实验五　频率响应法的控制系统设计

一.　实验目的

1.　熟悉并掌握 MATLAB 的工作环境。

2.　了解频率响应法的控制的基本理论。

3.　在 MATLAB 工作环境下，选择适当的例子，实现频率响应法控制，讨论控制效果。

二.　实验内容

The frequency response method may be less intuitive than other methods you have studied previously. However, it has certain advantages, especially in real-life situations such as modeling transfer functions from physical data.

The frequency response of a system can be viewed two different ways: via the Bode plot or via the Nyquist diagram. Both methods display the same information; the difference lies in the way the information is presented. We will study both methods in this tutorial.

The frequency response is a representation of the system's response to sinusoidal inputs at varying frequencies. The output of a linear system to a sinusoidal input is a sinusoid of the same frequency but with a different magnitude and phase. The **frequency response** is defined as the magnitude and phase differences between the input and output sinusoids. In this tutorial, we will see how we can use the open-loop frequency response of a system to predict its behavior in closed-loop.

To plot the frequency response, we create a vector of frequencies (varying between zero or "DC" and infinity) and compute the value of the plant transfer function at those frequencies. If G(s) is the open loop transfer function of a system and w is the frequency vector, we then plot G(j*w) vs. w. Since G(j*w) is a complex number, we can plot both its magnitude and phase (the Bode plot) or its position in the complex plane (the Nyquist plot). More information is available on plotting the frequency response.

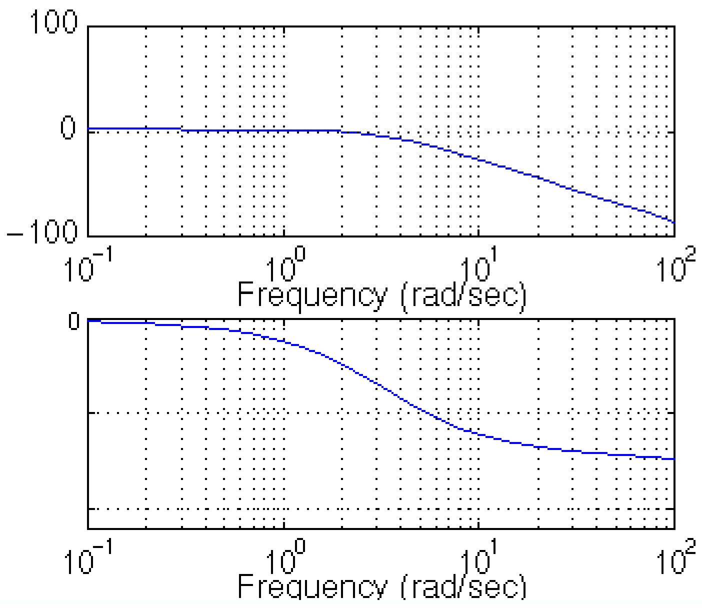
Bode Plots

As noted above, a Bode plot is the representation of the magnitude and phase of G(j*w) (where the frequency vector w contains only positive frequencies). To see the Bode plot of a transfer function, you can use the Matlab bode command. For example,

```
        bode(50,[1 9 30 40])
```

displays the Bode plots for the transfer function:

```
             50
  -----------------------
  s^3 + 9 s^2 + 30 s + 40
```



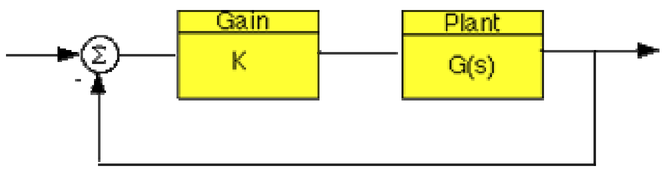
Please note the axes of the figure. The frequency is on a logarithmic scale, the phase is given in degrees, and the magnitude is given as the gain in decibels.

Note: a decibel is defined as 20*log10 (|G(j*w|)

Click here to see a few simple Bode plots.

Gain and Phase Margin

Let's say that we have the following system:



where K is a variable (constant) gain and G(s) is the plant under consideration. The **gain margin** is defined as the change in open loop gain required to make the system unstable. Systems with greater gain margins can withstand greater changes in system parameters before becoming unstable in closed loop.

