Research Article GU J Sci, Part A, 10(3): 286-300 (2023) 10.54287/gujsa.1342905

Gazi University
Journal of Science
PART A: ENGINEERING AND INNOVATION
http://dergipark.org.tr/gujsa

Genetic Algorithm based PID Tuning Software Design and Implementation for a DC Motor Control System

Zafer ORTATEPE[1*]

[1]*Pamukkale University, Department of Automotive Engineering, Denizli, Türkiye*

Keywords

DC Motor

Genetic Algorithm

PID Tuning

Abstract

This study presents the software and implementation for proportional-integral-derivative (PID) tuning of a DC motor control system using genetic algorithm (GA). The PID parameters for a specific control structure are optimized using GA in the proposed tuning procedure. Also, integral time absolute error (ITAE) is used as a fitness function to optimize the parameters. The robustness of the control system is compared with conventional mathematical method. Simulations are carried out in MATLAB/Simulink to compare the results of a DC motor control system. Simulation results show that in terms of overshoot, steady-state error, and settling time, GA-based PID tuning approach performed better than conventional method. Additionally, a sensitivity analysis is performed to evaluate how robust the proposed approach is to parameter variations. The analysis shows that compared to the conventional method, the GA-based PID tuning algorithm is more adaptable to variations in system parameters.

Cite

Ortatepe, Z. (2023). Genetic Algorithm based PID Tuning Software Design and Implementation for a DC Motor Control System. *GU J Sci, Part A, 10*(3), 286-300. doi:10.54287/gujsa.1342905

Author ID (ORCID Number)

0000-0001-7771-1677 Zafer ORTATEPE

Article Process

Submission Date 14.08.2023
Revision Date 29.08.2023
Accepted Date 01.09.2023
Published Date 21.09.2023

1. INTRODUCTION

Control systems play a significant role in many fields, from automotive to aerospace, robotics to industrial process control. One of the most used control methods in industry is proportional-integral-derivative (PID) controller. PID control, a closed-loop feedback technique, measures the error between a desired set point and the system's actual output continually and then modifies the control input to minimize this error. To maintain system stability and performance, it is essential to fine-tune of the PID parameters. However, conventional PID tuning methods can be time-consuming and require expert knowledge, making it challenging to tune the PID controller for complex systems (Borase et al., 2021).

In literature, a number of conventional PID tuning methods have been proposed, including Ziegler-Nichols (ZN) (Patel, 2020), Cohen-Coon (Taşören, 2021), and Tyreus-Luyben (Ibrahim et al., 2016) methods. These methods require expert knowledge of the system, and the tuning process can be time-consuming and challenging for complex systems.

To solve these drawbacks, some optimization techniques have been proposed to tune PID controllers, for instance particle swarm optimization (PSO) (Aranza et al., 2016) and differential evolution (DE) (Saad et al., 2012). A population-based optimization algorithm called PSO imitates its cooperative behavior of a swarm of particles. The PSO-based PID tuning method has been applied to various control systems, such as DC motor control (Alruim Alhasan & Güneş, 2017) and chemical processes (Fang et al., 2021). The other population-based optimization technique called DE imitates the natural process of crossover and mutation. DE-based PID

*Corresponding Author, e-mail: zortatepe@pau.edu.tr

tuning methods have been applied to various control systems, such as HVAC systems (Rout et al., 2013) and mobile robots (Martinez-Soltero & Hernandez-Barragan, 2018).

GA is one of the most widely used optimization technique in engineering, including in control system design. It has been successfully implemented in various applications due to their ability to search a large parameter space and find optimal solutions (Malhotra et al., 2011). Moreover, GA-based PID tuning methods can automatically generate optimal PID parameters that achieve the desired system performance. The GA-based PID tuning method can handle complex systems and requires less expert knowledge, making it an attractive alternative to conventional PID tuning methods (de Figueiredo et al., 2023).

Different GA-based PID tuning techniques have been proposed by researchers. For instance, Islam et al. (2020) propose a GA-based PID tuning method that uses fuzzy logic to adjust crossover and mutation ratios. Method is implemented for a DC motor and the results showed that the transient-state response and steady-state error are reduced. Similarly, Jayachitra and Vinodha (2014) propose a GA-based PID tuning method that uses a weighted combination of objective functions. The method is applied to continuous stirred tank reactor and combination of all objective functions demonstrated its effectiveness in improving system performance. Tiwari et al. (2018) propose a PID control method for the DC motor control system that is based on a GA. MATLAB/Simulink is used to simulate performance of the system, and GA optimization is implemented using the MATLAB toolbox. The results showed that GA-based tuning method significantly improved the performance of control system compared to conventional ones. Moreover, in a DC motor control system, a hybrid optimization strategy for PID controller tuning is proposed by Flores-Morán et al. (2020). To improve the search capabilities of the optimization process, the GA method is combined with the PSO algorithm. The results showed that, in comparison to ZN approach, the hybrid optimization method greatly enhanced performance of DC motor control system. GA-based PID tuning approach is proposed by Galotto et al. (2007). For estimating the unknown parameters of the DC motor and fine-tuning the PID controller parameters, the authors employ the recursive least squares (RLS) approach. Results shows that the proposed method successfully improved the performance of the DC motor control system by optimizing the PID controller parameters. Korkmaz et al. (2012) propose variable parameter nonlinear PID controller and GA based PID controller to compare the results of the proposed controllers. According to the results, it can be deduced that both control methods are superior to the conventional ZN control method. However, it is seen that the GA-based control method gives slightly better results than the variable parameter nonlinear PID controller. Pereira and Pinto (2005) propose GA optimization technique to define the system identification and parameter tuning for the optimum adaptive control. In proposed system two independent GA are used respectively. When the results of the cascaded GA are compared with the conventional ZN method, it is emphasized that the GA-based system is superior. Wati and Hidayat (2013) propose GA-based PID parameters optimization for air heater temperature control. The experimental results demonstrate that the step response of GA-based PID controller performs better than conventional ZN tuning method. Meena and Devanshu (2017) propose a PID tuning method for process control by using GA. Comparing the proposed GA tuned plant's performance to that of a conventionally tuned plant, it can be seen that the GA tuned PID controller outperforms the conventionally designed PID controller.

