model as its output. It has the format:-

Function [F] = fitness(K_D K_P K_I)

Fittness function in matlab :-

```
function F= tightnes(kd,kp,ki)
T1=tf([25.2*kd 21.2*kd+25.2*kp
25.2*ki+21.2*kp+3*kd21.2*ki+3*kp 3*ki],[1 16.58 25.41+25.2*kd
17.18+21.2*kd+25.2*kp  11.70+25.2*ki+21.2*kp+3*kd .
21.2*ki+3*kp+1 3*ki]);
S=stepinfo(T1,'RiseTimeLimits',[0.1 0.9]);
tr=S.RiseTime;
ts=S.SettlingTime;
Mp=S.Overshoot;
Ess=1/(1+dcgain(T1));
F=(1-exp(-0.5))*(Mp+Ess)+exp(-0.5)*(ts-tr);
```

We have used this fitness function for the performance evaluation of different combination of PID parameters reflected by the points in the three dimensional search space.

PARTICLE SWARM OPTIMIZATION

Introduction:-

James Kennedy an American Social Psychologist along with Russell C. Eberhart innovated a new evolutionary computational technique termed as Particle Swarm Optimization in 1995.The approach is suitable for solving nonlinear problem.The approach is based on the swarm behavior such as birds finding food by flocking. A basic variant of the PSO algorithm works by having a population (called a swarm) of candidate solution (called particles). These particles are moved around in the search-space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. When improved positions are being discovered these will then come to guide the movements of the swarm. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.Here in this technique a set of particles are put in d-dimensional search space with randomly choosing velocity and position.The initial position of the particle is taken as the best position for the start and then the velocity of the particle is updated based on the experience of other particles of the swarming population.

Algorithm for PSO :-

-The i^{th} particle in the swarm is represented as

$X_i = (x_{i1}, x_{i2}, x_{i3,}..................x_{id})$ in the d-dimensional space.

-The best previous positions of the i^{th} particle is represented as: Pbest = ($Pbest_{i,1}$,$Pbest_{i,2}$,$Pbest_{i,3}$..........$Pbest_{i,d}$)

-The index of the best particle among the group is $Gbest_d$.

-Velocity of the i^{th} particle is represented as $V_i = (V_{i,1}\ V_{i,2}\ V_{i,3}..........\ V_{i,d})$.

-The updated velocity and the distance from $Pbest_{id}$ to $Gbest_{i,d}$ is given as ; $V_{i,m}^{t+1} = W*V_{i,m}^{t} + C_1*rand()*(Pbest_{i,m} - X_{i,m}^{t}) + C_2*rand()*(Gbest_m - X_{i,m}^{t})$

- $X_{i,m}^{(t+1)} = X_{i,m}^{(t)} + V_{i,m}^{(t+1)}$

For i=1,2,3.......n.

m = 1,2,3.....d.

where,

n:- Number of particles in the group.

d:- dimension index.

t:- Pointer of iteration.

$V_{i,m}^{(t)}$:- Velocity of particle at iteration i.

W:- Inertia weight factor.

C_1 , C_2 :- Acceleration Constant.

rand() :- Random number between 0 and 1.

$X_{i,d}^{(t)}$:- Current position of the particle ‘i’ at iteration.

$Pbest_i$:- Best previous position of the ith particle.

Gbest:- Best particle among all the particle in the swarming population.

Algorithmic Approach for the Specified Design :-

In our case, we cast the PID controller design problem in PSO framework as given.We consider the three dimensional search space. K_P , K_I and K_D are the three dimensions.We consider the fitness function based on time domain characteristics for adaptation.We set the number of adaptation iterations based on expected parameters and time of computation.

A Small Illustration of Program :-

-Initially we fixed the values of PSO algorithm constants as :

Inertia weight factor W = 0.3

Acceleration constants C_1 , C_2 = 1.5

-As we have to optimize three parameters, namely K_P ,K_D ,K_I of the controller, we have to search for their optimal value in the three dimensional search space, so we randomly initialized a swarm of population “100” in the three dimensional search space with [$X_{i,1}$ $X_{i,2}$ $X_{i,3}$] and [V_{i1} V_{i2} V_{i3}] as initial position and velocity.

