Motor Modeling and Position Control Lab  
Week 3: Closed Loop Control 

1. Review 

In the first week of motor modeling lab, a mathematical model of a DC motor from first principles was 
derived to obtain a first order system. The open and closed loop (proportional-derivative) control was imple-
mented specifically for this motor model. In the second week, a physical DC motor (Quanser SRV-02) was 
used for open-loop control implementation and the first order transient characteristics were observed. Based 
on the model response, DC motor parameters (time constant) were estimated both by hand-calculations as 
well as using MATLAB. You should have also observed in the open loop control of actual DC motor that 
the motor positions start to drift over time indicating continuous accumulation of error within the system. 
Another observation that should have been made is that there is no way to enforce the output of the motor 
to track the input voltage in the absence of any feedback loop. 

In the final week of this lab, you will try to address some of these issues by realizing the benefits of 
closed-loop control of DC motor. In particular, you will: 

1. study transient characteristics of a typical second order system and evaluate model or system responses 
   using these specifications. 
2. extend the closed loop control implemented in the first week of this lab to the actual DC motor 
3. analyze the effects of proportional-, derivative- and integral- control individually and in combination 
   on the closed loop response of motor 
4. solve a position control problem by calculating PD controller gains analytically and validate the con-
   trol by monitoring the motor response for different desired trajectories 
5. design a PID controller for the actual DC motor using Ziegler-Nichols’ method and compare the 
   performance with that of the PD controller 

2. DC Motor Model 

We derived the mathematical model of DC motor earlier and obtained the following first order transfer 
function that relates the motor velocity (rad/s) to input voltage (V) as:

\[
\frac{\Omega_l(s)}{V_m(s)} = \frac{K}{\tau s + 1}. \quad (1)
\]

where \(\tau\) is the mechanical time constant of the system, and \(K\) is the steady state gain(also known as DC 

gain). 

Since, angular position can be obtained by integration of angular velocity, the open loop transfer function 
between angular position (rad) and input voltage (V) can be obtained from (1) as in (2):

\[
\frac{\Theta_l(s)}{V_m(s)} = \frac{K}{s(\tau s + 1)} = \frac{K}{\tau s^2 + s} = \frac{a}{s^2 + bs}. \quad \therefore \Theta_l(s) = \frac{1}{s}\Omega_l(s) \quad (2)
\]
The open loop control of the DC servo motor is given in Fig.1. You should remember that simulation of these two forms of mathematical motor models were performed in the first week of motor lab as well as the experiments with actual DC motor in the last week to estimate model parameters. Next, let us study about transient response characteristics of typical 2nd order systems.

![Figure 1: Open Loop Control of DC Servo Motor](image)

### 3. Transient Characteristics of Second Order Systems

Systems that store energy cannot respond instantaneously, so they exhibit a transient response when they are subjected to inputs or disturbances. Consequently, the transient response characteristics constitute one of the most important factors in system design. In many practical cases, the desired performance characteristics of control systems can be given in terms of transient response specifications and step inputs are commonly used input for this purpose, since such an input is easy to generate and is sufficiently drastic. Mathematically, if the response of a linear system to a step input is known, by principle of superposition and linear theory assumptions, it is possible to compute the system’s response to any input. However, the transient response of a system to a unit step input depends on initial conditions. For convenience in comparing the transient responses of various systems, it is thus a common practice to use standard initial conditions: the system is initially considered to be at rest, with its output and all time derivatives thereof zero to facilitate comparison of different transient responses (if you had not realized this earlier). By ensuring these protocols, it will then be possible to compare transient responses for different controller parameters and design our controllers for given transient response specifications.

In general, transfer function of a 2nd order system with input, \( u(t) \) and output, \( y(t) \) can be expressed as in (3),

\[
G(s) = \frac{Y(s)}{U(s)} = \frac{k \omega_n^2}{s^2 + 2\zeta \omega_n s + \omega_n^2} \tag{3}
\]

\( U(s) \) and \( Y(s) \) are Laplace transforms of \( u(t) \) and \( y(t) \) respectively.

The response for such a system is given in Fig.2 marked with the transient response specifications that are defined below and any or all of the following parameters can be used to specify such responses completely:

1. **Rise time \( t_r \):** Time required for a signal to change from a specified low value to a specified high value. Typically, these values are 10% and 90% of the input step size.

