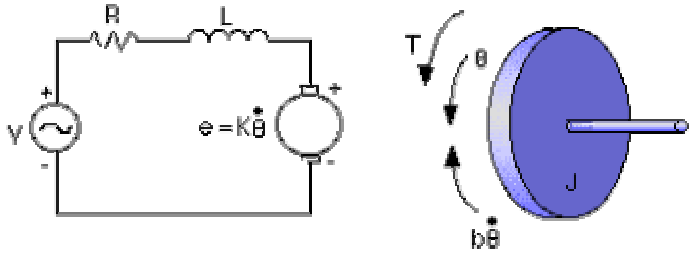
DC Motor Speed Modeling

Physical setup and system equations

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 following figure:



For this example, we will assume the following values for the physical parameters. These values were derived by experiment from an actual motor in Carnegie Mellon's undergraduate controls lab.

* moment of inertia of the rotor (J) = 0.01 kg.m^2/s^2
* damping ratio of the mechanical system (b) = 0.1 Nms
* electromotive force constant (K=Ke=Kt) = 0.01 Nm/Amp
* electric resistance (R) = 1 ohm
* electric inductance (L) = 0.5 H
* input (V): Source Voltage
* output (theta): position of shaft
* The rotor and shaft are assumed to be rigid

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

$$T = K_t i$$

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

In SI units (which we will use), **Kt** (armature constant) is equal to **Ke** (motor constant). From the figure above we can write the following equations based on Newton's law combined with Kirchhoff's law:

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

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

Transfer Function

Using Laplace Transforms, the above modeling equations can be expressed in terms of s.

$$s(Js+b)\Theta(s) = KI(s)$$

$$(Ls+R)I(s) = V - Ks\Theta(s)$$

By eliminating I(s) we can get the following open-loop transfer function, where the rotational speed is the output and the voltage is the input.

$$\frac{\dot{\theta}}{V} = \frac{K}{(Js+b)(Ls+R)+K^2}$$

State-Space

In the state-space form, the equations above can be expressed by choosing the rotational speed and electric current as the state variables and the voltage as an input. The output is chosen to be the rotational speed.

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix}=\begin{bmatrix}-\frac{b}{J} & \frac{K}{J}\\ -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix}+\begin{bmatrix}0\\ \frac{1}{L}\end{bmatrix}V$$

$$\dot{\theta}=\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix}$$

Design requirements

First, our uncompensated motor can only rotate at 0.1 rad/sec with an input voltage of 1 Volt (this will be demonstrated later when the open-loop response is simulated). Since the most basic requirement of a motor is that it should rotate at the desired speed, the steady-state error of the motor speed should be less than 1%. The other performance requirement is that the motor must accelerate to its steady-state speed as soon as it turns on. In this case, we want it to have a settling time of 2 seconds. Since a speed faster than the reference may damage the equipment, we want to have an overshoot of less than 5%.

If we simulate the reference input (r) by an unit step input, then the motor speed output should have:

- Settling time less than 2 seconds
- Overshoot less than 5%

- Steady-state error less than 1%

Matlab representation and open-loop response

Transfer Function

We can represent the above transfer function into Matlab by defining the numerator and denominator matrices as follows:

$$num = K$$
$$den = (Js + b)(Ls + R) + K^2$$

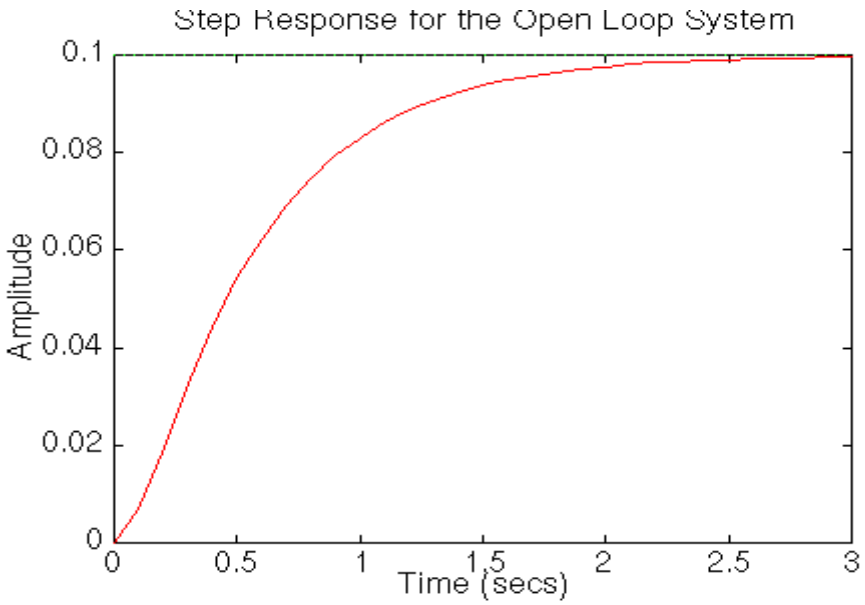
Create a new m-file and enter the following commands:

```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;
num=K;
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];
```