Major gap in the literature regarding this subject is the lack of presentation of the full algorithms used in the above studies. This paper presents GA-based PID tuning method and its implementation for a DC motor control system. The proposed GA based tuning software is completely presented in this work, which is the key element that separates this study from others in the literature. The software is written in MATLAB, and the DC motor implementation has been done in MATLAB/Simulink.

The rest of the paper is arranged as follows: section 2 describes the mathematical model and transfer function of the DC motor. Section 3 analyzes conventional PID control method. Section 4 discusses the proposed GA-based PID tuning method and Section 5 gives its fully software. Section 6 describes the manual solution for single iteration. Section 7 gives the implementation of the proposed method for a DC motor control system. Finally, section 8 concludes the paper.

2. MATHEMATICAL MODEL OF DC MOTOR

A DC motor is a device that converts electrical energy into mechanical energy through its interaction of a magnetic field and current-carrying conductors. The DC motor can be modeled using mathematical equations that describe its behavior in terms of its physical parameters. Fundamental DC motor model is given in Figure 1. Mathematical model of DC motor in Laplace domain can be represented by the following equations:

$$e_a(s) = R_a I_a(s) + sL_a I_a(s) + e_b(s) \tag{1}$$

$$T_m(s) = K_i I_a(s) \tag{2}$$

$$T_m(s) = sJ_m\Omega_m(s) + B_m\Omega_m(s) + T_L(s) \tag{3}$$

$$e_b(s) = K_b\Omega_m(s) \tag{4}$$

The equations can be written as follows when the input and output of the system are defined as $e_a(s)$ and $\Omega_m(s)$.

$$I_a(s) = \frac{e_a(s) - e_b(s)}{sL_a + R_a} \tag{5}$$

$$\Omega_m(s) = \frac{T_m(s) - T_L(s)}{sJ_m + B_m} \tag{6}$$

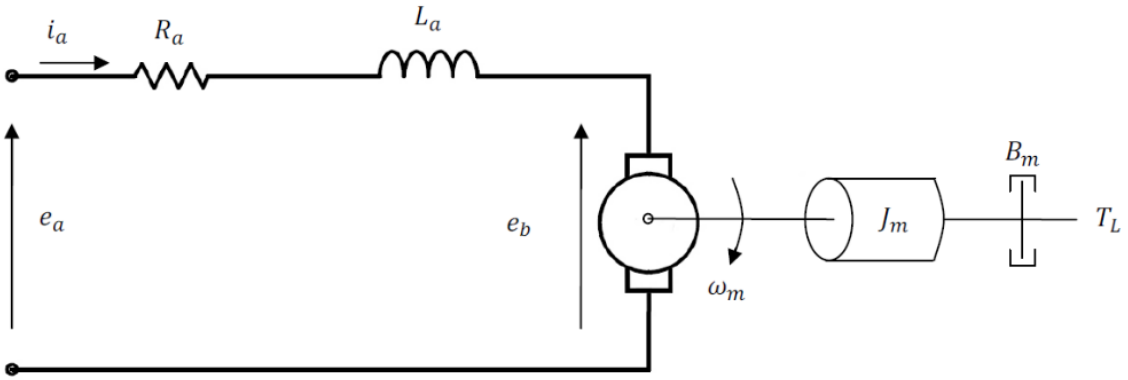

Figure 1. *DC motor model*

In the light of these equations, the DC motor control block diagram is shown in Figure 2.

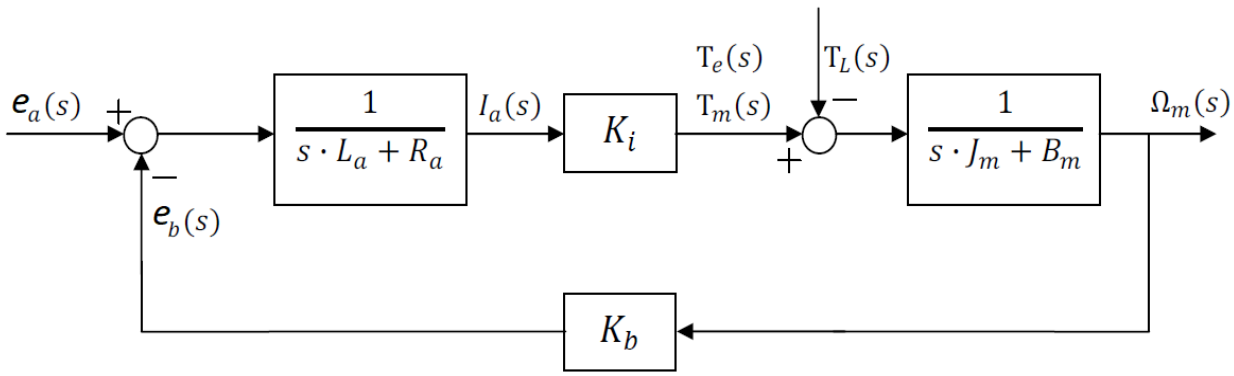

Figure 2. *Block diagram of the DC motor*

The transfer function of the DC motor must be modified to include the controller's effects when designing a controller for a DC motor control system. The resulting transfer function is:

$$\frac{\Omega_m(s)}{e_a(s)} = \frac{K_i}{s^2 J_m L_a + s(J_m R_a + L_a B_m) + R_a B_m + K_i K_b} \tag{7}$$

The electrical time constant is assumed to be zero due to the mechanical time constant of the system is much larger than the electrical time constant. So, if $L_a/R_a=0$, it must be $L_a=0$. In this case, overall transfer function of system is as follows:

$$\frac{\Omega_m(s)}{e_a(s)} = \frac{K_i}{sJ_m R_a + R_a B_m + K_i K_b} \tag{8}$$

Since it is often assumed that $K_T = K_i = K_b$ in practice, the equation can be written as:

$$\frac{\Omega_m(s)}{e_a(s)} = \frac{K_T}{sJ_m R_a + R_a B_m + {K_T}^2} \tag{9}$$

The system is similar to the general transfer function of the first-order system. So, let's bring the system to the general transfer function form of first-order systems with various arrangements.

$$\frac{K}{\tau s + 1} \cong \frac{\frac{K_T}{{K_T}^2 + R_a B_m}}{\frac{J_m R_a}{{K_T}^2 + R_a B_m} s + 1} \tag{10}$$

where K is the gain $K \cong \frac{K_T}{{K_T}^2 + R_a B_m}$ and τ is the time constant $\tau \cong \frac{J_m R_a}{{K_T}^2 + R_a B_m}$. The DC motor parameters used in this paper is given in Table 1.