2. **Maximum Overshoot \( M_p \):** It is measured as a ratio of value from the maximum peak of the response curve measured to the desired response of the system and final steady state value. Typically, it is

---

\(^1\)Note: When designing or analyzing a system, often it is useful to model the system graphically. Block Diagrams are a useful and simple method for analyzing a system graphically. You will see a lot more of these in this lab manual.
measured for step inputs in which case, the percentage overshoot (PO) is the maximum value minus the input step size divided by the step size % of ratio of value.

3. Settling time ($t_s$): Time elapsed from the application of an ideal instantaneous step input to the time at which the output has entered and remained within a specified error band (typically within 2% or 5% within the final value).

4. Delay time ($t_d$): Time required for the response to reach half the final value the very first time.

5. Peak time ($t_p$): Time required for the response to reach the first peak of the overshoot.

6. Steady-state error ($e_{ss}$): Difference between the desired final output ($y_{des}$) and the actual response when the system reaches a steady state ($y^{ss}$), when its behavior may be expected to continue if the system is undisturbed.

![Figure 2: Typical 2nd Order Motor Response and Transient Characteristics](image)

Given a 2nd order system response, these parameters can be manually estimated and responses for different inputs can be compared. At the same time, by following rigorous derivations, these parameters can be expressed as a function of $\omega_n$ and $\zeta$ and without delving into full details, the final results are summarized in the table 1. We will discuss how you can use MATLAB to trivially calculate these values later.

So once we give the values of $t_r$, $t_s$, $t_d$, $t_p$ and $M_p$, then the transient response from Fig.2 can be completely specified. Nevertheless, in most real applications, desired values of these parameters would be given and the objective will be to design controllers that can meet the requirements. Some desirable characteristics
### Table 1: Transient System Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol (unit)</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise Time</td>
<td>$t_r(s)$</td>
<td>$\frac{\pi - \beta}{\omega_d}$, where $\beta = \tan^{-1}\left(\sqrt{1 - \zeta^2}\right)$, $\omega_d = \omega \sqrt{1 - \zeta^2}$</td>
</tr>
<tr>
<td>Maximum Overshoot</td>
<td>$M_p(%)$</td>
<td>$-\frac{\zeta \pi}{\sqrt{1 - \zeta^2}} \times 100$</td>
</tr>
<tr>
<td>Delay Time</td>
<td>$t_d(s)$</td>
<td>$\frac{1 + 0.7\zeta}{\omega_n}$</td>
</tr>
<tr>
<td>Settling Time</td>
<td>$t_s(s)$</td>
<td>$\frac{4}{\zeta \omega_n}$ (2 % setting time) $\frac{3}{\zeta \omega_n}$ (5 % settling time)</td>
</tr>
<tr>
<td>Peak Time</td>
<td>$t_p(s)$</td>
<td>$\frac{\pi}{\omega_d}$</td>
</tr>
<tr>
<td>Steady State Error</td>
<td>$e_{ss}$</td>
<td>$\lim_{t \to \infty} y(t) - y_{des}^s$</td>
</tr>
</tbody>
</table>

in addition of requiring a dynamic system to be stable, i.e., its response does not increase unbounded with time (a condition that is satisfied for a second order system provided that $\zeta \geq 0$) are the system should possess:

- faster and “instantaneous” response
- minimal overshoot above the desired value (i.e., relatively stable) and
- ability to reach and remain close to the desired reference value in the minimum time possible.

We will use these parameters to analyze the DC motor system under different form of controls and optimize the controller gains to achieve desired performance by end of this lab session.

## 4. Closed Loop Control Systems

Closed loop control system uses feedback to determine the actual input to the system. Feedback means that the information about the current states (e.g., position, velocity) of the system is used by the controller to continuously correct the actual input to the system in order to reach the desired states. Typically, such systems are expressed using block representation and the resulting transfer function as in Fig. 3. Here ($G(s)$) refers to transfer function of the actual plant (or physical system) being controlled, [$G_c(s)$] refers to the Laplace transform of the controller, ($H(s)$) refers to output feedback transfer function and ($E(s)$) is Laplace transform of the error signal term, ($E(s) = \mathcal{L}[u(t) - y(t)]$) that is input to the controller. The controller output, $U_p(s)$, is given by, $U_p(s) = G_c(s) E(s)$ which is input to the controlled plant.

