# Performance Comparison of Electronic Printwheel System by PI and PID Controller Using Genetic Algorithms

*Sobuj Kumar Ray[1] ,Diponkar Paul[2]
[1]International University of Business Agriculture and Technology
[2] World University of Bangladesh
Corresponding Addresses
Sobuj_kumar_ray@yahoo.com, dipo0001@ntu.edu.sg

***Abstract* -** PID controller is employed in every aspect of industrial automation. The application of PID controller extends from small industry to high technology industry. For those who are in heavy industries such as refineries and ship-buildings, working with PID controller is like a routine work. We would optimize the PID controller. The PID controller was tuned by using the classical technique that has been taught to us like Ziegler-Nichols method. We make use of the power of computing world by tuning the PID in a stochastic manner. In this work it is proposed that the controller be tuned using the Genetic Algorithm technique. Genetic Algorithms (GAs) are a stochastic global search method that emulates the process of natural evolution. Genetic Algorithms have been shown to be capable of locating high performance areas in complex domains without experiencing the difficulties associated with high dimensionality or false optima as may occur with gradient decent techniques. Using genetic algorithms to perform the tuning of the controller will result in the optimum controller being evaluated for the system every time. For this study, the model selected is an Electronic Printwheel control system. The PI and PID controller have been initially tuned for printwheel system by using a classical technique Ziegler Nichols (Z-N). The same model optimizes using the GA method. The results of both designs will be compared, analyzed and conclusion will be drawn out of the simulation made.

## 1. Introduction

A control system for operating a single high-speed print wheel is disclosed. The control system includes a print wheel derive motor having a coded disc with transparent portion forming a four-level Gray code for indicating the position of the print wheel. An optical electronic system is used for positioning the print wheel and also for the purpose of providing an electronic defend which uses the motive power of the print wheel drive for holding the print wheel in selected printing position. A similar optical and electronic system is used for controlling a paper advance mechanism. The disclosed system also includes a print hummer with voice clock coil type drive to permit positive, bidirectional hammer control. A recent development in typewriters and word processors is the print wheel printer, a machine that has become a significant factor in current office machine production. With this unit typefaces are mounted on spokes projecting from a central wheel. These machines produce high quality work with limited individuality. The mode of identification is significantly different from the work of a type ball machine and in some respects from the earlier type bar typewriters. Thus it is important to distinguish between the work of print wheel machines and other classes of typewriters. Consideration is given to individual identifying characteristics; the effects of the machine's ability to produce justified or flush right margins; and the general limitations on their identification. The aim of this project is to create a PID and PI controller for the Electronic Printwheel systems that is tuned using genetic algorithms. Most system is notoriously difficult to control optimally using a PID and PI controller because the system parameters are constantly changing. It is for this reason that genetic algorithms tuning strategy was applied. Genetic Algorithms are effective at finding high performance areas in large domains and are the ideal choice to tune the PID and PI controller. Genetic Algorithms were examined in detail, it was decided to create an objective function that evaluated the optimum PID and PI gains based on the controlled systems overall error. GA's outperformed standard tuning practices, e.g. Ziegler Nichols, at designing PID and PI controllers, in the tests carried out. According to a survey for process control systems conducted in 1989, more than 90 of the control loops were of the PID type. PID control has been an active research topic for many years. Since many process plants controlled by PID controllers have similar dynamics it has been found possible to set satisfactory controller parameters from less plant information than a complete mathematical model. These techniques came about because of the desire to adjust controller parameters in situ with a minimum of effort, and also because of the possible difficulty and poor cost bent of obtaining mathematical models. The two most popular PID techniques were the step reaction curve experiment, and a closed-loop "cycling" experiment under proportional control around the nominal operating point. It has been pointed out that PID controllers can be used only for plants with relatively small time delay. When the delay constant increases; the PID controller cannot guarantee good responses. In fact, apart from the traditional PID control structure, other control strategies may also be used to deal with such case. PID control consists of three types of control,

Proportional, Integral and Derivative control [1].