Table 1. *DC motor parameters used in this study*

Parameters	Symbols	Values
Motor resistance	R_a	3.2Ω
Motor inductance	L_a	0H
Inertia moment	J_m	0.082 $kg.m^2$
Viscous friction coefficient	B_m	0.0275 $kg.m^2/s$
Motor torque constant	K_i	1.26 *N.m/A*
EMF constant	K_b	1.26 *V.s/rad*

As a result, the following is the DC motor's transfer function:

$$G_{S_s} = \frac{\Omega_m(s)}{e_a(s)} = \frac{0.752}{0.157s + 1} \tag{11}$$

3. PROPORTIONAL INTEGRAL DERIVATIVE (PID) CONTROL OF DC MOTOR

Due to their ability to precisely adjust speed and torque, DC motors are widely used in industrial applications. PID control is a popular technique for controlling motor speed. Besides, three elements are used in a closed loop PID control system: proportional, integral, and derivative. Proportional component adjusts the input voltage to the motor in proportion to the error between the desired and actual motor speed. Integral component integrates the error over time to compensate for any steady-state error. The motor's input voltage is changed by the derivative component to correspond with a measurement of the error's rate of change. PID controller transfer function is provided by:

$$G_{c_s} = K_P + \frac{K_I}{s} + K_D s \tag{12}$$

where K_P, K_I, and K_D are the proportional, integral, and derivative gains, respectively. "s" is the Laplace domain variable. A first-order system can be controlled by PID algorithm as if it is a second-order system, according to criteria such as desired overshoot and settling time. The criteria required in the controller design are determined as follows: Mp = %5 (overshoot ratio), Ts=0.4 s (settling time), ess = 0.02 (steady state error for ramp input signal). The formulations of these criteria are given as follows:

$$\%M_p = e^{-\frac{\xi}{\sqrt{1-\xi^2}}\pi}.100 \tag{13}$$

$$T_s = \frac{4}{\sigma} \qquad \sigma = \xi\omega_n \tag{14}$$

where ξ is the damping rate and ω_n is the damping frequency. When equations (13) and (14) are solved, $\xi = 0.69$ and $\omega_n = 14.4927$ are calculated. All of the system poles is given below:

$$s_{1,2} = -\xi\omega_n \mp \omega_n\sqrt{\xi^2 - 1} \tag{15}$$

$$\sigma = \xi\omega_n = 10 \tag{16}$$

$$\omega_n\sqrt{\xi^2 - 1} = 10.49 \tag{17}$$

$$s_{1,2} = -10 \mp j10.49 \tag{18}$$

Since e_{ss} $(\infty) = 0.02$

$$K_V = \frac{1}{e_{ss}} = \frac{1}{0.02} = 50 \tag{19}$$

$$K_V = \lim_{s\to 0} sG_{C_s}G_{S_s} \tag{20}$$

$$K_V = \lim_{s\to 0} s\left\{K_P + \frac{K_I}{s} + K_D s\right\}\left\{\frac{0.752}{0.157s+1}\right\} \tag{21}$$

Finally, the integral coefficient of the control system is determined as $K_I = 37.6$. The characteristic equation of the system is defined as:

$$F(s) = 1 + G_{C_s}G_{S_s} = 0 \tag{22}$$

Therefore $G_{C_s} = -\frac{1}{G_{S_s}}$ can be obtained. Where G_{C_s} is the controller block and G_{S_s} is the DC motor transfer function block.

$$K_P + \frac{K_I}{s} + K_D s = -\frac{1}{\frac{0.752}{0.157s+1}} \tag{23}$$

$$K_P + K_D(-10 + j10.49) = -\frac{0.157(-10 + j10.49) + 1}{0.752} - \frac{37.6}{-10 + j10.49} \tag{24}$$

Consequently, the proportional and derivative coefficients of the control system is determined as $K_P = 2.84$ and $K_D = 0.029$, respectively. A closed loop DC motor control system is given in Figure 3.

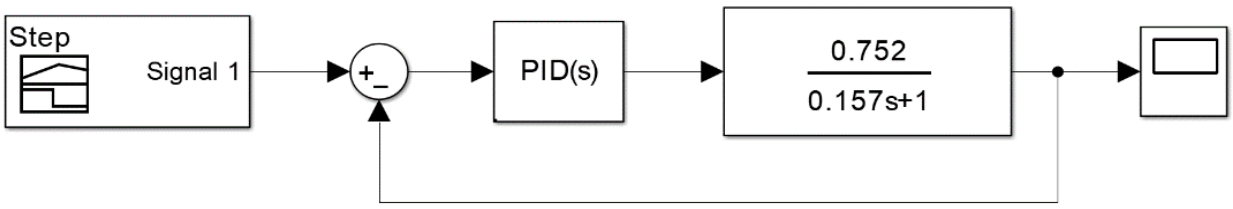

Figure 3. *Closed loop control system of the DC motor*

4. GENETIC ALGOTIRHM BASED PID TUNING SOFTWARE DESIGN

In this section, the software design for the GA-based PID tuning system is described. PID control method is widely used in closed loop systems. However, selection of the proportional, integral, and derivative gains is crucial to controller performance. To simplify the PID tuning process, a GA optimization algorithm is employed. The GA is an optimization method that draws inspiration from the genetics field's natural selection procedure. Crossover, mutation and selection processes are used to iteratively improve a starting population of potential solutions (PID gains). The following steps make up the software design:

Input and initialization: The DC motor transfer function, control targets, and PID limits are provided by the user as input to the software. Maximum number of generations, population size, mutation and crossover rates are also initialized.

Fitness evaluation: Fitness function is the objective metric used to evaluate each proposed solution's effectiveness. In this case, fitness function is integral time absolute error (ITAE) between the desired and actual motor speed over a certain time period.

Selection: The fittest members in the population are selected based on their fitness value. The tournament selection method is used, where pairs of individuals are randomly selected and compared based on their fitness value. The individual with highest fitness value is chosen for a next generation.

Crossover: Genetic information of two selected individuals is combined to generate new offspring. The one-point crossover strategy is applied, in which the genetic information beyond a predetermined point in the PID gain strings of two individuals are exchanged between them.

Mutation: Random changes in the genetic information of the offspring are introduced to diversify the population and avoid premature convergence.

Termination: When the maximum number of generations is reached or the fitness value of the best individual reaches a sufficient level, the GA process terminates.

