**Design of PID Controller for a given SISO system using Symbolic MATLAB and Genetic Algorithm**

Thesis Submitted in Partial Fulfillment of the Requirements

for the degree of

**Bachelor of Technology**

**in**

**Electrical Engineering**

By

Anil Karaka,

06EE1023,

Electrical Engineering.

Under the esteemed Guidance of

Prof. Jayanta Pal

(Department of Electrical Engineering)



Department of Electrical Engineering

Indian Institute of Technology

Kharagpur – May 2010

**CERTIFICATE**

This is to certify that the thesis titled “**Design of PID Controller for a given SISO system using Symbolic MATLAB and Genetic Algorithm**”submitted by Anil Karaka to the Department of Electrical Engineering in partial fulfillment for the award of the degree of Bachelor of Technology is a bona fide record of work carried out by him under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of this Institute and, in my opinion, has reached the standard needed for submission.

May 4th, 2010 **Prof. Jayanta Pal**

Department of Electrical Engineering

Indian Institute of Technology

Kharagpur - 721302

****

**CERTIFICATE OF EXAMINATION**

This is to certify that the thesis titled “**Design of PID Controller for a given SISO system using Symbolic MATLAB and Genetic Algorithm**”submitted by Anil Karaka to the Department of Electrical Engineering in partial fulfillment for the award of the degree of Bachelor of Technology is a bona fide record of work carried out by him under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of this Institute and, in my opinion, has reached the standard needed for submission.

**Examiners**

**Supervisor**

****

**ACKNOWLEDGEMENTS**

Acknowledgement is not a mere formality but a genuine opportunity to express the sincere thanks to all those, without whose active support and encouragement, this project would not have been successful.

I take the privilege to express my deep sense of gratitude and indebtedness to **Prof. Jayanta Pal** for his valuable suggestions, keen interest, patience, learned counsel, sagacious guidance, kind and constant encouragement in the course of my project work as a guide.

I am grateful to my parents for their support and inspiration. I am also thankful to all the faculty members, my batch mates and my friends for their support.

**Anil Karaka (06EE1023)**

**Department of Electrical Engineering**

**Indian Institute of Technology**

**Kharagpur 721302**

**West Bengal, India**

# Abstract

The objective of this project is to study various ways of designing a PID controller, mainly the use of Symbolic Math toolbox in MATLAB and Genetic Algorithms. The performance index used in both the methods is the integral of squared error over a large period of time.

In Symbolic case we obtained the error function in terms of the controller variables and from that by minimizing the cost function which is also in terms of controller variables further resulted in obtaining the controller variable. It also involves mathematical complexity which precludes this method in case of more than two controller variables

In genetic algorithm case performance index is used as a fitness function. Various methods of using genetic algorithm operators are also discussed. Genetic representation is based on the initial limits given for KP, KI, KD considering the optimal solution lies within that range. The minimization problem is converted into a maximizing problem by altering the fitness function.

Various combinations of GA operators are discussed including the changes in the crossover probability and mutation probability.

**Table of Contents**

**Table of Figures**

[Figure 1, Basic block diagram 8](#_Toc260753751)

[Figure 2, Cost Function Vs., KP in general case 11](#_Toc260753752)

[Figure 3, J Vs., KP 13](#_Toc260753753)

[Figure 4, Step response for a particular phenotype 17](#_Toc260753754)

[Figure 5, Squared error graph 17](#_Toc260753755)

[Figure 6, Roulette wheel selection 20](#_Toc260753756)

[Figure 7, two point crossover 21](file:///C:\Users\a\Desktop\BTP\06EE1023(Thesis).docx#_Toc260753759)

[Figure 8, initial population distribution 31](#_Toc260753757)

[Figure 9, population distribution after 5 generations 32](#_Toc260753758)

1. **Introduction**
   1. **Design of Control Systems**

Actual control systems are generally nonlinear. However, if they can be approximated by linear mathematical models, we may use one or more of the well-developed design methods. Generally, the performance specifications given to the particular system suggest which method to use. If the performance specifications are given in terms of transient-response characteristics and/or frequency-domain performance measures, then we have no choice but to use a conventional approach based on the root-locus and/or frequency-response methods. If the performance specifications are given as performance indexes in terms of state variables, then modern control approaches should be used. By applying modern control theory, the designer is able to start from a performance index, together with constraints imposed on the system, and to proceed to design a stable system by a completely analytical procedure. The advantage of design based on such modern control theory is that it enables the designer to produce a control system that is optimal with respect to the performance index considered.

