Genetic Algorithm: An approach to Velocity Control of an Electric DC Motor

M. B. Anandaraju  
HOD, Deptt of ECE, BGSIT,  
B.G.Nagara, Mandya, Dist.

Dr. P.S. Puttaswamy  
Prof, Deptt of E&E,  
PESCE, Mandya.

Jaswant Singh Rajpurohit  
Technical Director,  
Planet-I Technologies, Bangalore.

ABSTRACT
DC motor is a vital component in most of the process control industries. PID controllers are extensively used in DC motors for speed as well as position control. Tuning of PID controller parameters is an iterative process and needs complete optimization to achieve the desired performance. Genetic algorithm (GA) which is a well established tool for optimization has been used to extract PID controller parameters for the velocity control of the DC motor. Different error models are used for evaluating the fitness function. Velocity control is demonstrated using MATLAB/SIMULINK modeling.

Keywords
Component; DC motor, PID, Genetic Algorithm, Fitness Function

1. INTRODUCTION
Mechanical applications based on electric motors demand high speed accuracies [1]. Drive design for such systems is a complicated and involved process given the number of parameter optimizations involved. Many modern control methodologies like adaptive control [2], nonlinear control [3], optimal control [4] have been studied and attempted to deal with different types of motor controls. These approaches are mathematically complex and complicated to implement. Motor control based on proportional, integral and derivative (PID) controllers has been largely studied and implemented with success.

The present work deals with controlling the time domain performance parameters of a DC motor. The work based on developing a mathematical model of the DC motor which is then integrated with PID controller using SIMULINK. Determination of the controller gains that maximizes or minimizes a performance specification is a complex, constrained optimization problem.

Genetic Algorithms (GAs) [5] present a way to solve such complex optimization problems. GAs has been used extensively in adaptive control of continuous and discrete-time [6], decentralized PI controls [7] as well as optimal control problems [8]. In the present work GA is used as a computing tool to solve the optimization problem and derive the controller parameters.

2. MODELING A DC MOTOR
2.1 Modeling Scheme
Electric DC motor converts electrical energy to mechanical energy using the electromagnetic induction principle [9]. To achieve velocity control of a DC motor, the primary requirement is to model the DC motor system itself [10]. In control systems terminology it is referred to as plant modeling. The major steps in developing a DC motor model are as follows:
1. Represent the schematic as in Fig. 1
2. Formulate the system equations
3. Generate the transfer function.
4. Develop the simulink model

![Fig 1: Armature Schematic of a DC Motor](image)

By applying a voltage $V_a$ to the coil, it causes a current $I_a$ to flow which produces a proportional torque. This in turn accelerates the armature. As the armature gathers speed, a proportional electromagnetic field is generated which tends to oppose the current. Eventually, the rotor gathers sufficient speed such that the back electromagnetic field just equals the applied voltage. In the steady state, when $V_a = e_b$, no current flows in the coil thus there is no further acceleration and the rotor turns at a constant speed.

This condition is for a perfect motor with no friction at all in the bearings and no electrical power losses. In practice, running a motor at steady speed requires a small amount of current to flow – sufficient to just overcome the friction at that speed. The rotor coil will also have some resistance, so any current will cause ohmic losses and a small voltage drop. The coil also has self-
inductance; if the current in the coil changes then there will be an induced back emf which opposes that change.

2.2 System Equations

Fig. 1 shows the schematic representation of a DC motor $R_a, L_a$ and $e_b$ represents the armature resistance, inductance and the back emf, respectively. The motor torque $\tau$ force required to rotate the object about its axis is related to the armature current, $i$ by torque constant $K$ represented as:

$$\tau = Ki$$

The back emf $e_b$ is voltage or electromagnetic force that pushes against the current which is induced by armature is related to angular velocity as represented as:

$$e_b = K\omega = K\frac{d\theta}{dt}$$

In mechanical domain, if $J$ is the rotational moment of inertia of motor, $\omega$ is the motor angular velocity, $(\tau_m - \tau_l)$ is the net torque where $\tau_m$ the electromagnetic torque and $\tau_l$ load torque, the torque equation in terms of moment of inertia we can write as:

$$J\frac{d\omega}{dt} = (\tau_m - \tau_l) = Ki$$

Combining Newton’s law and Kirchhoff’s law we can write:

$$J\frac{d^2\theta}{dt^2} = Ki$$

$$L_a\frac{di}{dt} + Ra = Va - K\frac{d\theta}{dt}$$

From the above equations, we can derive the s-domain transfer function of the motor which can be used for complex frequency functions:

$$Js\omega(s) = KI(s)$$

$$LsI(s) + RI(s) = V(s) - K\omega(s)$$

To find the transfer function we required $\frac{\omega(s)}{V(s)}$ so

$$\omega(s) = \frac{K}{Js^2(1 + sLa/Ra)} + \frac{1/Ra}{1 + K^2/J^2 s^2(1 + sLa/Ra)}$$

3 GENETIC ALGORITHM

Genetic algorithm (GA) is a heuristic mimicking the natural evolution process and is routinely used to generate useful solutions to optimization problems. In this work the genetic algorithm is used to derive the PID controller parameters by optimizing the error in the DC motor angular velocity.