Output and visualization: The optimal PID parameters for the DC motor control system are output by the software based on the fittest individual in the final generation. The software also provides a plot of the actual output value and the desired reference over time to visualize the performance of the system.

Overall, the software design enables the adjusting of PID parameters for DC motor control using a GA optimization algorithm. The next section describes the implementation of the software on a MATLAB platform.

5. GA BASED PID TUNING ALGORITHM

Initialization, fitness evaluation, selection, crossover, and mutation are the different parts of the algorithm. Initial parameters used in GA based PID software is given in Table 2.

Part 1 of the GA tuned PID tuning algorithm is given in Figure 4. In this part, the number of genes in each chromosome, upper and lower bounds of the parameters, size of each gene, and mutation and crossover rates are determined. A random population is generated within the specified limits for the initial population. The start and end bits of the K_D, K_P, and K_I genes in the chromosome are determined. If the random gene exceeds the upper limit, it is set to the upper limit and if it is below the lower limit, it is set to the lower limit. In addition, if the sign is negative, the sign bit is set to zero, and if it is positive, the sign bit is set to 9, and the absolute value of the genes is taken.

Part 2 of the GA based PID tuning algorithm is given in Figure 5. In this part, transfer function of DC motor is defined and the genes in the chromosome are placed into the transfer function. The integral of the system is taken, and the total error value is transferred to the objective fitness. Fitness function values are calculated from all objective fitness values, and the total fitness value is found and stored for use in the roulette wheel

method. The best individuals are selected using the roulette wheel method for crossover and best chromosomes move to the next generation.

```
gene_number=3;
upper_limit=[0.5 5 50];
lower_limit=[0 0 0];
gene_size=[6 6 6]';
decimal=[2 2 2];
mutation_rate=0.01;
crossover_rate=0.7;
population_size=50;
generation=40;
loop=1;
g=1;
for i = 1:population_size
    for j = 1:gene_number
        gene(j,i,loop)=((rand*(upper_limit(j)-lower_limit(j)))+lower_limit(j));
    end
end
chromosome_size=sum(gene_size)+gene_number;
gene_start(1)=1;
for j=1:gene_number
gene_start(j+1)= gene_start(j)+gene_size(j)+1;
end
while loop <= generation
for i = 1:population_size
    for j = 1:gene_number
        if gene(j,i,loop)>upper_limit(j)
            gene(j,i,loop)=upper_limit(j);
            elseif gene(j,i,loop) < lower_limit(j)
            gene(j,i,loop)=lower_limit(j);
        end
        if gene(j,i,loop) < 0
            pop(gene_start(j),i)=0;
        else
            pop(gene_start(j),i)=9;
        end
        temporary_gene(j,i) = abs(gene(j,i,loop));
        temporary_gene(j,i) = temporary_gene(j,i)/10^(decimal(j)-1);
        for k = gene_start(j)+1:gene_start(j+1)-1,
            pop(k,i) = temporary_gene(j,i)-rem(temporary_gene(j,i),1);
            temporary_gene(j,i) = (temporary_gene(j,i)-pop(k,i))*10;
        end
    end
end
```

Figure 4. *Software design in a MATLAB environment (Part 1)*

Table 2. *Initial parameters of GA based PID software*

Parameters	**Values**
Population size	50
Generation	40
Number of gene	3
Limits of K_P	0-1
Limits of K_I	0-10
Limits of K_D	0-0.5
Size of gene	6, 6, 6
Mutation rate	0.01
Crossover rate	0.2

Part 3 of the GA tuned PID tuning algorithm is shown in Figure 6. In this part, the next step of the software is crossover and child chromosomes are generated from the parent chromosomes. During the process, chromosomes are randomly selected for crossover. If the crossover rate determined at the beginning of the software is greater than the randomly generated number between 0-1, crossover occurs, if it is small, it does not occur and the parent chromosomes are transferred to the child chromosomes exactly.

```
    a=1;
    sumfitness=0;
    for t= 1:population_size
    m=0:0.01:10;
    numerator_1=0.752;
    denumerator_1=[0.157 1];
    sis1=tf(numerator_1,denumerator_1);
    numerator_2=[gene(1,a,loop) gene(2,a,loop) gene(3,a,loop)];
    a=a+1;
    denumerator_2=[1 0];
    sis2=tf(numerator_2,denumerator_2);
    sistop=series(sis1,sis2);
    new_sis=feedback(sistop,1);
    y=step(new_sis,m);
    error=0;
        for i=1:1001
            error=error+abs((y(i)-1)*0.01);
        end
    objective_fitness(t)=error;
    end
    best_obj_function=objective_fitness(1);
    for h=1:population_size
        if objective_fitness(h)<best_obj_function
            best_obj_function=objective_fitness(h);
        end
    end
    value(g)=best_obj_function;
    g=g+1;
    for t= 1:population_size
    fitness(t)=1/objective_fitness(t);
    sumfitness = sumfitness + fitness(t);
    end
    [bestfitness(loop),bestmember]=max(fitness);
    bestindividual(:,loop)=gene(:,bestmember,loop);
    average_fitness(loop) = sumfitness/population_size;
    for i=1:population_size
        pointer=rand*sumfitness;
        members_number=1;
        toplam=fitness(1);
        while toplam < pointer
            members_number=members_number+1;
            toplam=toplam+fitness(members_number);
        end
```

Figure 5. *Software design in a MATLAB environment (Part 2)*

In case of crossover, it is determined randomly from which bit the crossover starts. The bits of the selected chromosome from 1 to the determined place are transferred to the new child chromosome. The bits of another chromosome from the determined place to the end are transferred to the new child chromosome.

Thus, the child chromosome is formed by crossing the two randomly selected chromosomes from the randomly determined dividing point. Then the chromosomes mutate according to the mutation rate. If the mutation rate determined at the beginning is greater than the randomly generated number between 0 and 1, the mutation will occur, if it is small, it will not occur. Finally, new chromosomes are transferred to the gene pool and the cycle is increased by one.