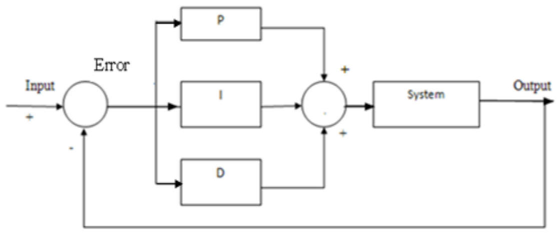


**Figure 1**. Schematic of PID Controller

The proportional controller output uses a 'proportion' of the system error to control the system. However, this introduces an offset error into the system.

$$P_{term} = K_p \times Error \tag{1}$$

The integral controller output is proportional to the amount of time there is an error present in the system. The integral action removes the offset introduced by the proportional control but introduces a phase lag into the system.

$$I_{term} = K_I \times \int Error\ dt \tag{2}$$

The derivative controller output is proportional to the rate of change of the error. Derivative control is used to reduce/eliminate overshoot and introduces a phase lead action that removes the phase lag introduced by the integral action.

$$d_{term} = K_D \times \frac{d(Error)}{dt} \tag{3}$$

The three types of control are combined together to form a PID controller with the transfer function:

$$C_{PID}(s) = \frac{K_D s^2 + K_P s + K_I}{s} \tag{4}$$

The PID controller is a "three mode" controller. That is, its activity and performance is based on the values chosen for three tuning parameters, one each nominally associated with the proportional, integral and derivative terms.
The block diagram of a closed–loop system with a PID controller in direct path [2] as shown in figure 2

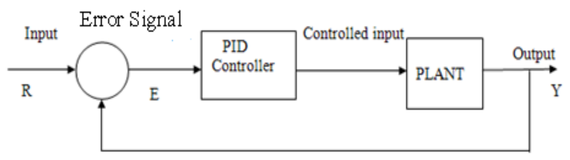


**Figure 2**. Block diagram of PID controller.
As the name suggests, the PI algorithm consists of two basic modes, the Proportional mode, and the Integral mode .When utilizing this algorithm it is necessary to decide which modes are to be used (Por I) and then specify the parameters (or settings) for each mode used. Generally, two basic algorithms are used P or PI.

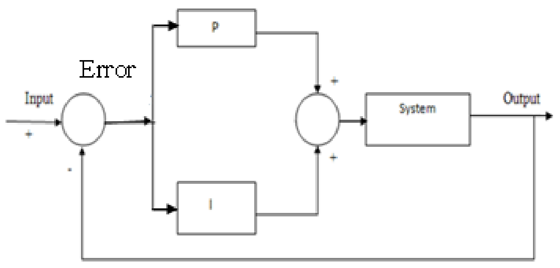


**Figure 3**. Schematic of PI Controller

The mathematical representation is,

$$\frac{mv(s)}{e(s)} = K_c \tag{5}$$

(Laplace domain) or $mv(t) = mv_{ss} + K_c(t)$ (time domain) the proportional mode adjusts the output signal in direct proportion to the controller input (which is the error signal, e). The adjustable parameter to be specified is the controller gain, $k_c$ .This is not be confused with the process gain, $K_p$. The larger $k_c$ the more the controller output will change for a given error. For instance, with a gain of 1 an error of 10% of scale will change the controller output by 10% of scale. Many instrument manufacturers use Proportional Band (PB) instead of $k_c$ the time domain expression also indicates that the controller requires calibration around the steady-state operating point. This is indicated by the constant term $mv_{ss}$. This represents the 'steady-state' signals for the mv and is used to ensure that at zero error the cv is at set point. In the Laplace domain this term disappears, because of the 'deviation variable' representation. A proportional controller reduces error but does not eliminate it (unless the process has naturally integrating properties), i.e. an offset between the actual and desired value will normally exist
The mathematical representation is,

$$\frac{mv(s)}{mv(t)} = K_C[1 + \frac{1}{T_I s}] \tag{6}$$

$$mv(t) = mv_{ss} + K_C[e(t) + \tfrac{1}{T_I s}\int e(t)dt] \tag{7}$$

The additional integral mode (often referred to as reset) corrects for any offset (error) that may occur between the desired value (set point) and the process output automatically over time. The adjustable parameter to be specified is the integral time ($T_i$) of the controller.
• Drive a certain distance in a straight line using encoders.
• Maintain position by maintaining a certain encoder tick count, this causes the motors to fight back against being pushed with the precise power output required.
• Driving tracking/shooting gimbals using the camera in the 2006 game.
• Maintain rotational velocity of impeller or collector wheels for balls by adjusting speed based on a target number of encoder ticks over time.
• Precision position of manipulators using encoders or potentiometers.
A method and apparatus for providing variable print hammer energy information and variable character spacing information for every character of every font in an electronic typing system having interchangeable print wheels, wherein the individual print wheels carry self-descriptive information in coded form on a read-only memory. The descriptive information is encoded on a portion of the print wheel and contains, in high density, machine readable, permanent form, and sufficient coded data to instruct the electronic typing system as to the optimal use of the particular font of characters contained on the type wheel. In a preferred embodiment, a single initializing revolution of the print

wheel upon each insertion or machine start-up cycle serves to load the encoded data into a read/write memory for subsequent call-up and use by the electronic typing system during the print-out of each character. (Dingyu Xue,YangQuan Chem,and Derek P.Athenton, 2007) The data is serially encoded on the print wheel for reading by single track optional sensing apparatus. Alternate embodiments using parallel data tracks and magnetic sensing apparatus are discussed.