-Calculated the initial fitness function of each point and the point with minimum fitness function is displayed as gbest (initial value of global best optima) and the optimal fitness function as fbest1(Initial best fitness function).

-Runned the program with the PSO algorithm with thousands (or even more numbers) of iterations and the program returned final optimal value of fitness function as “fbest” and final global optimum point as “Gbest”.

Program for PSO:

```
clc

close all

c1=1.5;

c2=1.5;

for i=1:50

   for j=1:3

      X(i,j)=i*rand;

      V(i,j)=i*rand;

      Pbest(i,j)=X(i,j);

   end

end

for i=50:100

   for j=1:3

      X(i,j)=0.5*i*rand;

      V(i,j)=0.5*i*rand;

      Pbest(i,j)=X(i,j);

   end

end
```

```
for i=1:100
   kd=X(i,1);
   kp=X(i,2);
   ki=X(i,3);
   F(i,1)=tightnes(kd,kp,ki);
end
k=1;
m=1;
fbest=F(1,1);
while m<100
   if fbest>F(m,1)
      fbest=F(m,1);
      k=m;
   end
   m=m+1;
end
k1=k;
fbest1=fbest;
Gbest=[X(k,1) X(k,2) X(k,3)]
gbest=Gbest;
k1
fbest1
gbest
for M=1:50
```

```
  for i=1:100
    for j=1:3
      V(i,j)=0.5*(100-i)*V(i,j)+c1*rand*(Pbest(i,j)-
X(i,j))+c2*rand*(Gbest(1,j)-X(i,j));
      X(i,j)=X(i,j)+V(i,j);
    end
    kd1=X(i,1);
    kp1=X(i,2);
    ki1=X(i,3);
    kd=Pbest(i,1);
    kp=Pbest(i,2);
    ki=Pbest(i,3);
    L=tightnes(kd,kp,ki);
    P=tightnes(kd1,kp1,ki1);
    if P<L
      Pbest(i,1)=X(i,1);
      Pbest(i,2)=X(i,2);
      Pbest(i,3)=X(i,3);
    end
  end
  for i=1:100
    kd=Pbest(i,1);
    kp=Pbest(i,2);
    ki=Pbest(i,3);
    F(i,1)=tightnes(kd,kp,ki);
    end
    m=1;
    k=1;
    while m<100
        if fbest>F(m,1)
            fbest=F(m,1);
            k=m;
        end
        m=m+1;
    end
    Gbest=[Pbest(k,1) Pbest(k,2) Pbest(k,3)];
end
k
fbest
```

The Program for the Simulation Plot

This program is to obtain the step response of various optimized systems with optimal [K_D K_P K_I] values

```
clc
close all
kd=input('enter the value of kd');
kp=input('enter the value of kp');
ki=input('enter the value of ki');
T1=tf([25.2*kd 21.2*kd+25.2*kp 25.2*ki+21.2*kp+3*kd 21.2*ki+3*KP
3*ki],[1 16.58 25.41+25.2*kd 17.18+21.2*kd+25.2*kp
11.70+25.2*ki+21.2*kp+3*kd 21.2*ki+3*kp+1 3*ki]);
ltiview
```

PSO based simulation and results

Fittness function of the open loop transfer function of the raw plant :