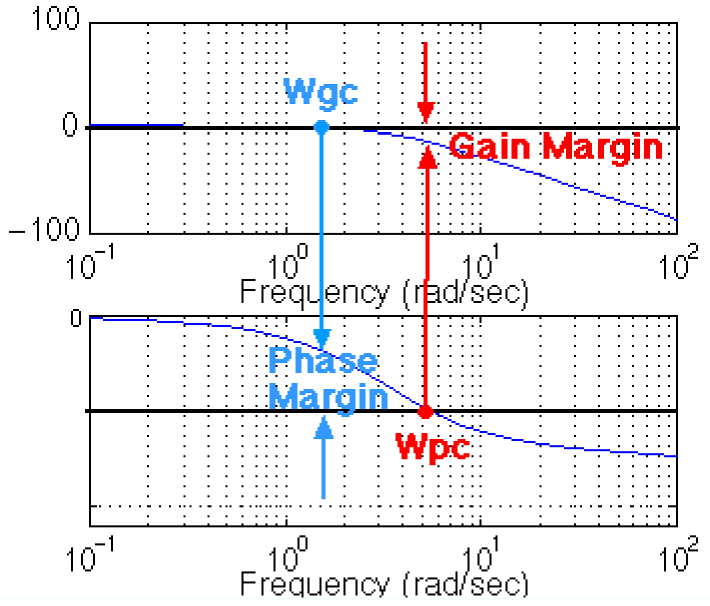
Keep in mind that unity gain in magnitude is equal to a gain of zero in dB.

The **phase margin** is defined as the change in open loop phase shift required to make a closed loop system unstable.

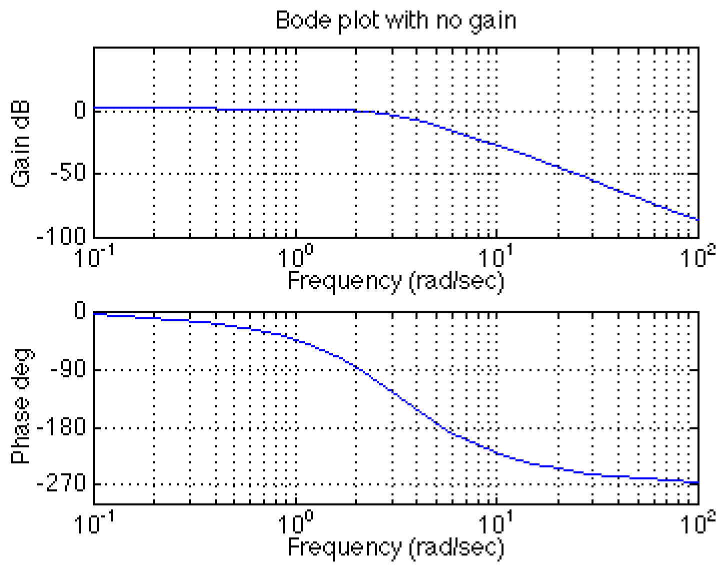
The phase margin also measures the system's tolerance to time delay. If there is a time delay greater than 180/Wpc in the loop (where Wpc is the frequency where the phase shift is 180 deg), the system will become unstable in closed loop. The time delay can be thought of as an extra block in the forward path of the block diagram that adds phase to the system but has no effect the gain. That is, a time delay can be represented as a block with magnitude of 1 and phase w*time_delay (in radians/second).

For now, we won't worry about where all this comes from and will concentrate on identifying the gain and phase margins on a Bode plot:

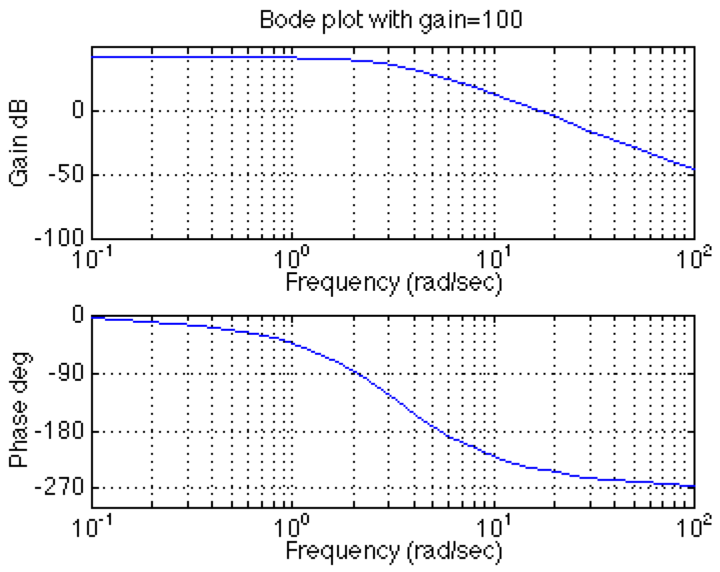
The phase margin is the difference in phase between the phase curve and -180 deg at the point corresponding to the frequency that gives us a gain of 0dB (the gain cross over frequency, Wgc). Likewise, the gain margin is the difference between the magnitude curve and 0dB at the point corresponding to the frequency that gives us a phase of -180 deg (the phase cross over frequency, Wpc).



One nice thing about the phase margin is that you don't need to replot the Bode in order to find the new phase margin when changing the gains. If you recall, adding gain only shifts the magnitude plot up. This is the equivalent of changing the y-axis on the magnitude plot. Finding the phase margin is simply the matter of finding the new cross-over frequency and reading off the phase margin. For example, suppose you entered the command bode(50,[1 9 30 40]). You will get the following bode plot:



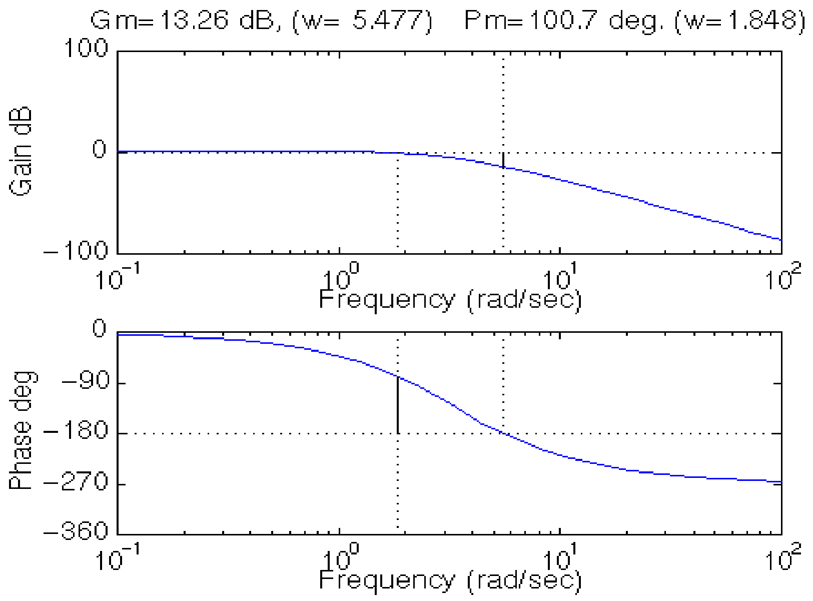
You should see that the phase margin is about 100 degrees. Now suppose you added a gain of 100, by entering the command bode(100*50,[1 9 30 40]). You should get the following plot (note I changed the axis so the scale would be the same as the plot above, your bode plot may not be exactly the same shape, depending on the scale used):



As you can see the phase plot is exactly the same as before, and the magnitude plot is shifted up by 40dB (gain of 100). The phase margin is now about -60 degrees. This same result could be achieved if the y-axis of the magnitude plot was shifted down 40dB. Try this, look at the first Bode plot, find where the curve crosses the -40dB line, and read off the phase margin. It should be about -60 degrees, the same as the second Bode plot.

We can find the gain and phase margins for a system directly, by using Matlab. Just enter the margin command. This command returns the gain and phase margins, the gain and phase cross over frequencies, and a graphical representation of these on the Bode plot. Let's check it out:

```
margin(50,[1 9 30 40])
```



Bandwidth Frequency

The bandwidth frequency is defined as the frequency at which the **closed-loop** magnitude response is equal to -3 dB. However, when we design via frequency response, we are interested in predicting the closed-loop behavior from the open-loop response. Therefore, we will use a second-order system approximation and say that the bandwidth frequency equals the frequency at which the **open-loop** magnitude response is between -6 and - 7.5dB, assuming the open loop phase response is between -135 deg and -225 deg. For a complete derivation of this approximation, consult your textbook.