In control system position-control is quite common, that has variable load inertia. For example, the load inertia seen by the motor in an electronic printer wheel change when different printwheels are used. To illustrate the design of a robust system that is insensitive to the variation of the load inertia, consider that the forward path transfer function of a unity feedback control system [2]

$$G_P(s) = \frac{K_i K_b}{s[(Js+B)(Ls+B)+K_i K_b]} \tag{8}$$

The system parameters are

$K_i$ = motor torque constant =1N-m/A

$K_b$ = motor back emf constant =1v/rad/sec

R= motor resistant =1 Ω

L=motor inductance =0.01H

B= motor and load viscous-friction coefficient =0

J =motor and load inertia, varies between 0.02 and 0.02 N-m/rad/ sec $^2$

K =amplifier gain

Substituting these parameters into (8) we get

For $J = 0.01$

$$G_p(s) = \frac{10000K}{s(s^2+100s+10000)} \tag{9}$$

For $J = 0.02$;

$$G_p(s) = \frac{5000K}{s(s^2+100s+5000)} \tag{10}$$

## 2. Method

Genetic Algorithms (GA's) are a stochastic global search method that mimics the process of natural evolution. The genetic algorithm starts with no knowledge of the correct solution and depends entirely on responses from its environment and Evolution operators (i.e. reproduction, crossover and mutation) to arrive at the best solution. By starting at several independent points and searching in parallel, the algorithm avoids local minima and converging to sub optimal solutions. In this way, GAs have been shown to be capable of locating high performance areas in complex domains without experiencing the difficulties associated with high dimensionality, as may occur with gradient decent techniques or methods that rely on derivative information [5]. A genetic algorithm is typically initialized with a random population consisting of between 20-100 individuals. This population (mating pool) is usually represented by a real-valued number or a binary string called a chromosome. For illustrative purposes, the rest of this section represents each chromosome as a binary string. How well an individual performs a task is measured is assessed by the objective function (D. E. Goldberg, 1989). The objective function assigns each individual a corresponding number called its fitness. The fitness of each chromosome is assessed and a survival of the fittest strategy is applied. In this project, the magnitude of the error will be used to assess the fitness of each chromosome. There are three main stages of a genetic algorithm; these are known as reproduction, crossover and mutation. Genetic Algorithms provides an adaptive searching mechanism inspired on Darwin's principle of the fittest. It is invented by John Holland of the university of Michigan, after David Goldberg gave a basic idea of GA in his book '' Genetic Algorithm Search, Optimization and Matching Learning'' .GA is a search and optimization techniques inspired by biological process namely. 'natural' selection' and natural genetics'. GA starts with no knowledge of correct solution and depends on responses from its environment. GA manipulates not just one potential solution to a problem, but a collection of potential solution, called a population. The potential solution in population is called 'individuals' or 'chromosomes', each of them is associated to a fitness value. The chromosomes are subjected to an evolutionary process which takes several cycles. Basic operations are selection, reproduction, crossover, and mutation. During crossover some reproduced individuals cross and exchange their genetic characteristics and such crossover create new chromosomes from the existing one s in the population. The selection mechanism for parent chromosomes takes the fitness of parent into account, ensuring that the better solution have a higher chance to procreate and donate their beneficial characteristics to their offspring. Newly generated individuals in time replace the existing ones. Through this process after a while the population will converge to a 'best' solution.