In our experiments, we only use unity feedback systems, meaning, $H(s) = 1$. 
In control systems application, the input signal, \( U(s) \) for the system in Fig. 3 or motor voltage, \( V_m(s) \) for the DC motor plant in (2) as well as in Fig. 1, is actually the desired trajectory or a set point that we would like our system to execute or reach respectively over time. The controller, \( G_c(s) \), can be any type of controller with transducer depending on the application. Then, the plant input, \( U_p(s) \) is simply the actual controlled signal input to the plant to ensure it reaches the desired state. For example, in the case of DC motor plant, \( V_m \), will be desired angular trajectory, \( \theta_d(t) \) or set-point position, \( \theta_d \) for constant position and \( V_i \), will be actual controlled input signal in volts generated by the controller, \( G_c(s) \) fed to the DC motor.

Most commonly used type of controllers are the ones discussed below, that we will also study in this lab.

### 4.1 Proportional Control (\( K_p \))

A proportional controller (P) consists of only a constant gain block as shown in the block diagram Fig. 4. The output of the controller will be a constant multiple of the difference between the input signal and current state. For example, by adding a proportional gain block for a generic closed loop system in Fig. 3, the input to the controller is the error term, \( e(t) = u(t) - y(t) \) if \( H(s) = 1 \) and the output is just, \( K_p [u(t) - y(t)] \). This is in essence equivalent to a virtual spring attached between the actual system under study to an imaginary ideal system following your desired trajectory at that instant. So, the difference between the current and desired states will cause a spring-type force to tend the system towards desired trajectory. Using this analogy, it is now easy to understand that using high proportional gains can render the system increasingly stiff, thereby making the system output highly oscillatory. (Note: remember the example of driving the vehicle with bumpy-ride characteristics!!)

![Figure 4: Proportional Controller of DC Motor Plant](image)

### 4.2 Derivative Control (\( K_d \))

A derivative controller (D) adds a differential gain as shown in the Fig. 5. This type of controller will now act as a virtual damper connected between your actual system and the imaginary ideal system that exerts corrective damping force on the actual system to maintain the desired trajectory at the velocity level. These
are highly useful for systems that are inherently stiff or for systems that has high frequency noise. In practice however, a pure derivative controller is not used — check how the derivative controller is implemented in your simulink model !!! (Hint: Velocity is obtained by differentiation of position which by itself contains noise, so in the simulink models given to you, check how the derivative signal is filtered out). Nonetheless, these are commonly used in combination with proportional controller (PD controller) to damp out excessive oscillations and improve overall set-point tracking performance. Therefore, the error term in this case is differentiated by the controller \( \frac{d(e(t))}{dt} = \frac{d(u(t) - y(t))}{dt} \) and amplified by the controller gain, \( (K_d) \) and the overall output from the controller block, \( G_c(s) \) is given by, \( L\left(\frac{d(K_de(t))}{dt}\right) = K_dL\left(\frac{d(e(t))}{dt}\right) = K_dse(s) \).

(Recall that multiplication by \( s \) in the Laplace domain serves as taking the derivative in time domain provided \( e(t = 0) = u(t = 0) - y(t = 0) = 0 \) : \( u(0) = 0 \) and \( y(0) = 0 \) due to zero-initial condition assumptions as discussed in Sec. 3.

\[
\text{TF} = \frac{\Theta(s)}{V_m(s)} = \frac{aK_ds}{s^2 + bs}
\]

**Figure 5: Derivative Controller of DC Motor Plant**

### 4.3 Integral Controller \( (K_i) \)

Integral control (I) is implemented by integrating the error in system and using it as input signal to drive the controlled system as in Fig. 6. The integral term considers the history of the error (summation of errors), or how long and how far the measured process variable or system state has been from the set point over time. If \( e(t) \) is non-zero for any length of time (for example, positive), the control signal gets larger and large as time goes on. It thus forces the plant to react in the event that the plant output starts to drift. The primary advantage of this controller is that the gain can be tuned so that it drives steady-state error in the system to zero. The challenge however is that integral controllers are commonly used with P- type controller and error outputs due to these two are dependent on each other. Therefore, when including the integrating controller it must be carefully designed to ensure system stability.

\[
\text{TF} = \frac{\Theta(s)}{V_m(s)} = \frac{aK_i}{s^3 + bs^2 + aK_i}
\]