In GA [5, 12], a population of strings called chromosomes encodes the possible solutions of an optimization problem and evolves for a better solution by process of reproduction. The process of evolution starts from a population of randomly generated individuals. Optimization is achieved in generations where in each generation, the fitness function evaluates each individual in the population and multiple individuals are selected stochastically based on their fitness. These selected individuals are modified to form a new population. The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. The various steps in GA based optimization are detailed below:

3.1. Initialization

From the initial population few individual solutions are generated. The population is generated randomly, covering the entire range of possible solutions.

3.2. Selection

In each generation, individual solutions are selected by evaluating the fitness function and the fitter solutions have a higher probability of selection.
3.3. Reproduction
Next set of population for the successive generation is generated by a process called reproduction and involves crossover (recombination) and mutation. This results in a new set of population derived from the fitter solutions of the previous population. Generally the average fitness of the population is heightened as compared to the population of the previous population.

3.4. Termination
The process of optimization is halted once a termination condition is achieved. The termination condition can be either the number of generations or the solution satisfying an optimum criterion.

4 MATLAB MODELING

4.1 Motor model
Motor model is developed using SIMULINK/MATLAB Fig 2. As the major objective of this work is to achieve velocity control, the GA optimization has been attempted for the PID controller for the velocity loop. The parameters for PID controller are the individuals in the GA population. For each individual, the SIMULINK model is run and the error is computed. This error is then used for evaluation of the fitness function. Different error models are used for the evaluation purpose. The feedback loop consists of a current sensor in the current loop (inner loop) and the tacho-generator in the velocity loop (outer loop). Generally the reference voltage to the motor $V_a$ is fed through a PWM controller. The PWM controller consists of a set of comparators that compare a reference signal with the tacho-feedback signal and generate PWM pulses for an H-bridge converter [11]. The H-bridge converter has been modeled as a single pole function as shown in Fig 4.

![Fig 3: Armature Schematic of a DC Motor](image)

Motor parameters used in the model are listed in Table 1.

### Table 1: DC Motor parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Armature Resistance ($R_a$)</td>
<td>0.0099Ω</td>
</tr>
<tr>
<td>Armature Inductance ($L_a$)</td>
<td>0.73mH</td>
</tr>
<tr>
<td>Motor Inertia ($J$)</td>
<td>1.94 kg-m$^2$</td>
</tr>
<tr>
<td>$K$</td>
<td>1.23 V-rad/s</td>
</tr>
<tr>
<td>Tacho DC gain</td>
<td>0.0273 V-rad/s</td>
</tr>
<tr>
<td>PWM frequency</td>
<td>1kHz</td>
</tr>
</tbody>
</table>

4.2 PID Controller
PID controller transfer function used in the model is shown below:

$$\frac{K_D s^2 + K_P s + K_I}{s(s + \omega_d)}$$

PID controller [13] takes the error input and computes the proportional, derivative and integral of the error. The derivative component improves the system response, whereas the integral and proportional components improve the steady state performance.

4.3 GA Optimization parameters
Genetic algorithm parameters used in the simulation are listed in Table 2.

### Table 2: GA Optimization parameters

<table>
<thead>
<tr>
<th>Initial Population Range</th>
<th>[0 10]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_p$</td>
<td>[0 10]</td>
</tr>
<tr>
<td>$K_d$</td>
<td>[0 100]</td>
</tr>
<tr>
<td>$\omega_d$</td>
<td>[1 50]</td>
</tr>
<tr>
<td>Maximum Generations</td>
<td>150</td>
</tr>
<tr>
<td>Mutation Function</td>
<td>Uniform</td>
</tr>
<tr>
<td>Fitness function</td>
<td>Minimize different errors</td>
</tr>
</tbody>
</table>

Fitness function has a return value which is computed based on a user defined parameter. In the present work, the fitness function has been configured to compute different types of
errors, ITSE (Integral Time Squared Error), ITAE (Integral Time Absolute Error), Peak Overshoot, Absolute Error and Squared Error.

The returned values from the fitness function are then used by genetic algorithm to evaluate fitness of the individual solutions in the population. The individuals which are found fit are selected to generate the population for the next generation.

The definitions of different errors are given below:

\[
ITSE = \int_0^T |e|^2 t \, dt
\]

\[
ITAE = \int_0^T |e| t \, dt
\]

**Peak Overshoot** = \text{max(output)}

Absolute Error is the error which will appear while taking measurement which is given as:

**Absolute Error** = \text{max(output)} – \text{command}

Where \text{max (output)} is the maximum output and command is measured output.

