DC Motor Control System
EE2013 Project

Matric No: U066584J

February 22, 2010

1 Introduction

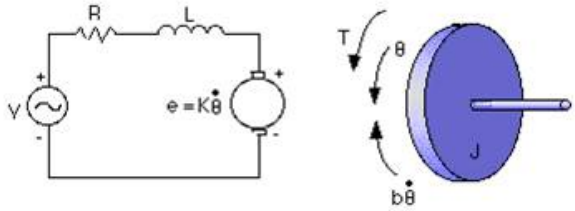

Figure 1: Simple DC Motor Control System

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide transitional motion. The electric circuit of the armature and the free body diagram of the rotor are shown in the Figure 1.

The motor torque, T, is related to the armature current, i, by a constant factor K. The back emf, e, is related to the rotational velocity by the following equations:

$$T = Ki$$

$$e = K\dot{\theta}$$

From Figure 1 above we can write the following equations based on Newton's law combined with Kirchhoff's law:

$$J\ddot{\theta} + b\dot{\theta} = Ki \tag{1}$$

$$L\frac{di}{dt} + Ri = V - K\dot{\theta} \tag{2}$$

The following values for the physical parameters are used.

1. Moment of inertia of the rotor, $J = 0.01kg.m^2/s^2$
2. Damping ratio of the mechanical system, $b = 0.1Nms$

3. Electromotive force constant, $K = 0.01 Nm/Amp$
4. Electric resistance, $R = 1 ohm$
5. Electric inductance, $L = 0.5 H$
6. The rotor and shaft are assumed to be rigid

2 Discussions

2.1 Represent in Matlab the open-loop transfer function $G(s)$, where the rotational speed ($d\theta/dt$) is the output and the voltage (V) is the input

From the equation (1) and (2), the state-space model of the system can be obtained as follows:

$$\ddot{\theta} = -\frac{b}{J}\dot{\theta} + \frac{K}{J}i$$
$$\dot{i} = -\frac{K}{L}\dot{\theta} - \frac{R}{L}i + \frac{V}{L}$$

Rewrite in the state-space form:

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1/L \end{pmatrix} V$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

where $x_1 = \dot{\theta}$ and $x_2 = i$

The following codes are typed in Matlab Command Window to generate the transfer function $G(s)$ of the system by converting the state-space representation to the transfer function representation:

```
>> J = 0.01; b = 0.1; K = 0.01; R = 1; L = 0.5;
>> A = [-b/J K/J; -K/L -R/L];
>> B = [0; 1/L];
>> C = [1 0];
>> D = 0;
>> motorTF = tf(ss(A,B,C,D))

Transfer function:
        2
------------------
s^2 + 12 s + 20.02

>>
```

2.2 Plot the unit step response of the plant, determine its rising time, settling time, overshoot, and steady state error.

To plot the unit step response of the plant, simple `step` command can be called. In order to determine the information of the step response such as rising time, settling time and overshoot, the command `stepinfo` is used as shown below.

```
>> step(motorTF)
>> stepinfo(motorTF)

ans =

        RiseTime: 1.1362
    SettlingTime: 2.0653
     SettlingMin: 0.0901
     SettlingMax: 0.0999
       Overshoot: 0
      Undershoot: 0
            Peak: 0.0999
        PeakTime: 5.2388

>>
```

Using Final Value Theorem, steady state error, $e = 1 - \lim_{s\to 0} sY(s) = 1 - \lim_{s\to 0} G(s) = \frac{901}{1001}$