$$T(s) = (25.2*S^2 + 21.2*S +3)/(S^5+16.58*S^4 +25.41*S^3 +17.18*S^2 +11.70*S+1)$$

F(Raw Plant) = 22.3066

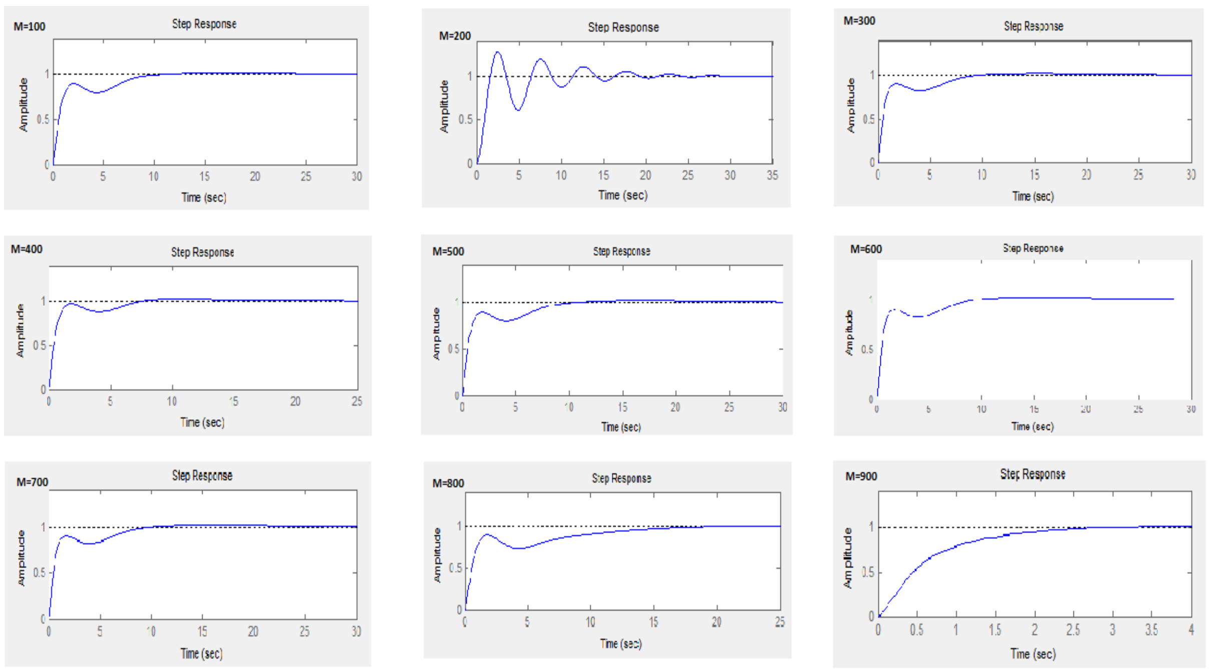
In our simulations using PSO algorithm, we have varied the number of iterations and kept the population of the swarm constant at 200.We present a comparative study of the performance of the initial global best position out of randomly initialized swarm particles to the performance of the final global best position which comes after the application of "particle swarm optimization" algorithm.

The result in the tabular format:

NUMBER OF ITERATIONS	OPTIMAL BEST FITNESS FUNCTION	OPTIMAL BEST POINT
100	2.4148	[0.711 0.678 0.247]
200	2.3973	[0.132 0.747 0.355]
300	2.3950	[0.972 0.767 0.292]
400	2.4703	[1.019 0.912 0.463]
500	2.399	[0.849 0.714 0.249]
600	2.4104	[1.0345 0.787 0.299]
700	2.402	[1.405 0.7937 0.285]
800	1.5531	[0.696 0.812 0.134]
900	1.0407	[1.039 0.424 0.662]
1000	0.5835	[4.12 1.8055 2.616]