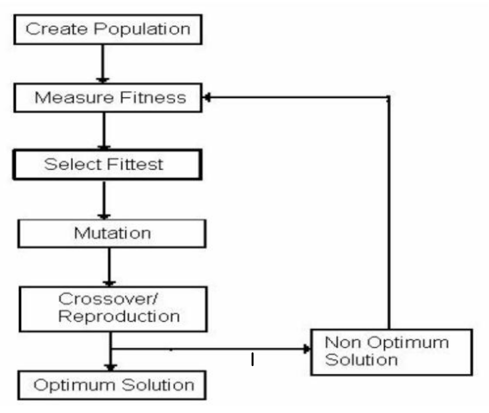


**Figure 4**. Graphical Illustration the Genetic Algorithm Outline

Whom it points is selected. This continues until the selection criterion has been met. The probability of an individual being selected is thus related to its fitness, ensuring that fitter individuals are more likely to leave offspring. Multiple copies of the same string may be selected for reproduction and the fitter strings should begin

More complex crossover techniques exist in the form of Multi-point and Uniform Crossover Algorithms. Multi-point crossover is an extension of the single point crossover algorithm and operates on the principle that the parts of a chromosome that contribute most to its fitness might not be adjacent. There are three main stages involved in a Multi-point crossover.

1. Members of the newly reproduced strings in the mating pool are 'mated' (paired) at random.
2. Multiple positions are selected randomly with no duplicates and sorted into ascending order.
3. The bits between successive crossover points are exchanged to produce new offspring.

Example: If the string 11111 and 00000 were selected for crossover and the multipoint crossover positions were selected to be 2 and 4 then the newly created strings will be 11001 and 00110 as shown in Figure 5.

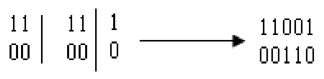


**Figure 5.** Illustration of a Multi-Point Crossover

In uniform crossover, a random mask of ones and zeros of the same length as the parent strings is used in a procedure as follows.
1. Members of the newly reproduced strings in the mating pool are 'mated' (Paired) at random.
2. A mask is placed over each string. If the mask bit is a one, the underlying bit is kept. If the mask bit is a zero then the corresponding bit from the other string is placed in this position.

Example: If the string 10101 and 01010 were selected for crossover with the mask
10101 then newly created strings would be 11111 and 00000 as shown in Fig. 6.

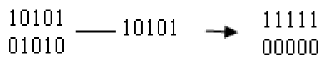


**Figure 6**. Illustration of a Uniform Crossover

Uniform crossover is the most disruptive of the crossover algorithms [4] and has the capability to completely dismantle a fit string, rendering it useless in the next generation. Because of this Uniform Crossover will not be used in this project.
The following schematic diagram illustrates the three types of children.

### 2.1 Mutation
Using selection and crossover on their own will generate a large amount of different strings however there are two main problems with this:
1) Depending on the initial population chosen, there may not be enough diversity in the initial string to ensure the GA searches the entire problem space.
2) The GA may converge on sub-optimum string due to a bad choice of initial population.

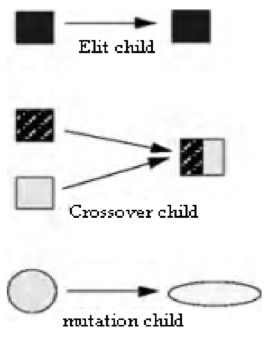


**Figure 7**: Schematic diagram illustrates the three types of children

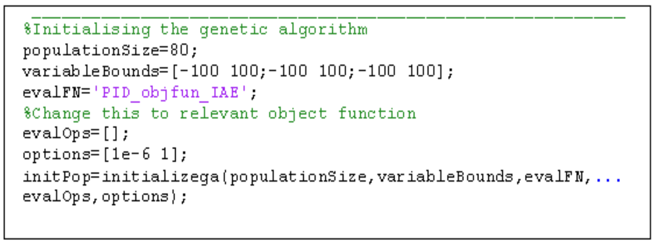
This problem may be overcome by the introduction of a mutation operator into the GA. Mutation is the occasional alteration of a value of a string position. It is considered a back ground operator in the genetic algorithm.
The probability of mutation is normally low because a high mutation rate would destroy fit of
Mutation and degenerate the genetic algorithm into a random search. Mutation probability values of around 0.1% to 0.01% are common, these values represent the probability that a certain string will be selected for mutation, that is for a probability of 0.1%: one string in one thousand will be selected for mutation ( C. R. Houck, J. Joines. and M.Kay., 1996). Once a string is selected for mutation, a randomly chosen element of the string is changed or 'mutated'.
For example, if the GA chooses bit position 4 for mutation in the binary string 10000, the resulting string is 10010 as the fourth bit in the string is flipped as shown in Figure 8

10000
10010

**Figure 8.** Illustration of Mutation Operation

### 2.2 Elitism
With crossover and mutation taking place, there is a high risk that the optimum solution could be lost as there is no guarantee that these operators will preserve the fittest string. To counteract this, elitist models are often used. In an elitist model, the best individual from a population is saved before any of these operations take place. After the new population is formed and evaluated, it is examined to see if this best structure has been preserved. If not, the saved copy is reinserted back into the population. The GA then continues on as normal [6] Fig 9 Initializing the Population of the Genetic Algorithm
The following code is based on the Genetic Algorithm Optimization Toolbox
(GAOT) [3].