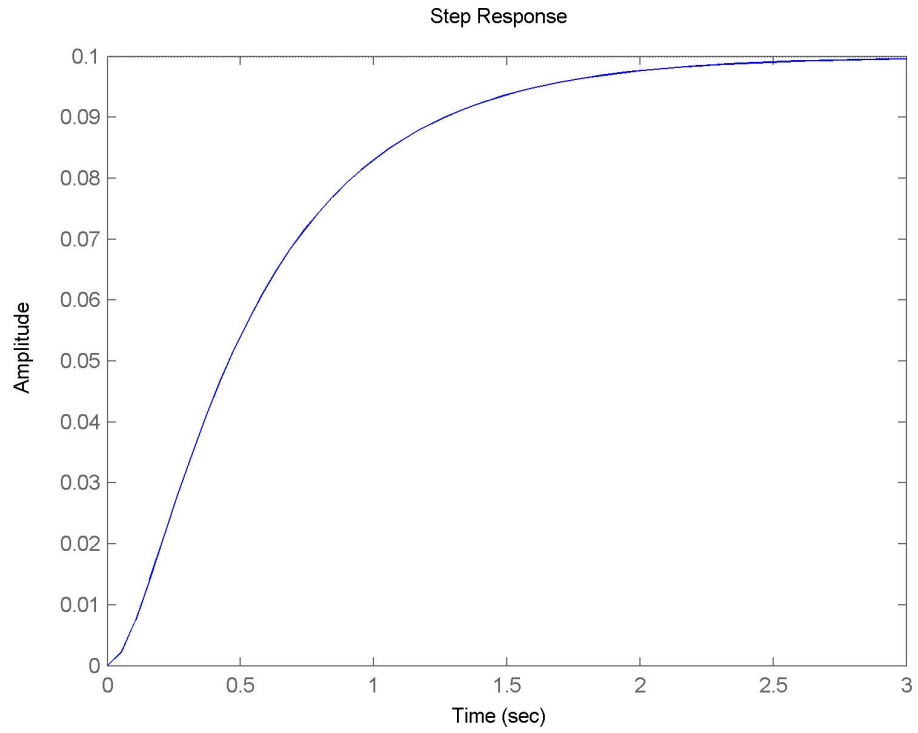

Figure 2: Step Response of the Open Loop System

2.3 Draw Bode, Nyquist plots and Root locus in Matlab for the plant.

The following Matlab codes are entered to plot the Nyquist, Bode and Root locus of the systems:

```
>> nyquist(motorTF)
```

```
>> bode(motorTF)
>> rlocus(motorTF)
>>
```