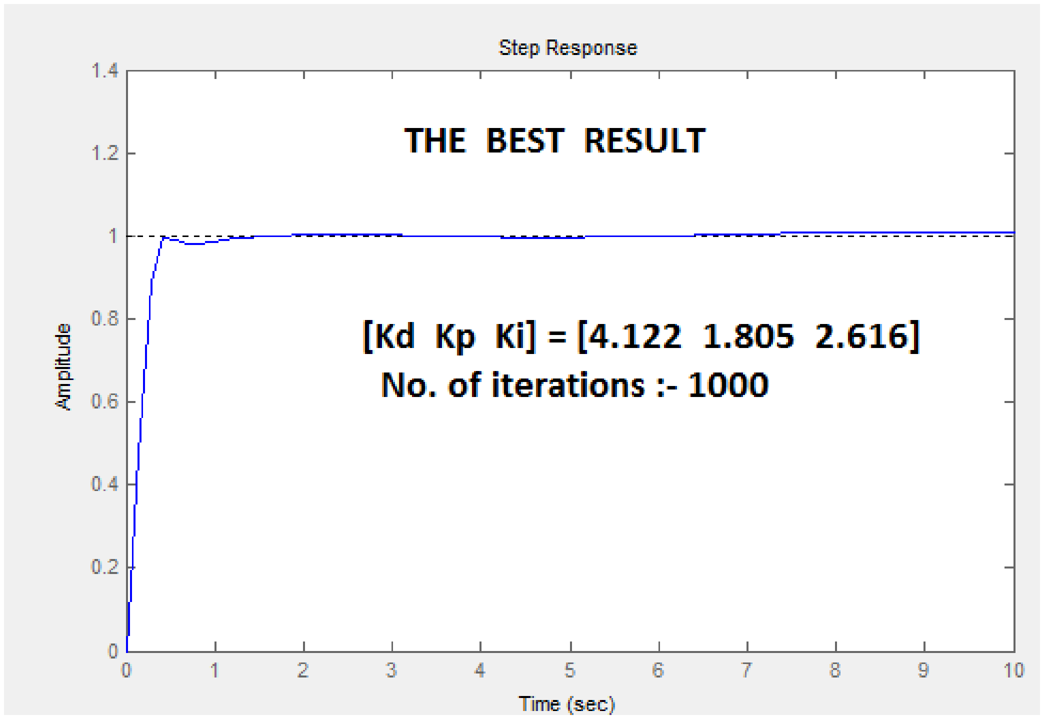
SIMULATION RESULTS WITH DIFFERENT NUMBER OF ITERATIONS

`



“M” represents the number of iterations

THE OPTIMAL DESIGN



Bacterial Foraging Optimization

Introduction :

Based on the research of foraging behaviour of E.colli bacteria Kevin M.Passino and Liu exploited a variety of bacterial foraging and swarming behaviour, discussing how to connect social foraging process with distributed non-gradient optimization.In the bacterial foraging optimization process four motile behaviours are mimicked:-

1)Chemotaxis:

A chemotactic step can be defined as a tumble followed by a tumble or a tumble followed by a run lifetime.To represent a tumble a unit length random direction, $\square(j)$, is generated ; this will be used to define the direction of movement after a tumble. In particular

$\square^i(j+1,k,l) = \square^i(j,k,l) + C(i)*\square(j)$,

Where $\square^i(j,k,l)$ represents the ith bacterium at jth chemotactic, kth reproductive and lth elimination and dispersal step.C(i) is the size of the step taken in the random direction specified by a tumble(run length unit).

2)Swarming:

E.Colli cellscan cooperatively self organize into highly structured colonies with elevated environmental adaptability using an intricate communication mechanism.Overall, cells provide an attraction signal to each other so they swarm together.The mathematical representation for swarming can be represented by

$$J_{cc}(\theta,P(j,k,l)) = J^i_{cc}(\theta,\theta^i(j,k,l)) = \sum[D_{attract} * \exp(-W_{attract} *\sum(\theta_m - \theta^i_m)^2)]$$
$$+\sum[H_{repellant} * \exp(-W_{repellant} *\sum (\theta_m - \theta^i_m)^2)]$$

Where $J_{cc}(\theta,P(j,k,l))$ is the cost function value to be added to the actual cost function to be minimized to present a time varying cost function,S is the total number of bacteria ,P is the number of parameters to be optimized which are present in each bacterium and $D_{attract}$,$W_{attract}$,$h_{repellant}$,$W_{repellant}$ are different coefficients that should be properly choosen.

3)Reproduction:

The least healthier bacteria die and the other each healthier bacteria split into two new bacteria each placed in the same location.

4) Elimination and Dispersal :

It is possible that in the local environment, the lives of a population of bacteria changes either gradually(eg, via consumption of nutrients) or suddenly due to some other influence.Events can occur that all the bacteria in aregion are killed or a group is dispersed into a new part of the environment.They have the effect of possibly destroying the chemotactic progress, but they have also the effect of assisting the chemotactic process, since dispersal may place bacteria near good food sources.From a board perspective, elimination and disprsal are parts of the population level long distance motile behaviour.