- ***PopulationSize*** **-** The first stage of writing a Genetic Algorithm is to create a population. This command defines the population size.



```
%Initialising the genetic algorithm
populationSize=80;
variableBounds=[-100 100;-100 100;-100 100];
evalFN='PID_objfun_IAE';
%Change this to relevant object function
evalOps=[];
options=[1e-6 1];
initPop=initializega(populationSize,variableBounds,evalFN,...
evalOps,options);
```

**Figure 9**. Codes Initializing the Population of a Genetic Algorithm

· ***VariableBounds* -** Since this project is using genetic algorithms to optimize the gains of a PID controller there are going to be three strings assigned to each member of the population, these members will be comprised of a P, I and a D string that will be evaluated throughout the course of the GA. The three terms are entered into the genetic algorithm via the declaration of a three-row variablebounds matrix. The number of rows in the variablebounds matrix represents the number of terms in each member of the population. Figure 9 illustrates a population of eighty members being initialized with values randomly selected between -100 and 100.

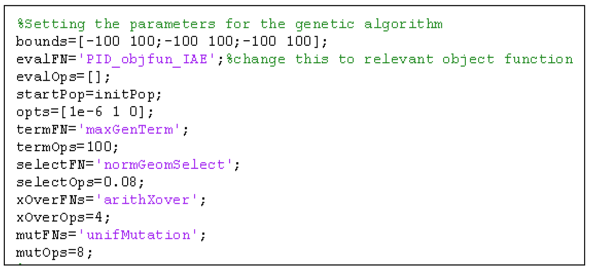
· ***EvalFN* -** The evaluation function is the declaration of the file name containing the objective function.

· ***Options* -** Although the previous examples in this section were all binary encoded,
this was just for illustrative purposes. Binary strings have two main drawbacks:
1. They take longer to evaluate due to the fact they have to be converted to/from binary.
2. Binary strings lose precision during conversion.
As a result of this and the fact that they use less memory, real (floating point) numbers will be used to encode the population. This is signified in the options command in Figure 9, where the '1e-6' term is the floating point precision and the '1' term indicates that real numbers are being used (0 indicates binary encoding is being used).

· **Initialisega** - This command combines all the previously described terms and creates an initial population of 80 real valued members between –100 and 100 with 6 decimal place precision.

## 2.3 Initializing the Population Genetic Algorithm

A genetic algorithm is initialized as shown in Figure 10.



```
%Setting the parameters for the genetic algorithm
bounds=[-100 100;-100 100;-100 100];
evalFN='PID_objfun_IAE';%change this to relevant object function
evalOps=[];
startPop=initPop;
opts=[1e-6 1 0];
termFN='maxGenTerm';
termOps=100;
selectFN='normGeomSelect';
selectOps=0.08;
xOverFNs='arithXover';
xOverOps=4;
mutFNs='unifMutation';
mutOps=8;
```