**Figure 6: Integral Control of DC Motor Plant**
4.4 PD Control Implementation

The proportional-derivative controller basically prescribes a new input into our system that is proportional to both the error signal, $e(t)$ and its derivative, $\dot{e}(t)$. You should be familiar with implementing a PD control of the motor model from the first week of this lab. So, the controller transfer function can be simplified as a sum of both the proportional and derivative controller transfer functions as: $G_c(s) = K_p + K_ds$ as given in the block diagram, Fig. 7. The final transfer function of such a system can be obtained as:

$$\frac{\Theta(s)}{V_m(s)} = \frac{aK_ds + aK_p}{s^2 + (b + aK_d)s + aK_p}$$

However, as we have already mentioned that in feedback control implementation, it is a common practice to replace $V_m$, by desired trajectory or a set point, which in the case of motor is, $\Theta_d(s) = L[\theta_d(t)]$. Substituting this in the above equation, we get:

$$\frac{\Theta(s)}{\Theta_d(s)} = \frac{aK_ds + aK_p}{s^2 + (b + aK_d)s + aK_p}$$

The input voltage fed into the DC motor is then given by the controller output, $V_i(t) = K_d(\dot{\theta}_d(t) - \dot{\theta}(t)) + K_p(\theta_d(t) - \theta(t))$, where $\theta_d(t)$ is the desired/target position. $K_d$ is called the derivative gain and $K_p$ is called the proportional gain as mentioned before.

To compare and validate different transient responses, it suffices only to compare the denominator of eqn. (3) and eqn.(5). Therefore, we obtain:

$$\omega_n^2 = aK_p \Rightarrow K_p = \frac{\omega_n^2}{a}$$

$$2\zeta\omega_n = aK_d + b$$

$$K_d = \frac{2\zeta\omega_n - b}{a}$$

The values $\omega_n$ and $\zeta$ to be substituted in the equations (6) and (7) can be determined from desired transient characteristics. Suppose if you are given maximum overshoot and rise time (or settling time), it will then be possible to calculate $\omega_n$ and $\zeta$ using the formula from 1.

4.5 PID- Control Implementation

Similar to PD control, PI- and PID- control shall be implemented by combining proportional-, derivative- and integral- control elements. Block representation of PID control is given in Fig. 8. Though several
methods exist to tune the P-, I-, and D- gains, there is no analytical method to calculate your controller gains as with PD control. The same is true with PI control of 2nd order system. You can tune these values by various methods — only one of the method (Ziegler Nichols Method) is given here or rather intuitively, which is what we will end up doing for this lab.

According to this method, first we will have to set the values of \( K_d \) and \( K_i \) to be zero, so that only proportional control is active. Then, \( K_p \) should be slowly increased until the system exhibits sustained oscillations. If at this point, the value \( K_p \) is represented as \( K_{p,\text{limit}} \) and the period of oscillations experimentally is found to be, \( T_{\text{limit}} \), then the values of \( K_d \) and \( K_i \) are obtained as in Table:2.

Once the controller gains are obtained, the resulting controller output fed to the DC motor plant will be, \( V(t) = K_d \left( \int (\theta_d(t) - \theta(t))dt \right) + K_i \left( \dot{\theta}_d(t) - \dot{\theta}(t) \right) + K_p (\theta_d(t) - \theta(t)) \), where \( \theta_d(t) \) is the desired/target position, \( K_p, K_i \) and \( K_d \) are controller gains mentioned before.

You should remember the values given in this table may not result in optimal response sometimes and might even cause high oscillatory response. In such scenarios, one may have to tune the gain around the calculated values “intuitively” and try to get improved responses. Even though powerful methods exist to determine optimal parameters, it is outside the scope of this course. So, we will use only this method to design our PID controller.

### Table 2: Tuning of PI- and PID- Controller Gains by Ziegler Nichols Method

<table>
<thead>
<tr>
<th>Control Mode</th>
<th>( K_p )</th>
<th>( K_i )</th>
<th>( K_d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>0.45( K_{p,\text{limit}} )</td>
<td>( \frac{1.2K_p}{T_{\text{limit}}} )</td>
<td>0</td>
</tr>
<tr>
<td>PID</td>
<td>0.6( K_{p,\text{limit}} )</td>
<td>( \frac{K_p}{0.5T_{\text{limit}}} )</td>
<td>0.125( K_pT_{\text{limit}} )</td>
</tr>
</tbody>
</table>

**Figure 8:** PID Control of DC Motor Plant

According to this method, first we will have to set the values of \( K_d \) and \( K_i \) to be zero, so that only proportional control is active. Then, \( K_p \) should be slowly increased until the system exhibits sustained oscillations. If at this point, the value \( K_p \) is represented as \( K_{p,\text{limit}} \) and the period of oscillations experimentally is found to be, \( T_{\text{limit}} \), then the values of \( K_d \) and \( K_i \) are obtained as in Table:2.