Algorithm for Bacterial Foraging Optimization Based Design:

The searching procedures of the proposed BF-PID controller is as follows:-

Step 1)

Initiallize parameters S ,D , N_S,N_C ,N_{re} N_{ed} ,P_{ed} ,□ ,C(i), $D_{attract}$,$W_{attract}$, $H_{repellant}$ and $W_{repellant}$, where

S: Number of bacteria to be used for searching the total region.

D: Number of parameters to be optimized.

N_S: Swimming length after which tumbling of bacteria will be done in a chemotactic step.

N_{re}: Maximum number of reproductions to be undertaken.

N_{ed}: Maximum number of elimination-dispersal events to be imposed over the bacteria.

P_{ed}: Probability with which the elimination-dispersal will continue.

□:The location of each bacterium whichis specified by random numberson [0,1]

C(i): This is chemotactic step size assumed constant for our design.

Step 2)

Elimination-Dispersal loop : l=l+1

Step 3)

Reproduction loop : k = k+1

Step 4)

Chemotaxis loop : j = j + 1

a)For i = 1,2,3,4..........S, take a chemotactic step for i as bacterium follows.

b)Compute J(i,j,k,l), let J(i,j,k,l) = J(i,j,k,l) + J_{CC}(□^{i} (j,k,l),P(j,k,l)) (i.e. add on the cell-to-cell attractant effect to the nutrient concentration).

c)Let J_{Last} = J(i,j,k,l) to save this value since we may find a better cost via run.

d) Tumble : Generate a random number vector □(i) € R^{P} with each element □ m(i) , m= 1,2,3,.........D, a random number on [-1,1].

e)Move : Let

□^{i}(j+1,k,l) = □^{i} (j,k,l) + C(i)* □(i)/(sqrt(□^{T} (i)* □(i))).

This results in a step of size C(i) in the direction of the tumble for bacterium i.

f) Compute J(i,j,k,l), and then let J(i,j,kl) = J(i,j,k,l) + J_{CC}(□(j,k,l),P(j,k,l))

g) Swim : note that we use an approximation since we decide swimming behaviour of each cell as if the bacteria numbered {1,2,.......,i} have moved and {i+1,i+2,i+3......S} have not; this much is simpler to simulate than simultaneous decisions about swimming and tumbling by all the bacteria at the same time:

- Let m = 0 (counter for swim length).

- While m<N_S (if have not climbed down too long)

  .Let m= m+1

  .If J(i,j,k,l) < J_{Last} (if doing better), let

    J_{Last} = J(i,j+1,k,l) and let

      □^{i}(j+1,k,l) = □^{i}(j,k,l) + C(i)* □(i)/(sqrt(□^{T} (i)* □(i))) and use this $\square^{i}(j+1,k,l)$ to compute the new $J(i,j+1,k,l)$ as we did in f).

.Else, let $m = N_S$, this is the end of the while statement.

h) Go to next bacterium (i+1) if ‘i’ is not equal to S(i.e, go to (b)) to process the next bacterium.

Step 5)

If $j < N_C$ go to step 3. In this case , continue chemotaxis , since the life of the bacteria is not over.

Step 6)

Reproduction:

a)For a given k and l,and for each $i = 1,2,3,4.......S$, let $J^{i}_{Health} = \sum J(i,j,k,l)$ be the healt of bacterium (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances). Sort bacteria on chemotactic parameters C(i) in order of increasing cost J_{Health} (higher cost means lower health).

b)The S_r bacteria with highest J_{Health} values die and the other S_r bacteria with the best values split and the copies that are made are placed at the same location as their parent.

Step 7)

If $k<N_{re}$, go to step 2. In this case, we have not reached the number of specified reproduction steps, so we start the next generation in the next chemotactic step.

Step 8)

Elimination-Dispersal : For $i = 1,2,3,4.......S$, with probability P_{ed} , eliminate and disperse each bacterium (this keeps the number of bacteria in the swarming population constant). To do this, if we eliminate a bacterium, simply disperse one into a random location in the optimization domain.