**Figure 10**. Initializing the Genetic Algorithm

***Bounds*** - The bounds for the genetic algorithm to search within are set using this command. These bounds may be different from the ones used to initials the population and they define the entire search space for the genet algorithm. **startPop -** The starting population of the GA, 'startPop', is defined as the population described in the previous section, i.e. 'initPop', see Figure 4.12.opts - The options for the Genetic Algorithm consist of the precision of the string values i.e. 1e-6, the declaration of real coded values, 1, and a request for the progress of the GA to be displayed, 1, or suppressed, 0.· ***TermFN* -** This is the declaration of the termination function for the genetic algorithm. This is used to terminate the genetic algorithm once certain criterion has been met. In this project, every GA will be terminated when it reaches a certain number of generations using the 'maxGenTerm' function. This termination method allows for more control over the compile time (i.e. the amount of time it takes for the genetic algorithm to reach its termination criterion) of the genetic algorithm when compared with other termination criteria e.g. convergence termination criterion. · ***TermOps* -** This command defines the options, if any, for the termination function. In this example the termination options are set to 100, which mean that the GA will reproduce one hundred generations before terminating. This number may be altered to best suit the convergence criteria of the genetic algorithm i.e. if the GA converges quickly then the termination options should be reduced. · ***SelectFN* -** Normalized geometric selection ('normGeomSelect') is the primary selection process to be used in this project. The GAOT toolbox provides two other selection functions, Tournament selection and Roulette wheel selection. Tournament selection has a longer compilation time than the rest and as the overall run time of the genetic algorithm is an issue, tournament selection will not be used. The roulette wheel option is inappropriate due to the reasons mentioned in section. · ***SelectOps* -** When using the 'normGeomSelect' option, the only parameter that has to be declared is the probability of selecting the fittest chromosome of each generation, in this example this probability is set to 0.08. · *XOverFN* - Arithmetic crossover was chosen as the crossover procedure. Single point crossover is too simplistic to work effectively on a chromosome with three alleles, a more uniform crossover procedure throughout the chromosome is required. Heuristic crossover was discarded because it performs the crossover procedure a number of times and then picks the best one. This increases the compilation time of the program and is undesirable. The Arithmetic crossover procedure is specifically used for floating point numbers and is the ideal crossover option for use in this project. · *XOverOptions* -This is where the number of crossover points is specified. In the example shown in Figure 5.8, the number of crossovers points is set to four.. The 'multiNonUnifMutation', or multi non-uniformly distributed mutation operator, was chosen as the mutation operator as it is considered to function well with multiple variables.. The mutation operator takes in three options when using the'multiNonUnifMutation' function. The first is the total number of mutations, normally set with a probability of around 0.1%. The second parameter is the maximum number of generations and the third parameter is the shape of the distribution. This last parameter is set to a value of two, three or four where the number reflects the variance of the distribution.5.9 performing the Genetic Algorithm The genetic algorithm is compiled using the command shown in Figure 4.13. Once this command is entered, the genetic algorithm will iterate until it fulfills the criteria described by its termination function. Writing an objective function is the most difficult part of creating a genetic Algorithm. An objective function could be created to find a PID controller that gives the smallest overshoot, fastest rise time or quickest settling time but in order to combine all of these objectives it was decided to design an objective function that will minimize the error of the controlled system. Each chromosome in the population is passed into the objective function one at a time. The chromosome is then evaluated and assigned a number to represent its fitness, the bigger its number the better its fitness. The genetic algorithm uses the chromosome's fitness value to create a new population consisting of the fittest members.

## 3. Simulation Result

Print wheel control system for resetting and setting a plurality of print wheels, said control system having means for generating timing pulses in accordance with the rotation of the print wheels, a pulse signal distributor for generating control signals associated with successive characters on each of the print wheels, and a selector network and controls responsive to the control signals for arresting the movement of each of the print wheels, depending upon the character of each of the print wheels to be selected.

### 3.1 Development Electronic Print Wheel System

To aid with the development of this project a system was chosen at random and a PID and PI controller was designed for it using conventional methods. A genetic algorithm was then created to evaluate the PID and PI coefficients of the same system and the results of the two techniques were compared. The system was selected as position control (Electronic print wheel system) [2] system is of order three. The system chosen was:

$$G(s) = \frac{5000}{s^3 + 100s^2 + 5000s} \tag{11}$$

### 3.2 Ziegler-Nichols Designed PID and PI Controller

The Ziegler-Nichols tuning method using root-locus was the 'conventional' method used to evaluate the PID and PI gains for the system. Using the 'rlocus' command in Matlab,The crossover point and gain of the system were found to be j71.1 and 101 respectively, as shown in Figure 11

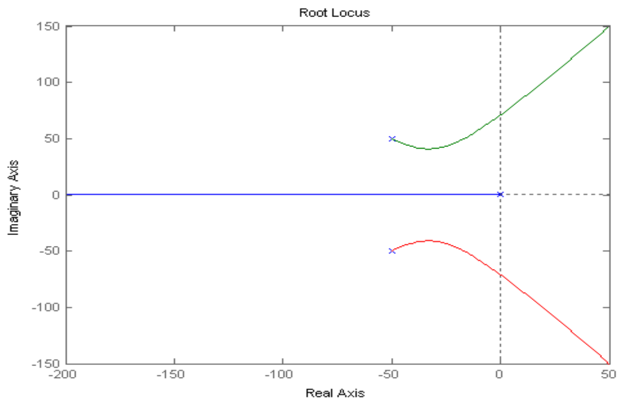


**Figure 11.** Plot of root locus for $G(s)$

With a frequency ($\omega_c$) of 1rad/s the period $T_c$ is calculated as,

$$T_c = \frac{2\pi}{\omega_c} \text{ Sec} \tag{12}$$

**Table 1.** Ziegler-Nichols PID and PI Tuning Parameters Gives

| Controller | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| PID | $0.6K_c$ | $\frac{T_c}{2}$ | $\frac{T_c}{8}$ |
| PI | $0.45K_c$ | $\frac{T_c}{1.2}$ | |

**Table 2.** Ziegler-Nichols PID Tuning Values

| Controller | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| PID | 60.60 | 0.044 | 0.011 |
| PI | 31.99 | 0.073 | |

Using the relationship $K_I = \frac{K_p}{T_I}$ and $K_D = K_P T_D$, the PID and PI gain can be evaluated.