Various design techniques include root-locus approaches to the design of various compensators, frequency response approach of designing compensators, PID controller design, and design of control systems in state space etc.,

* 1. **PID Controller**

More than half of the industrial controllers in use today utilize PID or modified PID control schemes. Because most PID controllers are adjusted on site, many different types of tuning rules have been proposed. Using these tuning rules delicate and fine tuning of PID controllers can be made on site. Some PID controllers possess on-line automatic tuning capabilities.

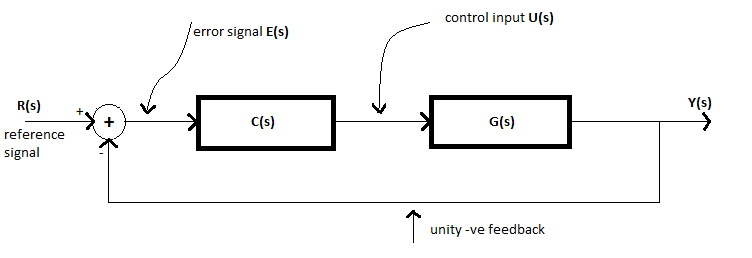


Figure 1, Basic block diagram

Figure 1 shows a PID control of a plant. If a mathematical model of the plant G(s) can be derived, then it is possible to apply various design techniques for determining parameters of the controller that will meet the design specifications of the closed-loop system. As it is a PID controller

The process of obtaining the controller parameters, here KP, KI and KD to meet design specification is called controller tuning. We have various tuning methods based on Zeigler-Nicholas rules that are based on experimental step-responses or KP values obtained from marginal stability when only the proportional control action is used.

In the first part of the thesis PID control design is done using Symbolic Math toolbox from MATLAB and its effectiveness is discussed. In the second part the controller parameters are obtained from Genetic Algorithm.

* 1. **Scope of Thesis**

The first part of this thesis includes design of a proportional controller for a given plant using Symbolic Math toolbox in MATLAB. In this method we obtain actual time response expression in terms of controller variables is obtained. We find the controller variables by minimizing the cost function which in this case is the integral of squared error over large period of time. While designing a PI controller using the same method the mathematical complexity involved shows the inefficiency of this method for more than one controller variable.

The second part of the thesis includes designing PID controller using Genetic algorithm. Various ways of obtaining the controller variables by using different crossover and mutation combinations for a particular plant is discussed.

1. **Design using Symbolic Math Toolbox**
   1. **Introduction**

The basic block diagram is shown in figure 1. The actual time response expression is obtained using symbolic toolbox in MATLAB. The cost function is the integral of squared error over large period of time. This error will be in terms of the controller variables by minimizing the cast function we can obtain the controller variables.

From the block diagram the following equations can be written.

…….1

sign usingracteristics and/or

From the above equation error function is obtained, and from this cost function which is nothing but the integral of squared error over a large period of time.

J here is the cost function. By using symbolic math toolbox we obtain it in terms of KP in case of proportional controller.