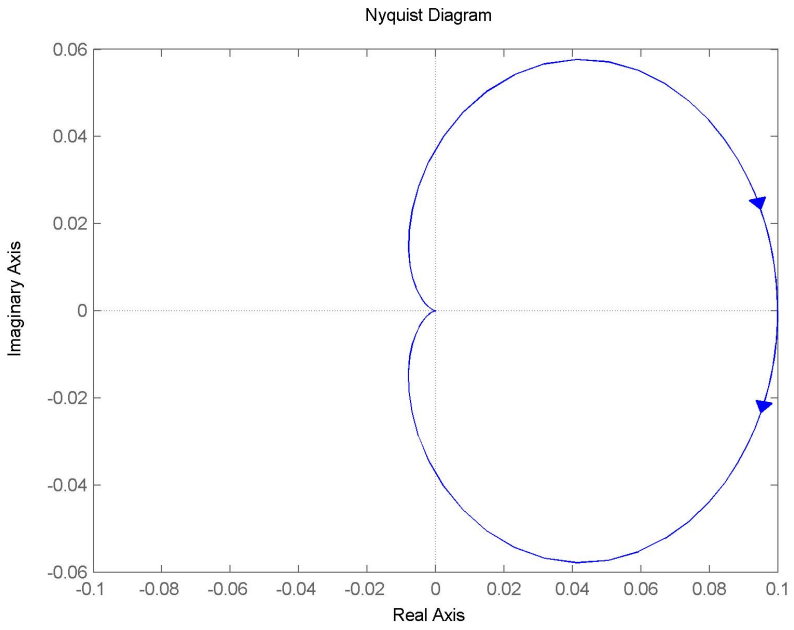

Figure 3: Nyquist Plot of the System

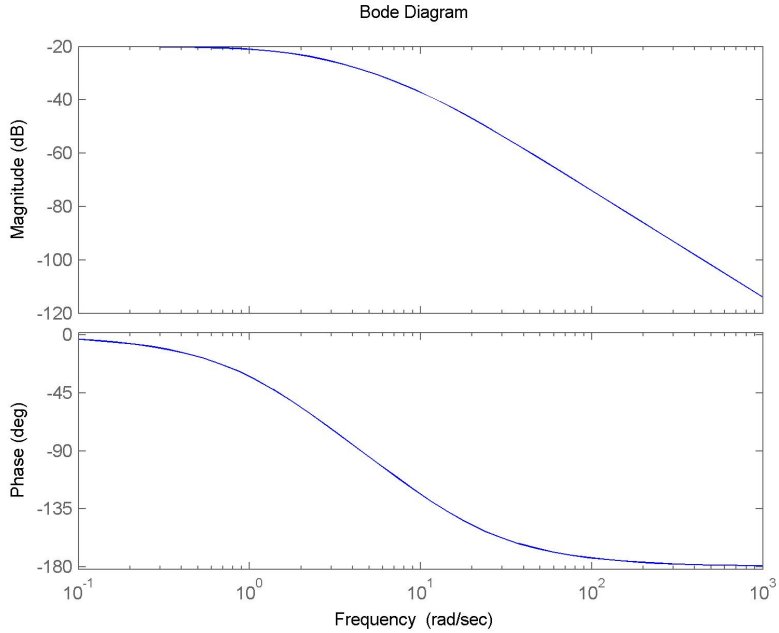

Figure 4: Bode Diagram of the System

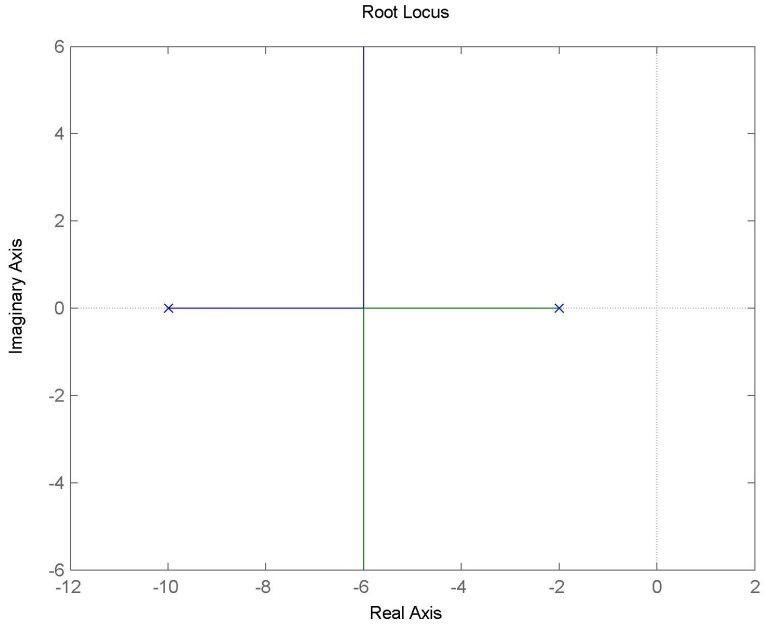

Figure 5: Root Locus of the System

2.4 Using root locus method to design a controller such that the closed loop motor speed output, with respect to an unit step input, should have: Settling time less than 2 seconds, Overshoot less than 5%, Steady-state error less than 1%