Squared Error = (\text{max (output)} – \text{command})^2

### 4.4 Simulation Model

The complete simulation model is built in SIMULINK and shown in Fig.4.

Table 3 lists the transfer functions of the different blocks in the complete system.

**Table 3: Transfer function of different blocks**

<table>
<thead>
<tr>
<th>Motor</th>
<th>( \omega(s) ) = ( \frac{G(s)}{1 + G(s)H(s)} ) where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Sensor</td>
<td>( \frac{1}{R_a} )</td>
</tr>
<tr>
<td>Tacho Generator</td>
<td>( \frac{1}{1 + 5 \times 10^{-3} s} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Motor</th>
<th>( G(s) = \frac{K}{js} \left( 1 + \frac{1}{R_a} \right) ; H(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Sensor</td>
<td>( \frac{5/823}{1 + 0.8 \times 10^{-3} s} )</td>
</tr>
<tr>
<td>Tacho Generator</td>
<td>( \frac{5/183.26}{1 + 25 \times 10^{-3} s} )</td>
</tr>
</tbody>
</table>

![Fig 4: MATLAB/SIMULINK model of complete system](image)
RESULTS

Fitness function in the GA function is configured to optimize different types of errors. These errors are the return values from fitness function. The Genetic Algorithm uses this return value as the fitness parameter to evaluate the individuals in the population and hence select the best candidates.

Different optimization functions and the resulting PID controller transfer function is listed below. Result for Motor Speed optimization for Absolute Error can be seen in Fig 8. The results for PID and GA parameters for different error models are complied in Table 4. The resulting time domain response is shown in Fig 5-9.

5.1 ITSE (Integral Time Square Error)

\[
\text{ITSE} = \frac{240/5}{(1 + 0.5 \times 10^{-8}s)}
\]

5.2 ITAE (Integral Time Absolute Error)

\[
\text{ITAE} = \frac{6.222 \times 10^{-3}s + 0.61}{(73.74 \times 10^{-3}s + 1)}
\]

5.3 Peak Overshoot

\[
\text{K}_d = \frac{240/5}{(1 + 0.5 \times 10^{-8}s)}
\]

5.4 Absolute Error

\[
f = \text{abs}((\text{max}(\text{velocity}) - 183.26/5*\text{command}))
\]

5.5 Squared Error

for i=1: length (simout)

\[\text{temp} = (183.26/5)*\text{command};\]
\[f = f + \text{abs}((\text{temp} - \text{velocity}(i)) \times \text{tout} (i));\]
\[\text{end}\]

\[
\text{PID} = \frac{3.376s^2 + 3.27s + 7.242}{s(s + 1.561)}
\]

5.6 Abs Error

\[
f = \text{max} (\text{velocity});
\]

Table 4: Optimized PID parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>(K_p)</th>
<th>(K_i)</th>
<th>(K_d)</th>
<th>(\omega_d)</th>
<th>Generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITSE</td>
<td>2.27</td>
<td>7.242</td>
<td>8.376</td>
<td>1.561</td>
<td>62</td>
</tr>
<tr>
<td>ITAE</td>
<td>9.917</td>
<td>7.689</td>
<td>3.804</td>
<td>2.019</td>
<td>67</td>
</tr>
<tr>
<td>Peak Overshoot</td>
<td>9.039</td>
<td>3.734</td>
<td>8.223</td>
<td>3.086</td>
<td>51</td>
</tr>
<tr>
<td>Abs Error</td>
<td>8.883</td>
<td>1.642</td>
<td>7.605</td>
<td>3.117</td>
<td>24</td>
</tr>
<tr>
<td>Squared Error</td>
<td>9.87</td>
<td>2.26</td>
<td>9.643</td>
<td>2.032</td>
<td>35</td>
</tr>
</tbody>
</table>
6. CONCLUSION & FUTURE WORK
Parallel form PID which is also known as non-interacting form and GA parameters are obtained for different error models. It is observed that the optimization for absolute error results in the minimum deviation of the output from the commanded input velocity. However the time taken to reach the desired value is higher. Optimization for ITAE on the other hand gives a fast response as well as minimum steady state error but at the cost of increased overshoot. Other optimizations give nearly similar results. Hence, it is inferred that optimization for ITAE gives the best system performance if the overshoot can be accommodated.

The optimization can be further improved by incorporating a weighted mean of different performance parameters as the return value of fitness function. GA will then try to minimize this parameter and evaluate the solutions.
7. ACKNOWLEDGMENTS
The authors would like to express their cordial thanks to Mr. Ashutosh Kumar and Mr. Kashyap Dhruve of Planet-i Technologies for their much valued support and advice.

8. REFERENCES