Now let's see how the original open-loop system performs. Add the following commands onto the end of the m-file and run it in the Matlab command window:

```
step(num,den,0:0.1:3)
title('Step Response for the Open Loop System')
```

You should get the following plot:



From the plot we see that when 1 volt is applied to the system, the motor can only achieve a maximum speed of 0.1 rad/sec, ten times smaller than our desired speed. Also, it takes the motor 3 seconds to reach its steady-state speed; this does not satisfy our 2 seconds settling time criterion.

State-Space

We can also represent the system using the state-space equations. Try the following commands in a new m-file.

```
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;

step(A, B, C, D)
```

Frequency Design Method for DC Motor Speed Control

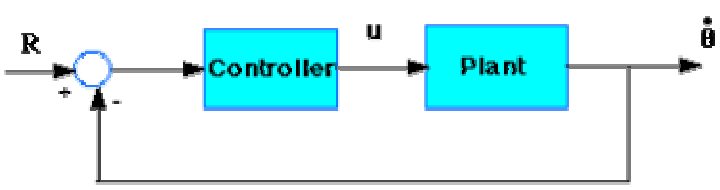
From the main problem, the dynamic equations and the open-loop transfer function of DC Motor Speed are:

$$s(Js + b)\Theta(s) = KI(s)$$

$$(Ls + R)I(s) = V - Ks\Theta(s)$$

$$\frac{\dot{\theta}}{V} = \frac{K}{(Js + b)(Ls + R) + K^2}$$

and the system schematic looks like:



With the 1 rad/sec step input, the design criteria are:

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 1%

Create a new m-file and type in the following commands (refer to the main problem for the details of getting those commands).

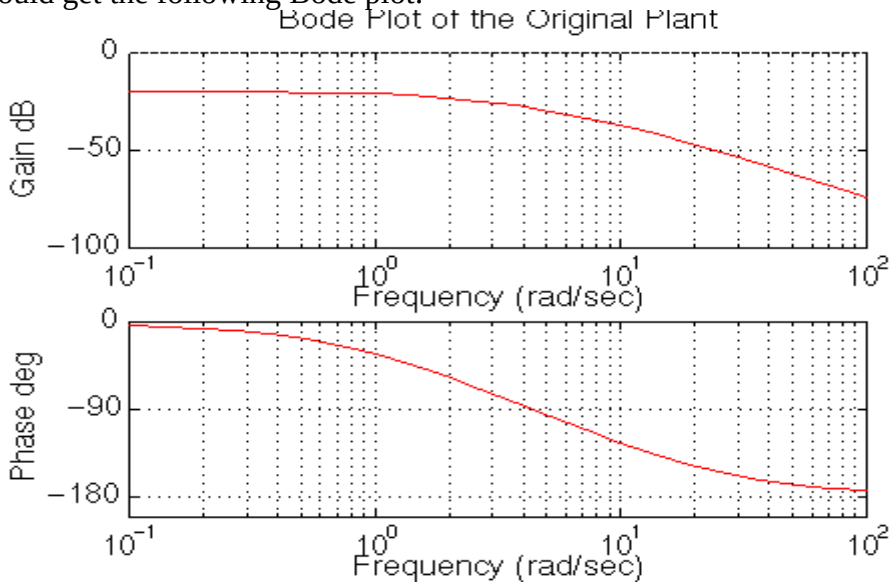
```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;
num=K;
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];
```

Drawing the original Bode plot

The main idea of frequency-based design is to use the Bode plot of the open-loop transfer function to estimate the closed-loop response. Adding a controller to the system changes the open-loop Bode plot, therefore changing the closed-loop response. Let's first draw the Bode plot for the original open-loop transfer function. Add the following code to the end of your m-file, and then run it in the Matlab command window.

```
bode(num,den)
```

You should get the following Bode plot:



Adding proportional gain

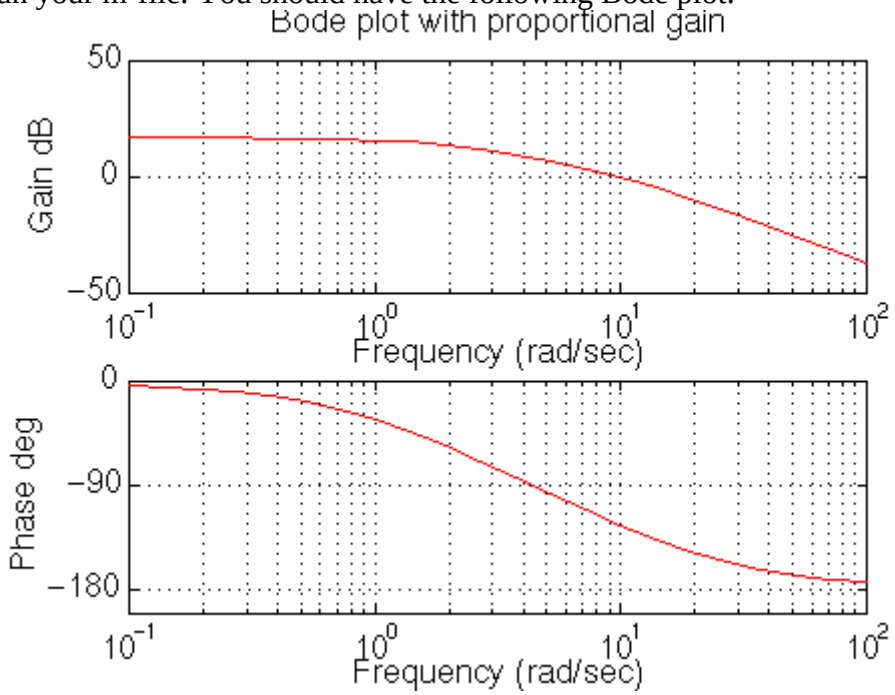
From the bode plot above, we see that the phase margin can be greater than about 60 degrees if w is less than 10 rad/sec. Let's add gain to the system so the bandwidth frequency is 10 rad/sec, which will give us a phase margin of about 60 degrees. To find the gain at 10 rad/sec, you can try to read it off the Bode plot (it looks to be slightly more than -40 dB, or 0.01 in magnitude). The bode command, invoked with left-hand arguments, can also be used to give you the exact magnitude:

```
[mag,phase,w] = bode(num,den,10)
mag =

    0.0139
```

To have a gain of 1 at 10 rad/sec, multiply the numerator by 1/0.0139 or approximately 72.

```
num = 70*num
```

and rerun your m-file. You should have the following Bode plot:

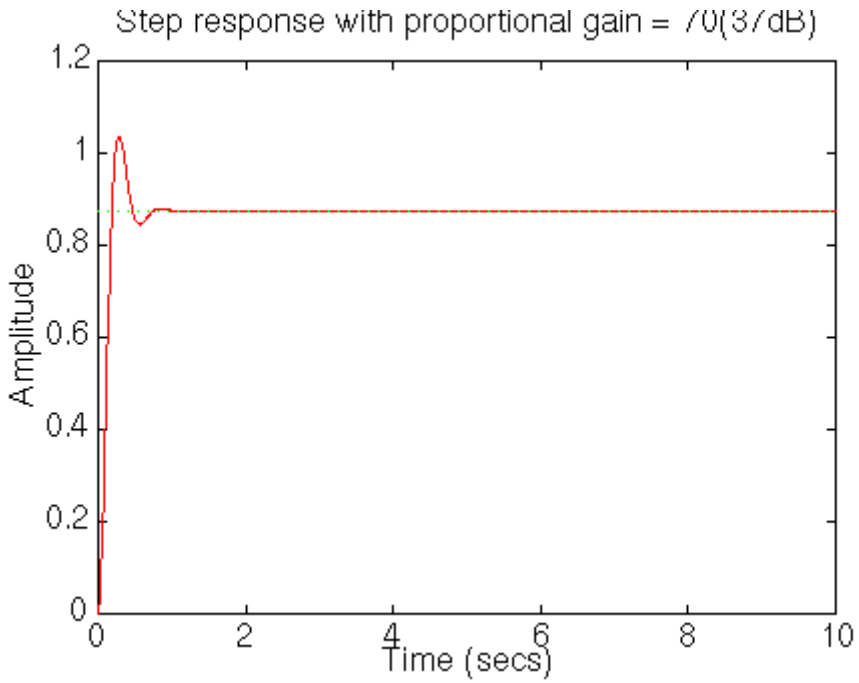


Plotting the closed-loop response

From the plot above we see that the phase margin is now quite large. Let's see what the closed-loop response look like. Add a % in front of the bode commands and add the following code to the end of your m-file:

```
[numc,denc]=cloop(num, den, -1);
t=0:0.01:10;
step(numc,denc,t)
```

You will see the following plot:



The settling time is fast enough, but the overshoot and the steady-state error are too high. The overshoot can be reduced by reducing the gain a bit to get a higher phase margin, but this would cause the steady-state error to increase. A lag controller is probably needed.