5. **Simulink Implementation**

Since QuaRC software is currently configured to run only with 64-bit MATLAB, when we only refer it as “MATLAB”, we imply 64- bit MATLAB version without any doubt. By now, you should have gotten familiar with Simulink. You can refer to to Ali’s handout for a more in-depth introduction to Simulink.

**DC Motor Model**

For the first part, copy the closed loop Simulink models of your 1st week of motor modeling into a new directory along with all MATLAB scripts that you may need to re-plot your results for post processing. Before actually starting the following, first try to refresh your control implementation for motor models. For
different closed loop system responses that you recorded during that lab, try to re-run the Simulink models and try to evaluate the transient characteristics both from the graph as well as using the MATLAB command explained below. Try to get an idea of tabulating the results similar to the one in Table:3 for all the model responses recorded in the week 1 lab. It is a good idea to present the tabulation of previous results in your report as well.

- Run the MATLAB code to initialize your model parameters and set $K_d = 0$ and $K_p = 50$ and run your Simulink model for a unit step input response
- You can use the following code to plot the recorded data from scope (you should also be able to find this section of code in MotorParam.m file that was given to you for the week 1’s motor modeling lab as well):

```matlab
% Plot first signal exported from Simulink scope
% if you changed the variable of ScopeData, make the changes in the code % below as well !!!
hFig = figure; % creates new figure everytime you run this line
plot(ScopeData.time,ScopeData.signals.values(:,1),'b--','linewidth',2);
hold on; % Future calls to 'plot' draw in the same figure
% Plot second signal exported from Simulink scope
plot(ScopeData.time,ScopeData.signals.values(:,2),'r','linewidth',2);
grid on;
title('Simulated Step Response of Closed Loop DC Motor System')
xlabel('Time(s)')
ylabel('Angular Position(rad)')
legend('Input: Desired Position $\theta^d(t)$',...
'Output: Actual Position $\theta(t)$','location','Southeast');
xlim([min(ScopeData.time)+1, 1.4+min(ScopeData.time)]);
%change the xlim values to adjust the display of graph
% ylim([0 1.2])

ymm = ylim;
xmm = xlim;
str = {{'K_p=' num2str(Kp)},{'K_d=' num2str(Kd) }};
text('Position',[(xmm(2)-xmm(1))*0.6+xmm(1) ...
(ymm(2)-ymm(1))*0.85+ymm(1)],'String', str, 'FontSize',13);
set(hFig,'position',[232 188 558 310]);
% to save the file as png , for other formats ref. MATLAB document
% for saveas
saveas(hFig,'ClosedLoopModelResponse_Case1_PD.png');

%%% Calculating Transient specifications automatically %%%
% MATLAB has its own in-built command that allows to calculate
% these parameters if the system response % data is recorded as
% three sets of arrays (output states, input signal and time array).
\begin{verbatim}
S = stepinfo(y,t,yfinal,'SettlingTimeThreshold',0.05,...
```

```
% where, S has all the desired parameters we are interested in,
% yfinal is the steady state value of system output states
% y is the acquired samples of system output states
% also, note the additional parameters in stepinfo command above
% In order to extract the right set of values from S, refer to the
% MATLAB documentation for the appropriate syntax.

- From the plotted graph, estimate the transient specifications as in Fig.9 and tabulate the result for proportional controller as in 3.

![Simulated Step Response of Closed Loop DC Motor System](image)

**Figure 9: Proportional Control of DC Motor Model**

- To design a PD controller, the following scheme based on your person no. to compute desired specification will be used (Assume, current maximum overshoot is $M_p$ and settling time is $T_s$ and the person number is xxxx-2457, if there are more than one member in your team, take an average):
  - Maximum overshoot ($M_{p}^{\text{new}}$) (sum of 7th and 8th digits): $(5+7) = 12\%$
  - % Reduction in Settling time desired: (5th 6th digits combined)$ = 24\% \therefore T_{s}^{\text{new}} = T_{s} \times (1-0.24)$

- You can then determine the new controller gains for your PD control as follows:
  1. Using the computed specifications for your group with the formulae in Table:1 try to compute the new natural frequency, $(\omega_{n}^{\text{new}})$ and new damping ratio, $\zeta_{n}^{\text{new}}$. [Hint: you can compute the new damping ratio using the maximum overshoot relation and the new natural frequency using the settling time relation for the specified values]
  2. For these new set of natural frequency and damping ratio, use the equations (6)-(7) under Sec.4.4 to compute the new gains for the PD control implementation as $K_{p}^{\text{new}}$ and $K_{d}^{\text{new}}$ and update the values in the proportional and derivative constant blocks in the Simulink model