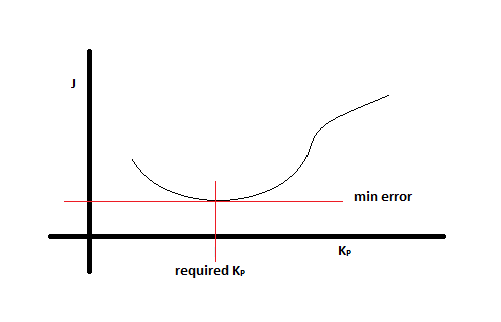


Figure , Cost Function Vs., KP in general case

Hence by minimizing the cost function we obtain KP

* 1. **Proportional Controller**

In case of proportional controller C(s) =KP, and given plant transfer function is

Using the following MATLAB code the equations above are solved

syms Kp s t; %defining symbolic variables

G=3/(s^2+4\*s+3);

C=KP;

Y= (1/s)\*(G\*C/(1+G\*C));

Y=simplify(Y);

y=ilaplace(Y);

y=simplify(y);

From the above code we obtain y(t) in terms of KP

where:

From this the cost function J is obtained in terms of KP.

J=f(KP)

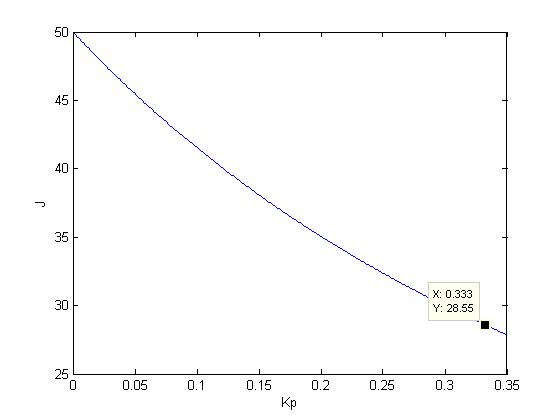


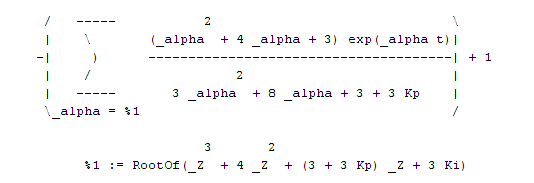
Figure , J Vs., KP

From the above plot KP is found to be 0.333 as it is the least possible value and the further lower values represent complex values of J and hence neglected.

* 1. **PI Controller**

In case of proportional plus integral controller the controller transfer function is as follows

By performing similar steps as in case of a proportional controller we get y(t) as



For getting KP and KI we have to calculate the cost function in terms of those variables and minimize it for different values of KP and KI. The algebraic complexity involved precludes this method for problems of imvolving more than one variable.

1. **Genetic algorithm**
   1. **Introduction**

The term genetic algorithm, almost universally abbreviated nowadays to GA, was first used by Holland, was instrumental in creating what is now a flourishing field of research and application that goes much wider than the original GA. The subject now includes evolution strategies (ES), evolutionary programming (EP), artificial life (AL), classifier systems (CS), genetic programming (GP) and etc. GA follows biological terminology fairly closely. GA can allow ‘populations’ of potential solutions to optimization problems to die or reproduce with variations, gradually becoming adapted to their ‘environment’, by means of an externally imposed measure of ‘fitness’.

GA follows biological terminology fairly closely, describing the encoded individuals that undergo reproduction as **‘chromosomes**’ the basic units of those individuals as ‘**genes**’, and the set of values that the genes can assume as ‘**alleles**’. The position of a gene in a string is called its ‘**locus**’. The distinction between the ‘**genotype**’- the actual genetic structure that represents an individual and the ‘**phenotype**’- its observed characteristics as an organism, has also become a popular analogy.

GA is a search technique used in computing to find exact or approximate solutions to optimization and search problems. These are implemented in a computer simulation in which a ‘**population**’ of abstract representations (called chromosomes or the ‘**genotype**’ of the genome) of candidate solutions (called individuals, creatures, or ‘**phenotypes**’) to an optimization problem evolves towards better solution. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation the fitness of each individual in the populations is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

Genetic Algorithms find applications in bioinformatics, computational science, engineering, economics, chemistry, manufacturing, mathematics, and many other fields.