For settling time lesser than 2 seconds and overshoot lesser than 5%, the following conditions must be met:

$$M_p = e^{-\pi \frac{\zeta}{\sqrt{1-\zeta^2}}} < 5\% \implies \zeta > 0.7$$

$$t_s \cong \frac{4}{\zeta \omega_n} < 2 \implies \zeta \omega_n > 2$$

Since $\zeta\omega_n$ is the negative real axis of the root locus diagram, a straight line at real axis = -2 is drawn together with the root locus figure. After generating the graph, the function `rlocfind` is used to determine the desired gain and pole locations. The following Matlab code is entered to generate the graph in Figure 6.

```
>> rlocus(motorTF)
>> sgrid(0.7, 50)
>> hold on
>> y = linspace(-10,10,100);
>> x = -2*ones(1,100);
>> plot(x,y,'k--')
>> [kd,poles] = rlocfind(motorTF)
Select a point in the graphics window

selected_point =
```

```
  -6.0065 + 5.3299i


kd =

   22.1938


poles =

  -6.0000 + 5.3299i
  -6.0000 - 5.3299i

>>
```

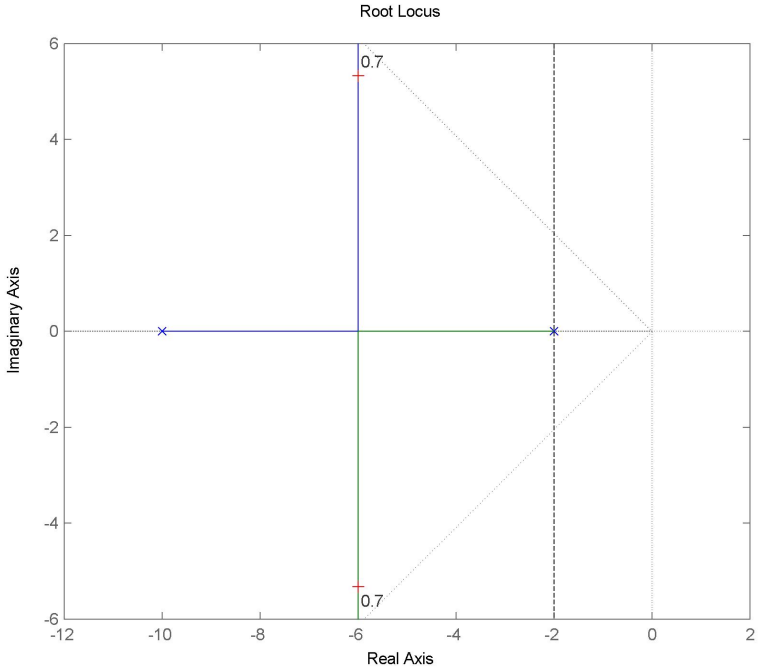

Figure 6: Root Locus of the System

To test whether the current controller design meet the specifications, step response of the closed loop system is plotted using Matlab:

```
>> motorCL = feedback(kd*motorTF,1)

Transfer function:
      44.39
------------------
s^2 + 12 s + 64.41
```

```
>> step(motorCL)
>>
```