Adding a lag controller

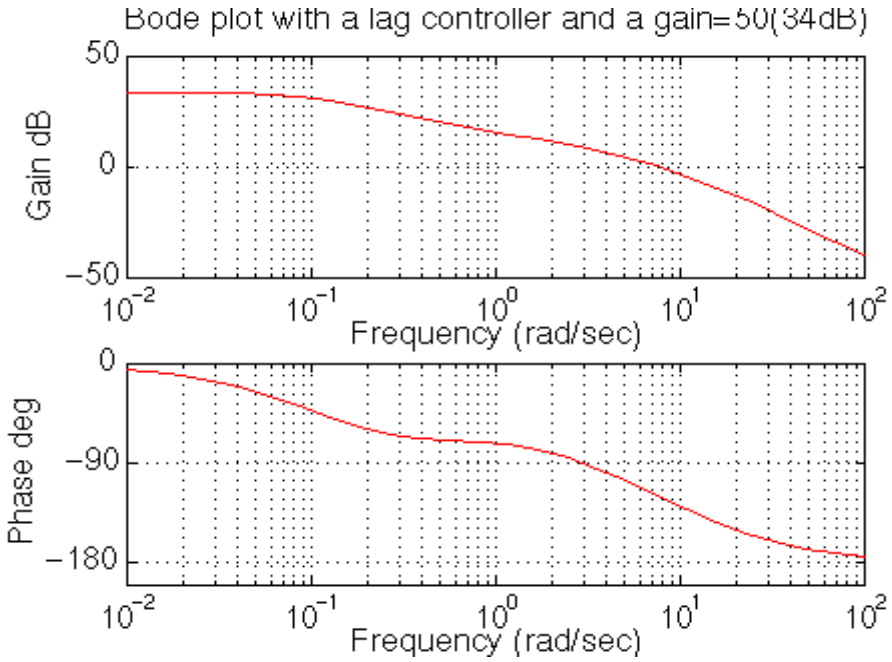
We can add a lag controller to reduce the steady-state error. At the same time, we should try to reduce the overshoot by reducing the gain. Let's reduce the gain to 50, and try a lag controller of

$$\frac{(s+1)}{(s+0.1)}$$

which should reduce the steady-state error by a factor of 1/0.01 = 100 (but could increase the settling time). Go back and change your m-file so it looks like the following:

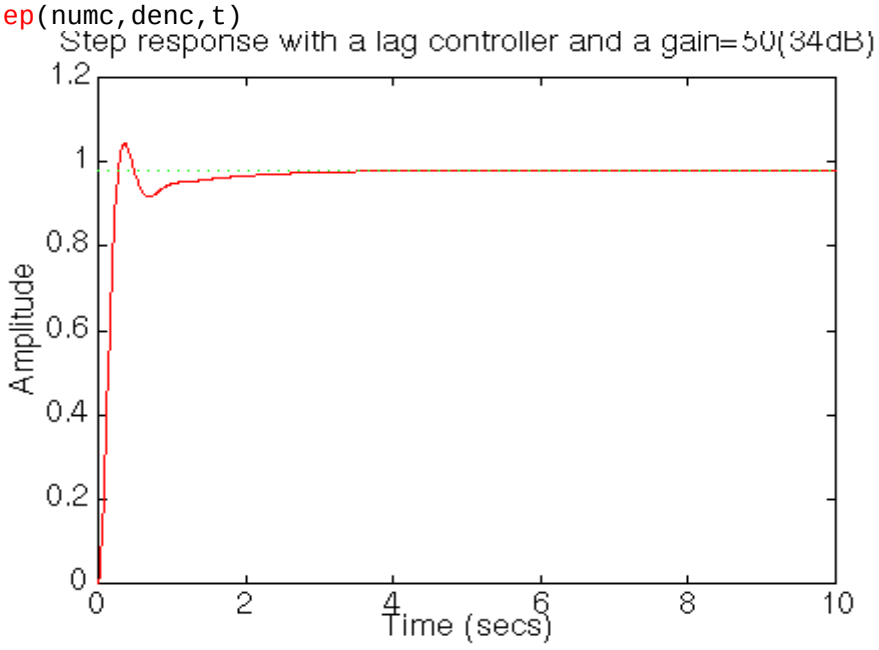
```
num=K;
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];
num=50*K;

z=1;
p=0.1;
numa=[1 z];
dena=[1 p];
numb=conv(num,numa);
denb=conv(den,dena);

bode(numb,denb)
```

Rerun the file and you will get this plot:



The phase margin looks good. The steady-state error is predicted to be about 1/40dB or 1%, as desired. Close the loop and look at the step response. Add the following lines of code to the end of you m-file and rerun.

```
[numc,denc]=cloop(numb, denb, -1);
t=0:0.01:10;
step(numc,denc,t)
```



Now you have a step response that meets the design requirements. The steady-state error is less than 1%, the overshoot is about 5%, and the settling time is about 2 seconds.