Finally, the PID parameters produced by the software and the numerically calculated parameters in section 3 are applied to the DC motor transfer function and the relevant code lines are given in Figure 7.

```
        parent_chrom(:,i)=pop(:,members_number);
    end
    for s=1:population_size
        q=s;
        while q==s
            q = rand*population_size;
            q = q-rem(q,1)+1;
        end
        if crossover_rate > rand
            bolum = rand*chromosome_size;
            bolum = bolum-rem(bolum,1)+1;
            child(1:bolum,s)=parent_chrom(1:bolum,s);
            child(bolum+1:chromosome_size,s)=parent_chrom(bolum+1:chromosome_size,q);
        else
            child(:,s)=parent_chrom(:,s);
        end
    end
    for s=1:population_size
        for p=1:chromosome_size
            if mutation_rate > rand
            rand_gen=rand*10;
            while child(p,s) == rand_gen-rem(rand_gen,1)
            rand_gen=rand*10;
            end
            child(p,s)=rand_gen-rem(rand_gen,1);
            end
        end
    end
    pop=child;
    loop=loop+1;
    for i=1:population_size
        for j=1:gene_number
            gene(j,i,loop)=0;
            bookmark=1;
            for k=gene_start(j)+1:gene_start(j+1)-1
                place=decimal(j)-bookmark;
                gene(j,i,loop)=gene(j,i,loop)+(pop(k,i))*10^place;
                bookmark=bookmark+1;
            end
        end
    end
    end
```

Figure 6. *Software design in a MATLAB environment (Part 3)*

```
t=0:0.01:10;
pay2=[response(1,1) response(2,1) response(3,1)];
payda2=[1 0];
sis2=tf(pay2,payda2);
sistop=series(sis1,sis2);
new_sis=feedback(sistop,1);
y=step(new_sis,t);
plot(t,y,'r:');
hold on
pay_2=[0.029 2.84 37.6];
payda_2=[1 0];
sis_2=tf(pay_2,payda_2);
sis_top=series(sis1,sis_2);
new_sis_=feedback(sis_top,1);
[y,x]=step(new_sis_,t);
plot(t,y,'g-');
```

Figure 7. *Software design in a MATLAB environment (Part 4)*

6. MANUAL SOLUTION FOR A SINGLE ITERATION

In this section, the process of a single iteration of the system is demonstrated. Consider a system with three chromosomes, and assume that the first genes are generated by MATLAB as follows:

$$K_D = 13.5162; \quad 27.4001; \quad 16.1503$$

$$K_P = 2.5146; \quad 4.5713; \quad 29.8840$$

$$K_I = 6.8693; \quad 24.7745; \quad 2.3453$$

Then the calculation of the start and end points of the genes will subsequently be performed by the software.

$$gene\_start = 1 \quad 8 \quad 15 \quad 22$$

After the start and end points are determined, a gene pool equivalent to the length of the system is created.

$$chromosome\ 1 = 9135162; \quad 9025146; \quad 9068693$$

$$chromosome\ 2 = 9274001; \quad 9045713; \quad 9247745$$

$$chromosome\ 3 = 9161502; \quad 9298840; \quad 9023452$$

As evident from the gene pool, a value of 9 has been assigned to the starting bits, which are positive. If the signs were negative, the sign bits would have been 0. Subsequently, the objective function and fitness function are calculated for these chromosomes.

$$objective\ function = 2.2370; \quad 1.5646; \quad 0.5798$$

$$fitness = 0.4470; \quad 0.6392; \quad 1.7246$$

After calculating the fitness values, the well-performing genes are selected using the roulette wheel selection method, and a crossover pool is generated. This pool is stored in a matrix called parent_chrom.

$$chromosome\ 3 = 9161502; \quad 9298840; \quad 9023452$$

$$chromosome\ 1 = 9135162; \quad 9025146; \quad 9068693$$

$$chromosome\ 3 = 9161502; \quad 9298840; \quad 9023452$$

As can be seen, the well-performing genes are placed into the gene pool named parent_chrom. Subsequently, the crossover process takes place, and chromosomes are randomly selected and transferred to the child matrix from randomly chosen positions. This results in a gene exchange among the chromosomes.

After this process, the mutation stage is initiated, and it is decided whether each gene will undergo a mutation at a rate of 0.01. If a gene is to be mutated, a number between 0 and 10 is randomly generated and substituted for the corresponding gene. The resulting gene pool after mutation is as follows:

$$chromosome\ 3 = 9161502; \quad 9298840; \quad 9023452$$

$$chromosome\ 1\&3 = 9135162; \quad 90//98840; \quad 9023452$$

$$chromosome\ 3 = 9161502; \quad 9298840; \quad 9023452$$

Subsequently, the chromosomes named child undergo a mutation with a probability of 0.01. The resulting chromosomes after mutation are as follows:

$$new\ chromosome\ 1 = 9161502;\quad 9298840;\quad 9023452$$

$$new\ chromosome\ 2 = 9135162;\quad 9098840;\quad 9023452$$

$$new\ chromosome\ 3 = 9161502;\quad 9298840;\quad 9023452$$

If it is examined the new chromosomes, none of the bits have undergone a mutation with a probability of 0.01. However, this does not necessarily mean that there will never be any mutations. There were no mutations in the first generation.

After the mutation process, the chromosomes in this gene pool are converted to parameters and the second generation begins. After this step, the parameter values will be as in Table 3.

Table 3. *PID parameters obtained from the first generation*

	Chromosome 1	**Chromosome 2**	**Chromosome 3**
K_D	16.1502	13.5162	16.1502
K_P	29.8840	9.8840	29.8840
K_I	2.3452	2.3452	2.3452

7. RESULTS

The step responses of conventional PID and GA tuned PID are given in Figure 8. As calculated in the third section, parameters $K_D = 0.029$, $K_P = 2.84$ and $K_I = 37.6$ are selected for the conventional PID controller. Besides, when the GA-based PID tuning algorithm given in section 5 is run, the obtained PID parameters are $K_D = 0.012$, $K_P = 5$ and $K_I = 36.4872$, respectively. As can be seen from the figure, the parameters obtained from the GA-based PID algorithm are superior to the conventional PID calculation method in terms of overshoot and settling time.