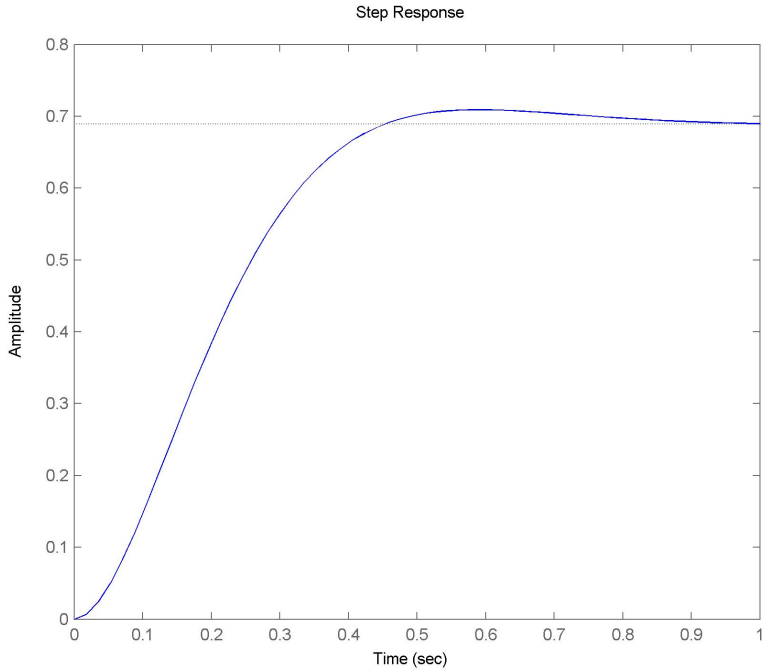

Figure 7: Step Response of the System

Clearly, the closed loop system does not meet the requirement in terms of steady state errors. To reduce the steady state error, a lag compensator can be used. However, adding a lag compensator alone might cause the system to be unstable. Hence a lead-lag compensator is designed to handle this problem:

```
>>leadlag = zpk([-3 -4],[-0.1 -30],1)

Zero/pole/gain:
 (s+3) (s+4)
--------------
(s+0.1) (s+30)

>> sys = motorTF*leadlag;
>> rlocus(sys)
>> sgrid(0.7,50)
>> motorFB = feedback(237*sys,1)

Zero/pole/gain:
           474 (s+4) (s+3)
-----------------------------------------
(s^2 + 5.642s + 8.931) (s^2 + 36.46s + 643.6)
```

```
>> step(motorFB);
```

Using the `rlocfind` function in Matlab, the desired gain that satisfies the requirement is obtained as 237. As observed in Figure 8, the settling time is around 0.64 seconds, the overshoot is about 0.6% while the steady state error is 1%.