A typical genetic algorithm requires:

* A ‘**genetic representation**’ of the solution domain
* A ‘**fitness function’** to evaluate the solution domain
  1. **Genetic representation**

A standard representation of the solution is an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Genetic representation is nothing but defining how phenotypes and genotypes are connected. The length of the array and population strength is highly dependent on the problem. Empirical results from many authors suggest that the population size as small as 30 are quite adequate in many cases. In this case also the population is chosen as 30.

In this design of PID controller problem, the phenotype is KP, KI, KD. It has to be converted into genotype based on the limits given in the problem statement. Each variable is converted into an 8-bit binary form and constitutes a gene. These 3 genes were combined to form the chromosome. It is done as shown in the below table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| KP | | | | | | | | KI | | | | | | | | KD | | | | | | | |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Based on the limits of each variable and the binary form of equivalent phenotypes are obtained.

Here the above chromosome represents KP=10.2745, KI=8.1373, KD=2.9765. Given respective ranges are KP(5,12), KI(5,10), KD(1,5)

* 1. **Fitness function**

A fitness function is defined over the genetic representation and measures the quality of represented solution. The fitness function is always problem dependent. Fitness function defines how much is a particular chromosome fit to enter next generation.

In this particular problem the fitness function nothing but the area under squared error over a large period of time. For example a chromosome with KP, KI, KD values (10, 8.7033, 1.33) respectively the fitness is as shown below

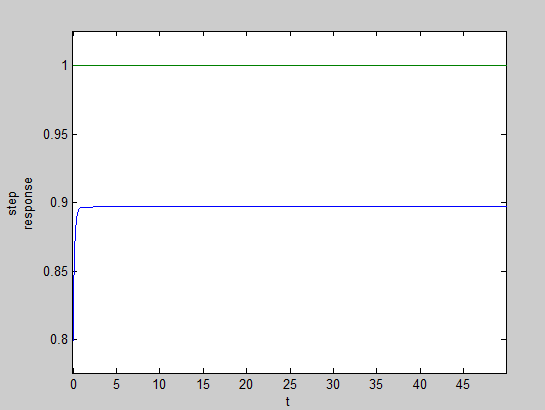


Figure , Step response for a particular phenotype

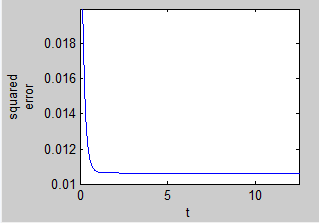


Figure , Squared error graph

Fitness=0.5354 (area under above curve)

In this particular problem the area under squared error graph must be minimized, hence it’s essentially a minimization problem. This fitness can be modified accordingly to make it a maximization problem which will be discussed further in the thesis.

1. **Various stages in the algorithm**

Various important stages in a GA are

* Initialization
* Selection
* Crossover
* Mutation
* Termination

The above stages are discussed individually in below sections, crossover and mutation together comprises reproduction

* 1. **Initialization**

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the *search space*). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

In this problem the population is seeded in areas where it is likely found, the range of KP, KI, KD is given. As initialization is done randomly, we generate a 30X24 size matrix containing 1s and 0s is generated as initial population using the command

C=round(rand(30,24));

This results in

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| KP | | | | | | | | KI | | | | | | | | KD | | | | | | | |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

And so on till 30 chromosomes

The respective phenotypes are

|  |  |  |
| --- | --- | --- |
| KP | KI | KD |
| 10.2745 | 8.1373 | 2.9765 |
| 7.3529 | 8.8627 | 1.5490 |
| 9.5098 | 6.5882 | 3.9647 |
| 10.4902 | 8.7451 | 2.6000 |
| 9.9602 | 7.3529 | 2.1765 |
| 10.7255 | 5.4902 | 2.5373 |
|  |  |  |

And so on till 30 sets

* 1. **Selection**

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming.

Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection.