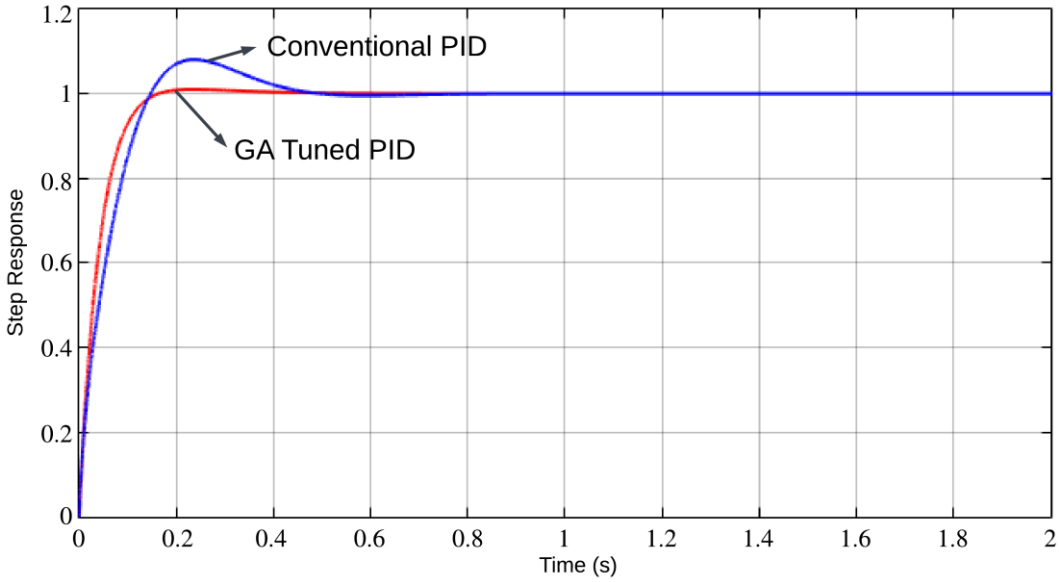

Figure 8. *Step responses of conventional PID and GA-based PID control methods*

The multi-step responses of conventional PID and GA-based PID is given in Figure 9. The same parameters as above are used for both conventional PID and GA-based PID. As can be seen from the figure, the parameters obtained from the GA-based PID algorithm are superior to the conventional PID calculation method in terms of overshoot and settling time for all unit steps.

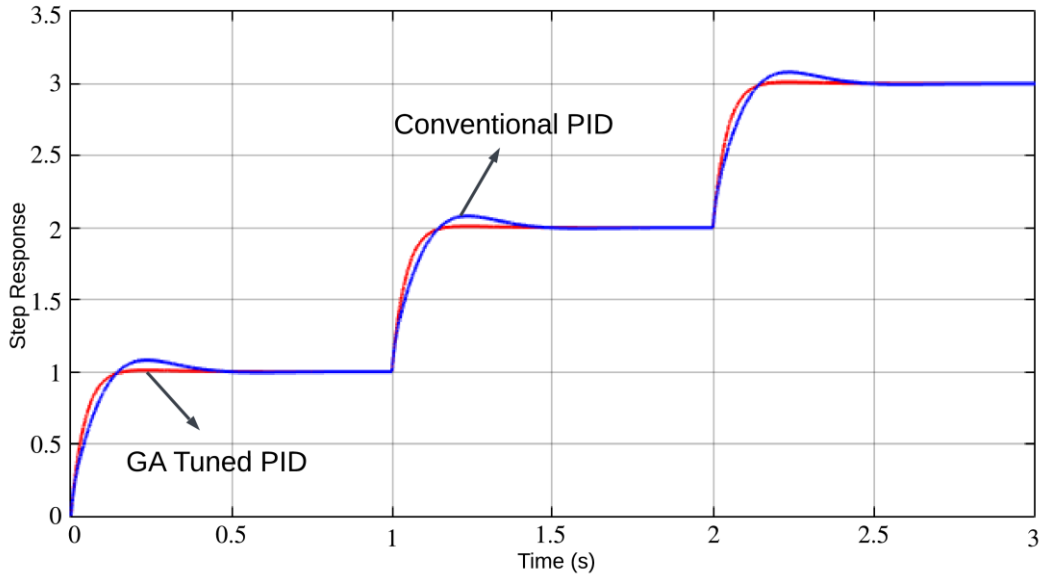

Figure 9. *Multi-step responses of conventional PID and GA-based PID control methods*

The step responses of conventional PID and two GA-based PID are given in Figure 10. Derivative, proportional and integral terms ($K_D = 0.029$, $K_P = 2.84$, $K_I = 37.6$) are selected for the conventional PID controller. Moreover, $K_D = 0.012$, $K_P = 5$ and $K_I = 36.4872$ are obtained GA-based PID tuning algorithm when the population size is 50 and iteration is 40 (response 1) and $K_D = 0$, $K_P = 5$ and $K_I = 34.1696$ are obtained when the population size is 100 and iteration is 80 (response 2), respectively. As can be seen from the figure, the algorithm gives better results in terms of overshoot and settling time as the number of populations and iterations increase. Moreover, all the parameters obtained from the GA-based PID algorithm are superior to the conventional method.

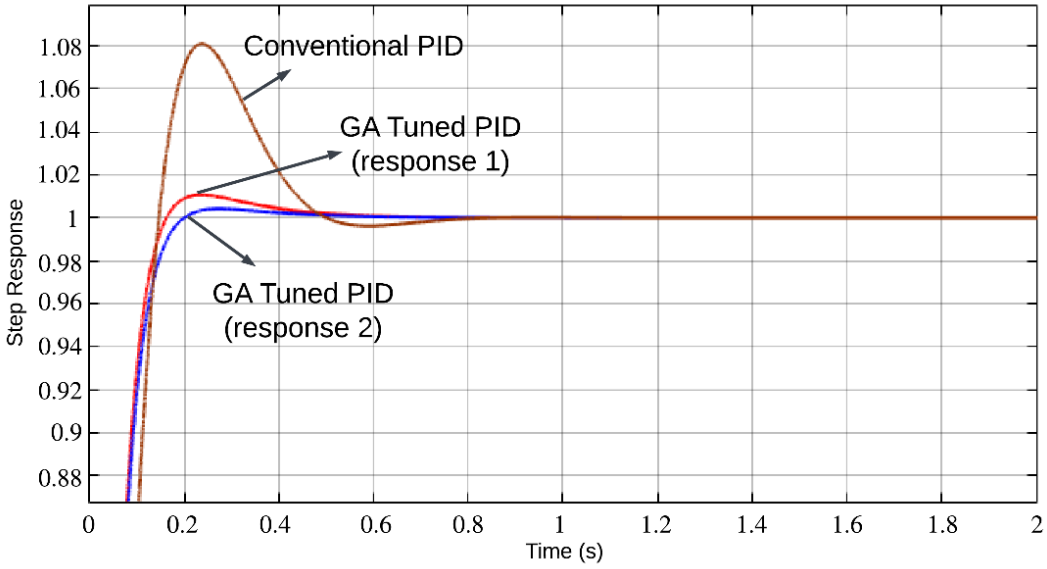

Figure 10. *Step response of conventional and two GA-based PID control methods*

The step response of conventional PID and GA tuned PID to the limitation of derivative, proportional and integral terms is given in Figure 11. In this simulation, all parameters are limited in the range of 0-50. Therefore, $K_D = 14.0465$, $K_P = 45.9608$ and $K_I = 50$ are obtained GA-based PID tuning algorithm when the population size is 50 and iteration is 40. Whereas, in all previous simulations, this limitation was set to $0 < K_D < 5$, $0 < K_P < 5$ and $0 < K_I < 50$. When the limits of the derivative and proportional terms are increased, the system responds quickly, but the settling time to the reference is too long. Therefore, the