Step 9)

If $l < N_{ed}$, then go to step 1, otherwise end

Program for BFO:

```
clc
close all
s=99;
P=0.25;
Nc=4;
Nre=6;
Ned=2;
Ns=3;
Datt=0.05;
Watt=0.02;
Hrep=0.05;
Wrep=0.05;
for i=1:s
   for n=1:3
      q(i,n)=rand;
   end
end
for l=1:Ned
        for i=1:s
            for n=1:3
              R(i,n)=rand;
              if(R(i,n)<P)
                 q(i,n)=R(i,n);
              end
          end
      end
      for i=1:s
          sum=0;
          for m=1:s
              st=0;
              for n=1:3
                 st=st+(q(i,n)-q(m,n))*(q(i,n)-q(m,n));
              end
              sum=sum+(-Datt)*exp((-Watt)*st)+(Hrep)*exp((-Wrep)*st);
          end
          Kd=q(i,1);
          Kp=q(i,2);
          Ki=q(i,3);
          F(i,1)=tightnes(Kd,Kp,Ki);
          J(i,1)=F(i,1)+ sum;
      end
      for k=1:Nre
              for i=1:s
                 H(1,i)=J(i,1);
                 c=rand;
                 for j=2:Nc+1
                     sum=0;
```

```
for n=1:3
   vec(1,n)=rand;
   sum = sum + vec(1,n)*vec(1,n);
end
abs= sqrt(sum);
for n=1:3
   q(i,n)=q(i,n)+ c*vec(1,n)/abs;
end
sum=0;
for m=1:s
   st=0;
   for n=1:3
      st=st+(q(i,n)-q(m,n))*(q(i,n)-q(m,n));
   end
   sum=sum+(-Datt)*exp((-Watt)*st)+Hrep*exp((-Wrep)*st);
end
Kd=q(i,1);
Kp=q(i,2);
Ki=q(i,3);
F(i,1)=tightnes(Kd,Kp,Ki);
J(i,j)=F(i,1)+sum;
for count=1:Ns
      for n=1:3
         a(1,n)=q(i,n);
```

```
            end
            for n=1:3
                q(i,n)=q(i,n)+c*vec(1,n)/abs;
            end
            sum=0;
            for m=1:s
                st=0;
                for n=1:3
                    st=st+(q(i,n)-q(m,n))*(q(i,n)-q(m,n));
                end
                sum=sum+(-Datt)*exp((-Watt)*st)+Hrep*exp((-
Wrep)*st);
            end
            Kd=q(i,1);
            Kp=q(i,2);
            Ki=q(i,3);
            F(i,1)=tightnes(Kd,Kp,Ki);
            sum=sum+F(i,1);
            if(sum<J(i,j))
                J(i,j)=sum;
            else
                for n=1:3
                    q(i,n)=a(1,n);
                end
            end
```

```
        end
        H(1,i)=H(1,i)+J(i,j);
    end
end
for i=1:s
    HD(1,i)=H(1,i);
end
for i=1:s-1
    for j=i+1:s
        if(H(1,j)<H(1,i))
            t=H(1,i);
            H(1,i)=H(1,j);
            H(1,j)=t;
        end
    end
end
for i=1:s
    for m=1:s
            if((H(1,i)-HD(1,m))==0)
            for n=1:3
                q(i,n)=q(m,n);
            end
            end
    end
```

```
            end
            sr=(s+1)/2;
            for i=1:sr-1
               for n=1:3
                  q(sr+i,n)=q(sr-i,n);
               end
            end
      end
end
for i=1:s
   Kd=q(i,1);
   Kp=q(i,2);
   Ki=q(i,3);
   F(1,i)=tightnes(Kd,Kp,Ki);
end
j=0;
fbest=F(1,1);
for i=1:s
   if(fbest<F(1,i))
      F(1,i)=fbest;
      j=i;
   end
end
fbest
```

Kd=q(j,1)

Kp=q(j,2)

Ki=q(j,3)

BFO based simulations and results:

Just as in the case of previous design of PID controller with PSO, here also we have designed the PID controller for the same low damping plant using BFO algorithm.