We used Roulette wheel selection method. This method is based on the probability of each chromosome. Its probability is based on the fitness of each chromosome. More fit the chromosome more is the probability for it to be selected for further generations. The basic idea is that it should relate to fitness

Its explained below, consider a population set of 5. The chromosomes are distributed around a wheel based on their respectively, angle made by the sector representing their fitness. And a pseudo random number is selected each time a chromosome is required

Wheel is rotated

Selection point

1

2

3

4

5

Weakest individual

Strongest individual

Figure , Roulette wheel selection

The same thing is achieved for a population of 30 in this case. Also as the original problem is a minimization problem the fitness has to be altered a little bit so that the fittest chromosome has the highest probability

Or

By doing this we are making the fittest individual have more value to the fitness obtained from the fitness function.

* 1. **Crossover**

In genetic algorithm, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. There are many crossover techniques

* One-point crossover
* Two-point crossover
* Uniform crossover and etc.

Here two-point crossover is used.

Two point crossover is nothing but two points are selected randomly in both the parent chromosomes and the middle part is swapped between the two to generate children chromosomes.

parents

children

Figure , two point crossover

The next generation is produced using this mating process. This is performed by two parents creating some offspring. The offspring will consist of the genetic material of both parents. There are three options regarding the fitness of the offspring, they can be weaker, the same or fitter than their parents. If they are weaker they will tend to die out – if they are stronger their chances of survival are better. It is of general note that the stronger the parents are in terms of fitness then the fitter the offspring will be. The variation caused by this process allows the offspring to search out different available niches, find better fitness values and subsequently better solutions.

The only general requirement is that the offspring carry forward the important genetic material of the parents, whilst introducing enough variation that they can potentially become fitter. This crossover method emulates this process by exchanging chromosome patterns between individuals to create offspring for the next generation.

In a two=point crossover we select two parents from the general population using selection techniques. We take a random value between 0 and 1. This is measured against the crossover probability of *Px* (usually about 0.7). This emulates the fact that not all individuals mate every generation. If the random value is greater than *Px* the *parents* are simply placed into the next generation unaltered. If the value is less than *Px* crossover takes place.

Crossover begins by generating two random numbers in the range of the number of bits that make up the chromosome length. A gene slice is then taken from both *parents* and exchanged. These then form two offspring - the *child* chromosomes, which are then placed into the next generation. Mating is deemed to have taken place and chromosome patterns exchanged. It is clearly shown in the above figure-6

The children and unmated parents generated using this part of the overall algorithm is then potentially subjected to a mutation operator.

* 1. **Mutation**

Given that the previous parts of the **general algorithm** have measured fitness, performed **selection based on fitness** and then emulated the **mating process**; it may be assumed that enough distribution of the chromosome gene patterns has taken place to fully investigate all possible solutions. This is not so. If we consider the spread of chromosome patterns that existed with the initial random population it is possible to see that the processes carried out so far have not actually introduced any new variation in patterns.

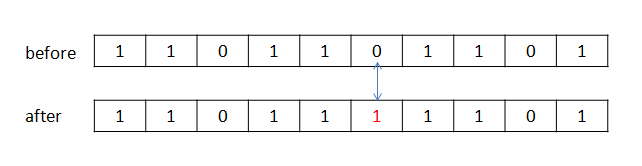
Although this occurs very infrequently many believe this is a main driving force for evolution. The result of mutation can often result in a weaker individual. Occasionally the result might be to produce a stronger one. In each case we may assume that mutation is doing something new by subtly changing some part of the chromosome. The assumption is also made that such changes occur spontaneously and with no reference or affect from other members of the population. In general mutation is something new that is happening.

If the change is beneficial to the general population then that individual will tend to survive and will pass the changed gene onto future generations. If the change causes a weakness then it is likely the individual (and any offspring) will die out.

It can be achieved in many ways but the most important aspects are that it should occur rarely and that one individual should have its pattern adjusted very slightly. Mutation should occur infrequently; otherwise they will have a disruptive effect on the fitness of the overall population.