- Rerun your Simulink model with modified gains and plot your results to determine transient specifications (both manually and from MATLAB)$^2$. Check if there is an improvement, of course !!!

$^2$The manually estimated values and values given by “StepInfo” command will be more or less the same. Nevertheless, try to record these specifications by both the method.
Now, intuitively, change the gains of your PD controller to further improve the transient response of your model without carrying out any of the above calculations. Replot and recompute the transient specifications again to confirm the improvement. You should be able to explain how you chose those gains.

To implement PI and PID controller, reset the values of $K_p$ and $K_d$ in the Simulink model to values of 50 and 0 respectively.

Before proceeding, you will have to add an integrator block to your Simulink model from Simulink library browser and follow the steps as in Fig.10.

Try to manually tune the $K_p$ and $K_i$ gains first and see if you are able to get a better response than just using proportional control. You should be now able to approximately estimate your overshoot as well as settling time by directly looking at the corresponding scope.

Now, manually tune the $K_d$ gain for adjusting the derivative control part and see if the performance could be any better than the PI- control. [Note: You can use the Ziegler-Nichol’s method discussed in Sec.4.5 and check if the method is useful for this particular application and provide your insights]

For the two illustrative cases (need not the best case) of PI- and PID- control simulation, plot the graphs ($\theta_{des}, V_i(t)$ and $\theta(t)$) against $t$ and tabulate your results as before. Check the controlled input voltage range fed to the motor model.

For each of the case above (P-, PD-, PI- and PID- control), simulate the model response for different inputs like sine, saw tooth wave etc. (need to be verified by TA). You can evaluate the tracking performance by plotting error signal against time, where ($errort Signal = ScopeData.signals.values(:,1) - ScopeData.signals.values(:,2)$) if same conventions are being followed for the scope block.

### 5.1 DC Motor Plant (Quanser SRV02)

We will now implement the position control of the DC motor subjected to a step input using Simulink. By now, you should have become familiar with the PD control of motor model. So, we follow the same steps for DC motor control as above and tabulate your results. If you need help to modify the Simulink model for the actual DC motor, please refer to the addendum document — *DCMotorSRV02_closedLoopInstruction.pdf*

### 6. Report Requirements

Your report for this lab should therefore include the following:

- Implementation of Proportional-, Derivative-, Integral- and other hybrid controllers for the motor model and tabulate the model responses similar to the Table:3

- Extend the same to the actual motor plant and determine the transient characteristics for each of the above case specifically for step input as in Table.3 and evaluate the performance of each control mode.

- For a given desired transient specification, design a PD control based on step input response and evaluate the responses for ramp, sine and saw tooth signal inputs

- Modify your input signal (signal generator block) to ramp, sine and saw tooth signal types and evaluate the tracking performance for the best control mode that you deduced from the previous exercise
Figure 10: Adding an Integrator Block for Position Control
Table 3: Summary of Position Control Experimental Results [Note: You can have multiple runs for each mode for different step input sizes]

References

## Appendix

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Nominal Value (SI Units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_{eq}$</td>
<td>Viscous damping coefficient</td>
<td>$4.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>$\eta_g$</td>
<td>Gearbox efficiency</td>
<td>0.9</td>
</tr>
<tr>
<td>$\eta_m$</td>
<td>Motor efficiency</td>
<td>0.69</td>
</tr>
<tr>
<td>$J_{eq} = \eta_g K_g^2 J_m + J_l$</td>
<td>Equivalent moment of inertia</td>
<td>$2.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>$J_m$</td>
<td>Motor moment of inertia</td>
<td>$3.87 \times 10^{-7}$</td>
</tr>
<tr>
<td>$K_g$</td>
<td>Gear ratio</td>
<td>$14 \times 5 = 70$</td>
</tr>
<tr>
<td>$K_m$</td>
<td>Back-emf constant</td>
<td>$7.67 \times 10^{-3}$</td>
</tr>
<tr>
<td>$K_t$</td>
<td>Motor-torque constant</td>
<td>$7.67 \times 10^{-3}$</td>
</tr>
<tr>
<td>$R_m$</td>
<td>Armature resistance</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Table 4: DC motor system parameters. These parameters are typically found on the motor vendor’s datasheets.