The open loop transfer function of the raw plant :

$T(s) = (25.2*S^2 + 21.2*S +3)/(S^5+16.58*S^4 +25.41*S^3 +17.18*S^2 +11.70*S+1)$

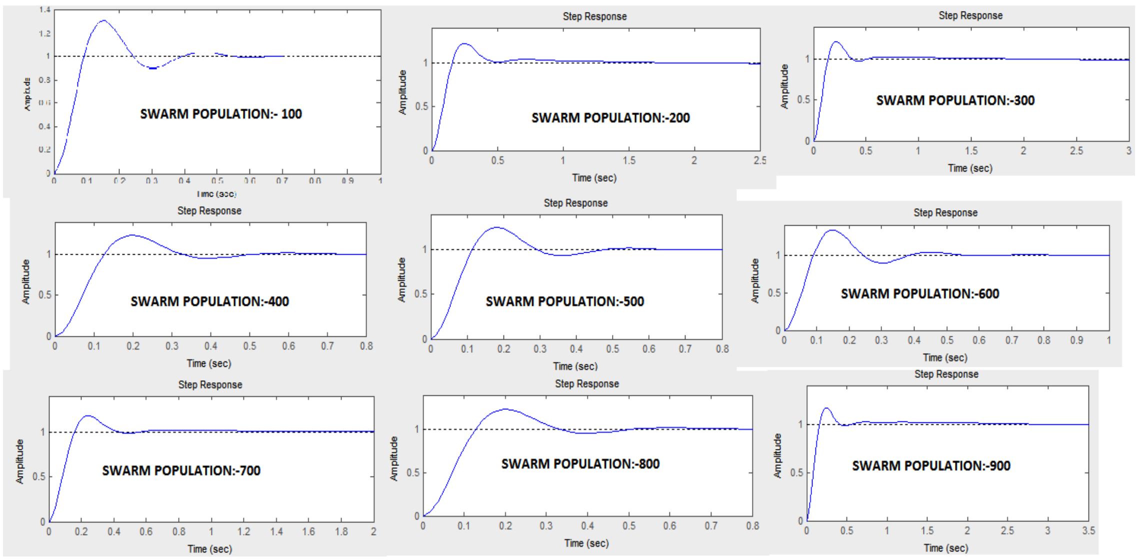
F(raw plant) = 22.3066

In our simulations using BFO algorithm,we have varied the swarm population from 100 to 1000 keeping other constraints fixed.We present a study of the performance of designs with different values of swarming population.

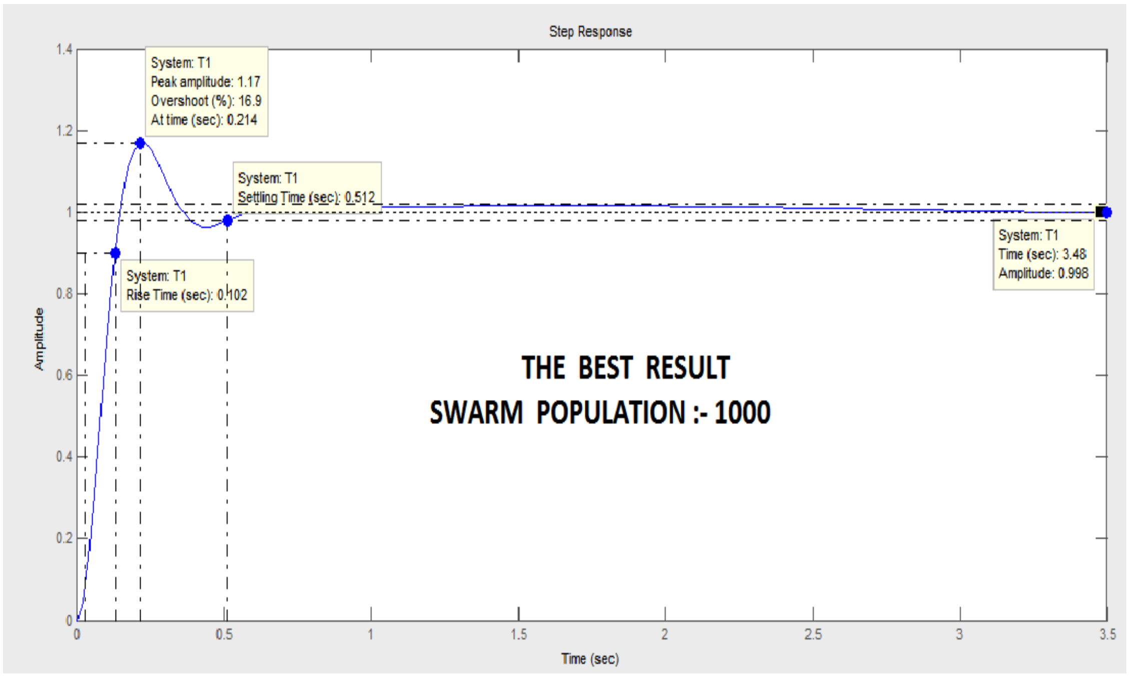
The result in tabular format:

SWARMING POPULATION	FINAL OPTIMAL FITNESS FUNCTION	FINAL BEST OPTIMAL POINT[K_D,K_P,K_I]
100	11.5042	[19.354 11.422 14.576]
200	9.4437	[8.751 12.55 11.1589]
300	9.2013	[10.831 10.41 13.22]
400	8.7138	[12.30 10.23 9.77]
500	8.6759	[14.29 8.748 12.5363]
600	7.6163	[19.64 17.94 7.629]
700	6.4021	[9.0734 8.1502 7.3855]
800	5.9524	[12.11 22.029 20.7385]
900	5.6177	[9.0314 6.2261 9.2851]
1000	5.1379	[10.19 4.8509 9.4878]

SIMULATION RESULTS WITH DIFFERENT NUMBERS OF . SWARMING POPULATION



THE OPTIMAL DESIGN WITH BEST RESULT



CONCLUSION:

According to the analysis done on the basis of results obtained, we have landed to a conclusion that for the design of a PID controller for the low damping plant Particle Swarm Optimization technique gives a better result than Bacterial Foraging Optimization technique.In the case of PSO implementation we have varied the number of iterations that means the number of steps to be taken by the swarming particles in the search space.The results obtained indicate that as the number of iterations went on increasing the performance of the system also went on improving.We have varied the number of iterations from 100 to 1000 and the best performance was obtained with 1000 number of iterations.

While implementing Bacterial Foraging Optimization technique for the design, we have varied the swarming population from 100 to 1000 keeping all the other constraints fixed.As the result we observed that as the swarming population went on increasing the performance of the system also went on improving.The best result was obtained with the swarming population of 1000.

A comparative study of both the algorithms for the specified design shows that the best fitness function obtained with the PSO algorithm was 0.5835 and the best fitness function obtained with the BFO algorithm was 5.1379 indicating that PSO technique is performing better than the BFO technique for the specified design.

A Concluding Remark:

Undoubtedly the bio inspired evolutionary computational techniques have increased the human reach in the field of Artificial Intelligence Technology and we are surfacing up with better and efficient design solutions,but still the best is yet to come.Right now with these probability and randomness based technologies we cannot claim that we can trace the whole optimization domain. Moreover, we cannot always guarantee that the algorithm is not going to be trapped at local optima.So we conclude with the positive hope that in near future we will have best technologies of Artificial Intelligence which will sort out all the above mentioned shortcomings of contemporary technologies.

REFERENCE

[1] Anandanatarajan R., Chidambaram M. and Jayasingh T., "Limitations of a PI controller for a first-order nonlinear process with dead time", *ISA Transactions,* Vol. 45,

[2] Åström K.J., "Automatic Tuning of PID Controller", Instrument Society of America, Research Triangle Park, 1995

[3] Åström K.J. and Hägglund T., "Automatic tuning of simple regulators with specification on phase and amplitude margins", *Automatica*, Vol. 20, pp. 645-651, 1984

[4] Åström K., and Hägglund T., "PID controllers: Theory, Design and Tuning", ISA, Research Triangle Park, NC, 1995

[5] Åström K., and Hägglund T., "Revisiting the Ziegler-Nichols Step Response method for PID control ", *Journal of Process Control,* Vol. 14, pp. 635-650, 2004