boundaries of the terms specified in the algorithm should be evaluated separately for each system and appropriate limitations should be selected.

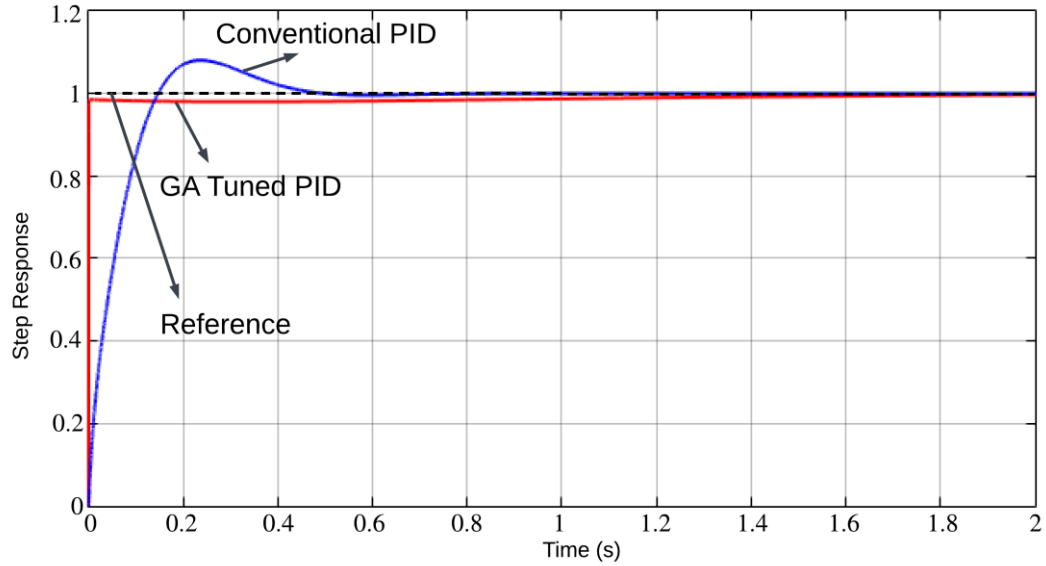

Figure 11. *Step response of conventional and GA tuned PID based DC motor*

Finally, the ramp signal responses of conventional PID and GA-based PID are given in Figure 12. Derivative, proportional and integral terms ($K_D = 0.029$, $K_P = 2.84$, $K_I = 37.6$) are selected for the conventional PID controller. Moreover, $K_D = 0.025$, $K_P = 3.95$ and $K_I = 43.832$ are obtained GA-based PID tuning algorithm when the population size is 50 and iteration is 40. As can be seen from the figure, the algorithm gives better results than conventional method in terms of overshoot and steady state error.

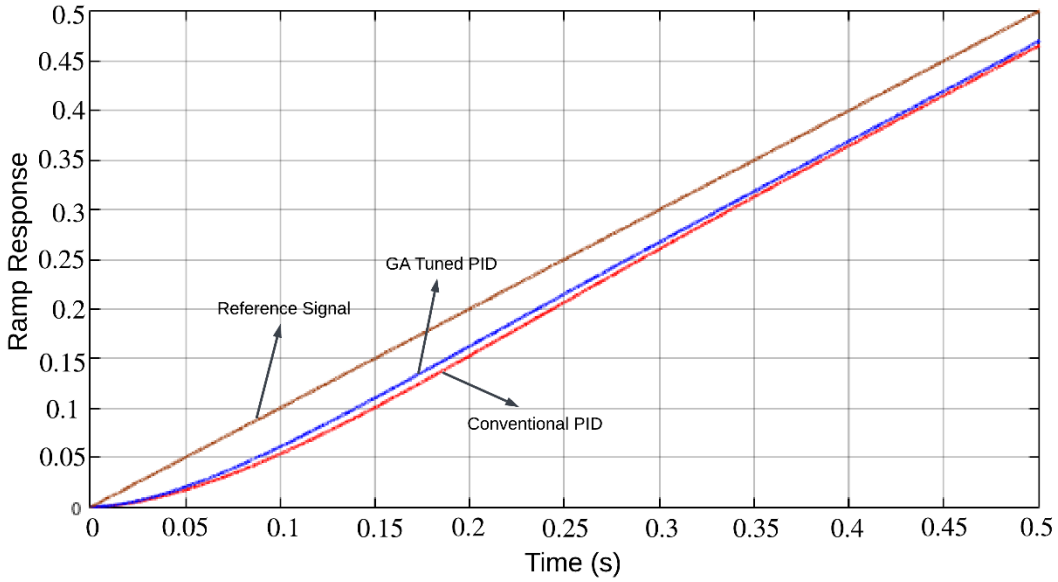

Figure 12. *Ramp signal response of conventional and GA tuned PID methods*

8. CONCLUSION

In this paper, GA based PID tuning software design and implementation for a DC motor control system is presented. To optimize the performance of DC motor control system, the proposed software architecture automates the process of adjusting proportional, integral, and derivative gains.

The output of the optimal proportional-integral-derivative (PID) parameters is examined following the conclusion of 40 iteration runs in a 50 population system. The overshoot is shortened, the steady-state error is minimized, and the settling time is greatly decreased, according to the GA-enhanced system. A comprehensive analysis of these results confirms that the PID-developed systems implemented in the industry can yield highly favorable outcomes. It is worth noting that increasing the generation and population size within the system could lead to even greater optimization of the outcomes. However, it is also necessary to take into account the fact that such increases will inevitably lead to a significant increase in processing time, potentially prolonging solution times.