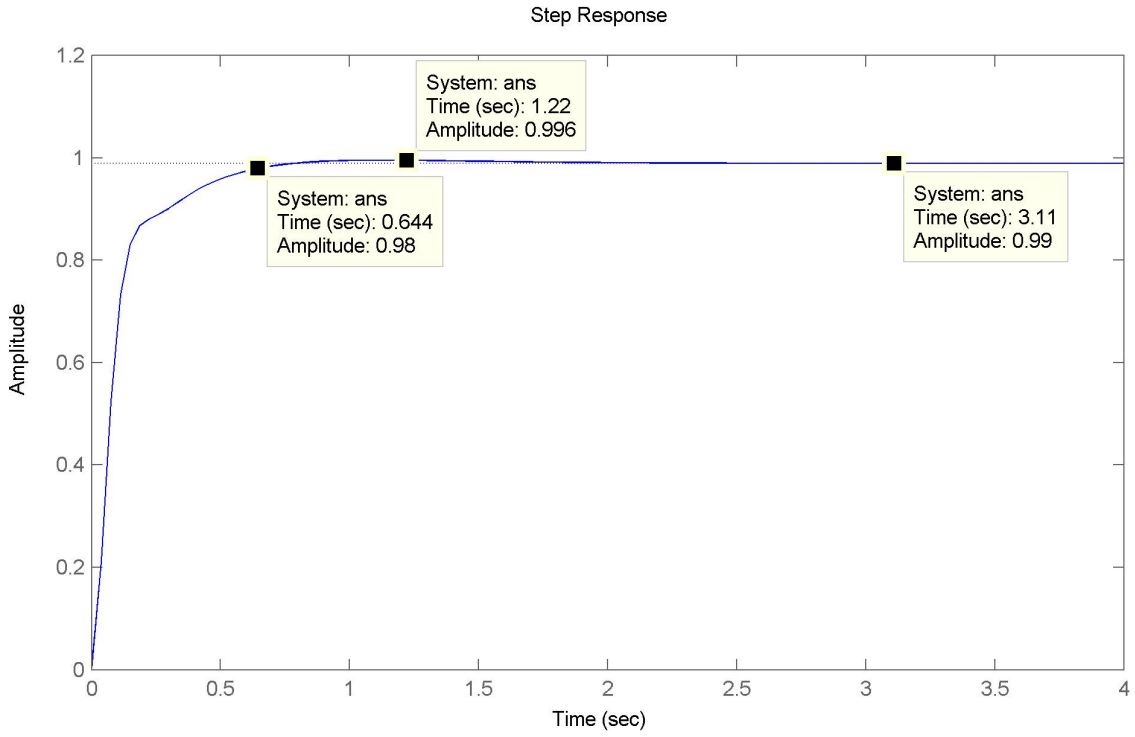

Figure 8: Step Response of the System

2.5 Determine the phase and gain margin after the design

The gain and phase margin of the system can be obtained by entering the following codes:

```
>> margin(sys)
```

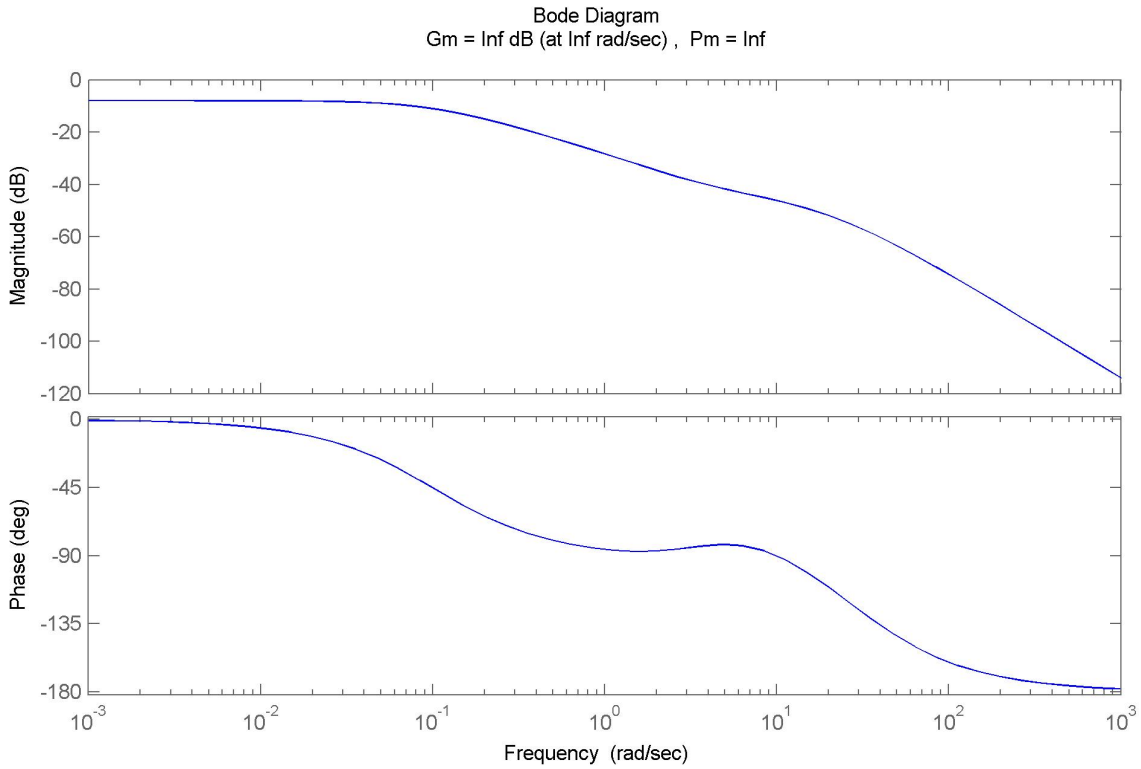

Figure 9: Phase and Gain Margins

As observed, the DC gain of the system does not reach 0 dB due to the steady state error in the design. Hence the phase margin of the system is infinity.

3 Conclusion

Matlab can be used to model a control system by 4 different representation methods. Besides modelling the control system, Matlab is useful to design the controller and simulate the controlled system. However, experiences and knowledge about control are much needed in tuning the controller.