**Table 3.** Ziegler-Nichols PID and PI Gain values

| Controller | $K_P$ | $K_I$ | $K_D$ |
|---|---|---|---|
| PID | 60.60 | 1377.27 | 0.66 |
| PI | 31.99 | 434.39 | |

Table 3 shows the PID and PI gain values for the system G(s). A genetic algorithm, Initial PID GA. Initial PI GA. was created to evaluate the optimum PID and PI gain values for the system G(s). A number of objective functions were created in order to evaluate the PID and PI values chosen by the Genetic Algorithm.

Comparisons of Steady Step Response of Integral of Time Multiplied by Absolute Error (ITAE)

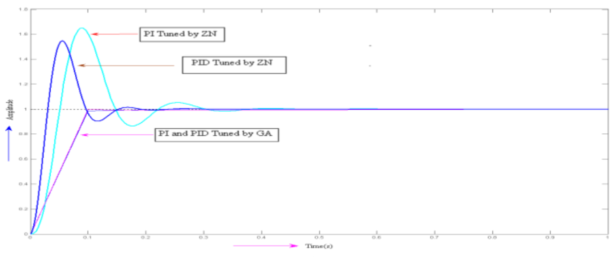


**Figure 12.** Step Response of Integral of Time Multiplied by Absolute Error (ITAE)

Comparisons of Steady Step Response Integral of Absolute Magnitude of the Error (IAE)

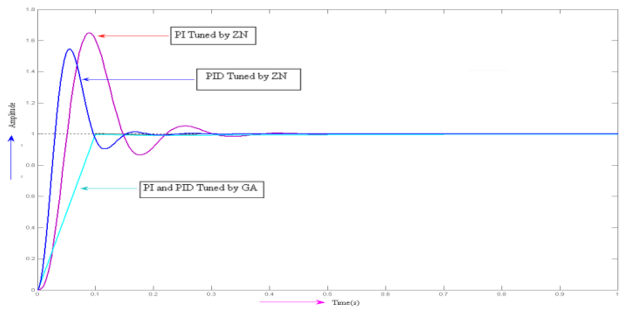


**Figure 13.** Step Response Integral of Absolute Magnitude of the Error (IAE)

Comparisons of Steady Step Response Integral of the Square of the Error (ISE)

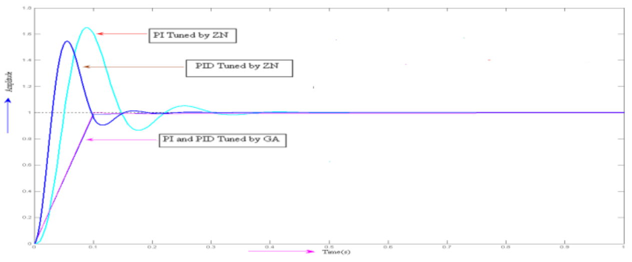


**Figure 14**. Step Response Integral of the Square of the Error (ISE)

Comparisons of Steady Step Response Mean of the Square of the Error (MSE)

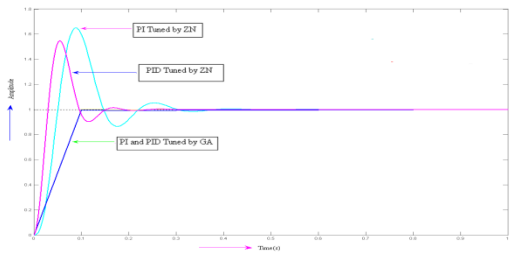


**Figure15.** Step Response Mean of the Square of the Error (MSE)

## 4. Discussion

Figure 12 to 15 shows Ziegler-Nichols designed PID and PI controller Vs GA designed PID and PI Controller using ITAE, IAE, ISE and MSE as performance Criterion. Under the conditions of this experiment, it can be seen that the IAE Objective functions performs having a smaller rise time, smaller Overshoot and smaller settling time than the other PI controllers (Ms Jennifer Bruton, 2003). Again the ITAE Objective functions performs having a smaller rise time, smaller Overshoot and smaller settling time than other PID controllers. Each of the genetic algorithm-tuned PID and PI controllers outperforms the Ziegler-Nichols tuned controller in terms of rise time, overshoot and settling time. Each of the PID controller performance is more than PI in terms of rise time, overshoot and settling time. The ITAE objective function was chosen as the primary performance criterion for the remainder of this project due to its smaller settling time and smaller overshoot than any other method in conjunction with a slightly faster compile time due to there being just one multiplication to be carried after the error has been calculated. This is coupled with the fact that ITAE has been a proven measure.