To perform mutation we iterate through the entire population. For each individual we first select a random value between 0 and 1 and compare this with the mutation probability fact *Pm*. Since mutation occurs very infrequently this is usually set very low (typically *Pm* = 0.001). If the random value is greater than *Pm* the individual is left alone. If the value is less than *Pm* (which is very rare) we perform a mutation operation on the individual.

Select a random value across the length of the chromosome if the generated number is less than Pm and apply a not gate to that allele. The net result of this occasional operation is that the chromosome patterns of the population are adjusted so that the potential exists for the full search space to be investigated.



Mutation is vital in ensuring population diversity.

* 1. **Termination**

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

* A solution is found that satisfies minimum criteria
* Fixed number of generations reached
* Allocated budget (computation time/money) reached
* The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
* Manual inspection
* Combinations of the above

1. **Genetic Algorithm Flow Chart**

initialization

Fitness function

selection

crossover

mutation

next generation

Termination condition

no

yes

Final population

solution

1. **Various functions included in the MATLAB folder**
   1. **k=decimal(c)**

If we input the entire population (chromosomes) into this function we get the respective phenotype i.e. we get KP, KI, KD.

The input will be a 30X24 matrix, output will be 30X3 matrix

function k=decimal(c)

for i=1:30

for j=1:8;

u(j)=c(i,j); %Kp in binary form

end

for j=9:16;

v(j-8)=c(i,j); %Kd in binary form

end

for j=17:24;

w(j-16)=c(i,j); %Ki in binary form

end

dec1=0;dec2=0;dec3=0;

l=7;

for j=1:8;

dec1=dec1+u(j)\*(2^l);

dec2=dec2+v(j)\*(2^l);

dec3=dec3+w(j)\*(2^l);

l=l-1;

end

ki(i)=5+5\*dec2/(bin2dec('11111111')); %based on the respective ranges

kd(i)=1+4\*dec3/(bin2dec('11111111'));

kp(i)=7+5\*dec1/(bin2dec('11111111'));

k(i,1)=kp(i);

k(i,2)=ki(i);

k(i,3)=kd(i);

end

end

* 1. **a=roulette(c)**

This function is to select a chromosome based on the roulette wheel method. Each time this function is performed on the entire population it returns a selected chromosome so that it can move further

Input is a 30X24 matrix and output is a 1X24 matrix

function a=roulette(c)

e=0; %e is the sum of all fitneses

d=0; %d differentiation points in roulette wheel

k=decimal(c);

for i=1:30

f(i)=funct(k(i,1),k(i,2),k(i,3));

e=e+f(i);

end

for i=1:30

d=f(i)/e+d;

r(i)=d;

end

x=rand;

for i=1:30

if x<r(i)

y=i;

break

end

end

a=c(y,:); %selected chromomsome for crossover

end

* 1. **cnew=cross(c,pc)**

This function performs the cross operation as discussed above and returns children chromosomes. The input is the parent generation and the crossing coefficient, the rate at which crossing has to be done. Input and output both are 30X24 size matrices.

function cnew=cross(c,pc)

i=1;

for i=1:15

d=roulette(c);

e=roulette(c);

a=rand;

if a>pc

cnew(2\*i-1,:)=d;

cnew(2\*i,:)=e;

else

a1=round(rand\*24);

a2=round(rand\*24);

a3=min(a1,a2);

a4=max(a1,a2);

if a3==0

a3=1;

else

end

if a4==0

a4=1;

else

end

for j=1:a3

cnew(2\*i-1,j)=d(j);

cnew(2\*1)=e(j);

end

for j=a3:a4

cnew(2\*i-1,j)=e(j);

cnew(2\*i)=d(j);

end

for j=a4:24

cnew(2\*i-1,j)=d(j);

cnew(2\*i,j)=e(j);

end

end

end

end

* 1. **cnew=mutate(c,pm)**

This function performs mutation on existing population on which crossover is operated. The input will be a population and mutation coefficient.