If you would like to see how the bandwidth of a system can be found mathematically from the closed-loop damping ratio and natural frequency, the relevant equations as well as some plots and Matlab code are given on our Bandwidth Frequency page.

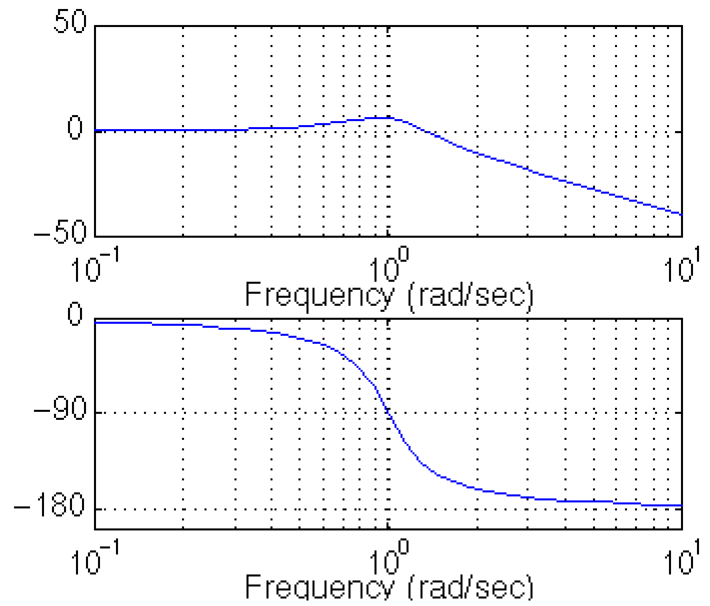
In order to illustrate the importance of the bandwidth frequency, we will show how the output changes with different input frequencies. We will find that sinusoidal inputs with frequency less than Wbw (the bandwidth frequency) are tracked "reasonably well" by the system. Sinusoidal inputs with frequency greater than Wbw are attenuated (in magnitude) by a factor of 0.707 or greater (and are also shifted in phase).

Let's say that we have the following closed-loop transfer function representing a system:

```
       1
---------------
s^2 + 0.5 s +  1
```

First of all, let's find the bandwidth frequency by looking at the Bode plot:

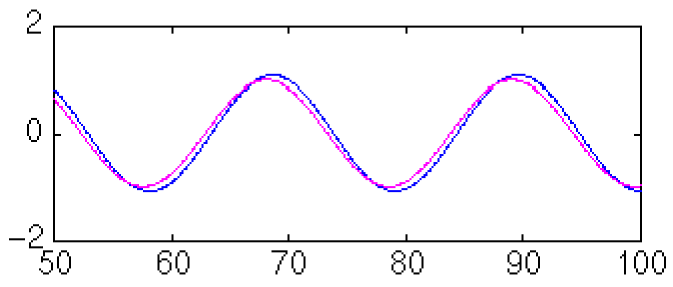
```
bode (1, [1 0.5 1 ])
```



Since this is the closed-loop transfer function, our bandwidth frequency will be the frequency corresponding to a gain of -3 dB. looking at the plot, we find that it is approximately 1.4 rad/s. We can also read off the plot that for an input frequency of 0.3 radians, the output sinusoid should have a magnitude about one and the phase should be shifted by perhaps a few degrees (behind the input). For an input frequency of 3 rad/sec, the output magnitude should be about -20dB (or 1/10 as large as the input) and the phase should be nearly -180 (almost exactly out-of-phase). We can use the lsim command to simulate the response of the system to sinusoidal inputs.

First, consider a sinusoidal input with a **frequency lower than Wbw**. We must also keep in mind that we want to view the steady state response. Therefore, we will modify the axes in order to see the steady state response clearly (ignoring the transient response).