REFERENCES

Alruim Alhasan, H., & Güneş, M. (2017). A New Adaptive Particle Swarm Optimization Based on Self-Tuning of PID Controller for DC Motor System. Çukur*ova University Journal of the Faculty of Engineering and Architecture, 32*(3), 243-249.

Aranza, M. F., Kustija, J., Trisno, B., & Hakim, D. L. (2016). Tunning PID controller using particle swarm optimization algorithm on automatic voltage regulator system. *IOP Conference Series: Materials Science and Engineering, 128*, 012038. doi:10.1088/1757-899X/128/1/012038

Borase, R. P., Maghade, D. K., Sondkar, S. Y., & Pawar, S. N. (2021). A review of PID control, tuning methods and applications. *International Journal of Dynamics and Control*, *9*(2), 818-827. doi:10.1007/s40435-020-00665-4

Fang, H., Zhou, J., Wang, Z., Qiu, Z., Sun, Y., Lin, Y., Chen, K., Zhou, X., & Pan, M. (2022). Hybrid method integrating machine learning and particle swarm optimization for smart chemical process operations. *Frontiers of Chemical Science and Engineering*, *16*(2), 274-287. doi:10.1007/s11705-021-2043-0

de Figueiredo, R., Toso, B., & Schmith, J. (2023). Auto-Tuning PID Controller Based on Genetic Algorithm. In: M. Shamsuzzoha & G. L. Raja (Eds.), *Disturbance Rejection Control*. IntechOpen. doi:10.5772/INTECHOPEN.110143

Flores-Morán, E., Yánez-Pazmiño, W., Espín-Pazmiño, L., Carrera-Manosalvas, I., & Barzola-Monteses, J. (2020, October 13-16). *Particle Swarm Optimization and Genetic Algorithm PID for DC motor position controllers*. In: Proceedings of the 2020 IEEE ANDESCON, Quito, Ecuador. doi:10.1109/ANDESCON50619.2020.9272127

Galotto, L., Pinto, J. O. P., Bottura Filho, J. A., & Lambert-Torres, G. (2007, November 5-8). *Recursive least square and genetic algorithm based tool for PID controllers tuning*. In: Proceedings of the 2007 International Conference on Intelligent Systems Applications to Power Systems (ISAP), Kaohsiung, Taiwan. doi:10.1109/ISAP.2007.4441623

Ibrahim, O., Yahaya, N. Z. B., & Saad, N. (2016). PID Controller Response to Set-Point Change in DC-DC Converter Control. *International Journal of Power Electronics and Drive Systems (IJPEDS)*, *7*(2), 294-302. doi:10.11591/IJPEDS.V7.I2.PP294-302

Islam, Md. T., Karim, S. M. R., Sutradhar, A., & Miah, S. (2020). Fuzzy Logic and PID Controllers for DC Motor Using Genetic Algorithm. *International Journal of Control Science and Engineering*, *10*(2), 37-41. doi:10.5923/J.CONTROL.20201002.03

Jayachitra, A., & Vinodha, R. (2014). Genetic Algorithm Based PID Controller Tuning Approach for Continuous Stirred Tank Reactor. *Advances in Artificial Intelligence, 2014*, 791230. doi:10.1155/2014/791230

Korkmaz, M., Aydoğdu, Ö., & Doğan, H. (2012, July 2-4). *Design and performance comparison of variable parameter nonlinear PID controller and genetic algorithm based PID controller*. In: Proceedings of the 2012

International Symposium on Innovations in Intelligent Systems and Applications (INISTA), Trabzon, Türkiye. doi:10.1109/INISTA.2012.6246935

Taşören, A. E. (2021). Design and Realization of Online Auto Tuning PID Controller Based on Cohen-Coon Method. European *Journal of Science and Technology*, 24 (Special Issue), 235-239. doi:10.31590/ejosat.897727

Malhotra, R., Singh, N., & Singh, Y. (2011). Genetic Algorithms: Concepts, Design for Optimization of Process Controllers. *Computer and Information Science*, *4*(2), 39-54. doi:10.5539/CIS.V4N2P39

Martinez-Soltero, E. G., & Hernandez-Barragan, J. (2018). Robot Navigation Based on Differential Evolution. *IFAC-PapersOnLine*, *51*(13), 350-354. doi:10.1016/J.IFACOL.2018.07.303

Meena, D. C., & Devanshu, A. (2017, January 19-20). *Genetic algorithm tuned PID controller for process control*. In: Proceedings of the 2017 International Conference on Inventive Systems and Control (ICISC), Coimbatore, India. doi:10.1109/ICISC.2017.8068639

Patel, V. V. (2020). Ziegler-Nichols Tuning Method: Understanding the PID Controller. *Resonance*, *25*(10), 1385-1397. doi:10.1007/s12045-020-1058-z

Pereira, D. S., & Pinto, J. O. P. (2005, July 24-28). *Genetic Algorithm based system identification and PID tuning for optimum adaptive control*. In: Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Monterey, CA, USA, (pp. 801-806). doi:10.1109/AIM.2005.1511081

Rout, U. K., Sahu, R. K., & Panda, S. (2013). Design and analysis of differential evolution algorithm based automatic generation control for interconnected power system. *Ain Shams Engineering Journal*, *4*(3), 409-421. doi:10.1016/J.ASEJ.2012.10.010

Saad, M. S., Jamaluddin, H., & Darus, I. Z. M. (2012). Implementation of PID controller tuning using differential evolution and genetic algorithms. *International Journal of Innovative Computing Information and Control*, 8(11), 7761-7779.

Tiwari, S., Bhatt, A., Unni, A. C., Singh, J. G., & Ongsakul, W. (2018, October 24-26). *Control of DC Motor Using Genetic Algorithm Based PID Controller*. In: Proceedings of the 2018 International Conference and Utility Exhibition on Green Energy for Sustainable Development (ICUE), Phuket, Thailand. doi:10.23919/ICUE-GESD.2018.8635662

Wati, D. A. R., & Hidayat, R. (2013, November 25-27). *Genetic algorithm-based PID parameters optimization for air heater temperature control*. In: Proceedings of the 2013 International Conference on Robotics, Biomimetics, Intelligent Computational Systems (ROBIONETICS), Jogjakarta, Indonesia, (pp. 30-34). doi:10.1109/ROBIONETICS.2013.6743573