Output is a 30X24 size matrix which is the next generation of chromosomes.

function cnew=mutate(c,pm)

cnew=c;

for i=1:30

if rand<pm

j=round(rand\*24);

if j==0

j=1;

end

cnew(i,j)=~c(i,j);

end

end

end

* 1. **f=funct(kp,ki,kd)**

This function is to calculate the fitness of a given phenotype. This takes KP, KI and KD as inputs and returns the fitness.

function f=funct(kp,ki,kd)

s=tf('s');

g=3/(s^2+4\*s+3);% plant

c=(kd\*s^2+kp\*s+ki)/s;%controller

cloop=feedback(g\*c,1);%closedloop transfer function

%step response

t=0:.01:50;

[y,t]=step(cloop,t);

%plot(t,y);%step response

i=1;

while i<5002;

z(i)=(y(i)-1)^2;

i=i+1;

end

%plot(t,z);

f=trapz(t,z); %area under squared error function

f=1/(1+f); %for a mimimizing problem fitness has to be

end %changed so that we can continue as miximizin problem

* 1. **plot(c)**

This function plots the population distribution in 3 dimensional axes.

function plot(c)

k=decimal(c);

for i=1:30

x(i)=k(i,1);

y(i)=k(i,2);

z(i)=k(i,3);

end

x=x';

scatter3 (x, y, z, 'DisplayName', 'kp, ki, kd'); figure(gcf);

end

1. **Results**

For the given plant transfer function G(s), and the ranges of KP, KI, KD

KP= (7, 12)

KI= (5, 10)

KD= (1, 5)

And fitness used in roulette wheel selection method used is

The initial population distribution is as follows

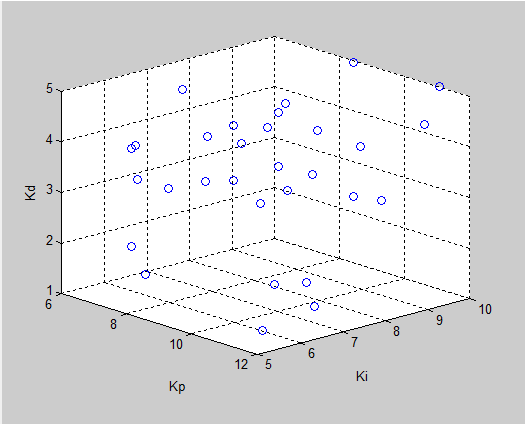


Figure , initial population distribution

Population distribution after 5 generations is as follows, and is seems to be converging in the corner.

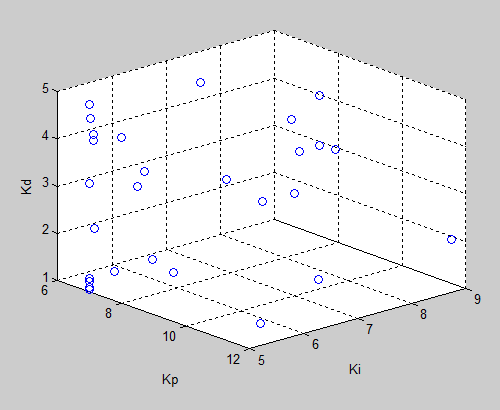


Figure , population distribution after 5 generations

KP=7.1569

KI=5.0196

KD=1.9569

1. **Conclusion**

The results obtained may not represent the most optimum PID controller and there are many ways in which it can be improved. By changing different ways in which crossover can be done, the results can be improved. Also while modifying the fitness function for making it a maximizing function during roulette wheel selection, its effect may not be transferred effectively. There are many combinations of operators that can be applied in the algorithm to get best results. Also the results are converging much earlier in the generations.

1. **References**

* Genetic Algorithms-Principles and perspectives-A guide to GA theory –by Colin R. Reeves and Jonathan E. Rowe
* Foundations of Genetic Algorithm –by Worthy N. Martin and William M. Spears
* Modern Control Engineering –by Katsuhiko Ogata
* Control System Engineering –by Norman S. Nice(symbolic math toolbox)