## 5. Conclusion and discussion

It was established that the steady state characteristics of GA's outperformed standard tuning practices when designing a PID and PI controller. It was determined that the Steady Step Response of Integral of Absolute Magnitude of the Error **(IAE)** performance criterion based objective function produced the most effective PI controllers when compared with other performance criterion i.e. MSE, ITAE and ISE. Again Integral of Time Multiplied by Absolute Error **(ITAE)** performance criterion based objective function produced the most effective PID controllers when compared with other performance criterion i.e. IAE, MSE, and ISE. It was proved by comparison of their steady state characteristics that PID outperformed standard tuning practices than PI controller. When testing the genetic algorithm component, it was discovered to frequently produce controllers that made the overall controlled system unstable. The exact cause for this could not be determined. To rectify this problem, the genetic algorithm was modified so it would analyze the controller it evaluates. If the controller produces an unstable system, it is replaced with the last stable controller evaluated by the genetic algorithm. The genetic algorithm online tuned PID controller proved to be a capable controller. Adequate testing of the controller could not be performed due to the simulation difficulties mentioned previously. It is illustrative of the fact that we have worked on a transfer function which is fixed for printwheel system parameter. But in real life its parameter is not constant. Again we have used only step response but ramp response, impulse etc could be used as the remarkable scope for the future work on it. This process can be applied to many other control systems such as ball and hoop control system, sun seeker system etc.

## References

[1] Ms Jennifer Bruton , Ian Griffin" On-line PID Controller Tuning using Genetic Algorithms" 2003.

[2] Linear Feedback Control by Dingyu Xue,YangQuan Chem,and Derek P.Athenton Chapter 6,Copyright @ 2007

[3] U.S Patent March 14,1978 Sheet 7 of 7 4,078,485 Print wheel control John Gilkeson Guthrie http://patents?id=hjIrAAAAEBAJ&printsec=abstract&zoom=4&source=gbs_overview_r&cad=0#v=onepage&q=&f=false

[4] Automatic Control systems, 7th Ed. By Benjamin C.Kuo

[5] O' Mahony, T., Downing,C.J. and Klaudiuz, F., 'Genetic Algorithms for PID Parameter Optimisation: Minimising Error Criteria', [online], URL:

http://www.pwr.wroc.pl/~i-8zas/kf_glas00.pdf

[6] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Co., Inc., 1989

[7] C. R. Houck, J. Joines. and M.Kay. A genetic algotithm for function optimisation: A Matlab implementation. ACM Transactions on Mathematical Software, 1996, [Online], URL: http://www.eos.ncsu.edu/eos/service/ie/research/kay_res/GA ToolBox/gaot

## First Author's Biography

Mr. Sobuj Kumar Ray was born in Bogra, Bangladesh in 1987. Mr. Ray received his Bachelor degree in Electrical and Electronic Engineering from the Rajshahi University of Engineering and Technology (RUET), Rajshahi, Bangladesh in April 2010. Now he is a faculty in the department of Electrical and Electronic Engineering, Internal University of Business Agriculture and Technology (IUBAT), Uttara,Dhaka, Bangladesh(www.iubat.edu). The major fields of study of Mr. Ray comprise control system and power system.

## Second author's Biography

Mr. Diponkar Paul is currently working as Assistant Professor in the department of Electrical and Electronic engineering at Stamford University Bangladesh. After passing his master degree from March 2008 he was serving as Assistant Professor, EEE at Bangladesh University upto July 2010. He is having qualifications: B.Sc. Engg., DISM (software engineering), M.Sc. Engg. His research interests are in the area of energy conversions, power system modeling and advanced control theories covering the application of IT. From 0ct 2004 to July 2006, he was working as Lecturer in department of computer science and engineering at Pundra University of science & technology, Bogra. In Singapore during his master degree at Nanyang technological university, he was involved in financial service operation integrated to IT system administration jobs from Dec 2006 to February 2008.