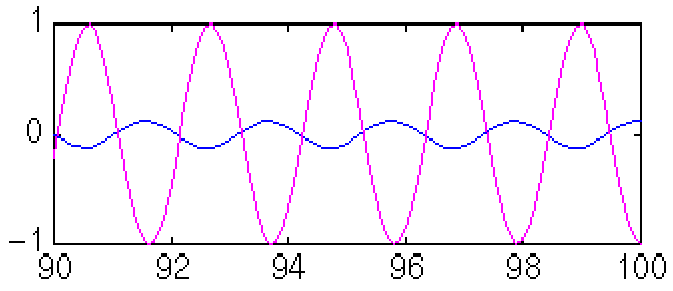
```
w= 0.3;
num = 1;
den = [1 0.5 1 ];
t=0:0.1:100;
u = sin(w*t);
[y,x] = lsim(num,den,u,t);
plot(t,y,t,u)
axis([50,100,-2,2])
```



Note that the output (blue) tracks the input (purple) fairly well; it is perhaps a few degrees behind the input as expected.

However, if we set the frequency of the input **higher than the bandwidth frequency** for the system, we get a very distorted response (with respect to the input):

```
w = 3;
num = 1;
den = [1 0.5 1 ];
t=0:0.1:100;
u = sin(w*t);
[y,x] = lsim(num,den,u,t);
plot(t,y,t,u)
axis([90, 100, -1, 1])
```



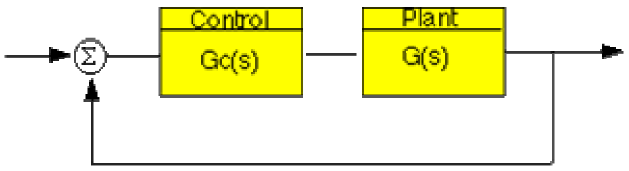
Again, note that the magnitude is about 1/10 that of the input, as predicted, and that it is almost exactly out of phase (180 degrees behind) the input. Feel free to experiment and view the response for several different frequencies w, and see if they match the Bode plot.

Closed-loop performance

In order to predict closed-loop performance from open-loop frequency response, we need to have several concepts clear:

- The system must be stable in open loop if we are going to design via Bode plots.
- If the gain cross over frequency is less than the phase cross over frequency(i.e. Wgc < Wpc), then the closed-loop system will be stable.
- For second-order systems, the closed-loop damping ratio is approximately equal to the phase margin divided by 100 if the phase margin is between 0 and 60 deg. We can use this concept with caution if the phase margin is greater than 60 deg.
- For second-order systems, a relationship between damping ratio, bandwidth frequency and settling time is given by an equation described on the bandwidth page.
- A very rough estimate that you can use is that the bandwidth is approximately equal to the natural frequency.

Let's use these concepts to design a controller for the following system:



Where Gc(s) is the controller and G(s) is:

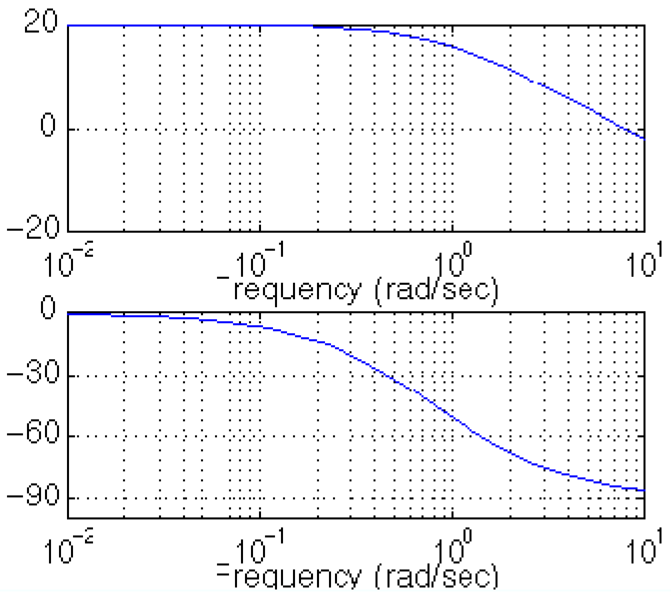
$$\frac{10}{1.25s + 1}$$

The design must meet the following specifications:

- Zero steady state error.
- Maximum overshoot must be less than 40%.
- Settling time must be less than 2 secs.

There are two ways of solving this problem: one is graphical and the other is numerical. Within Matlab, the graphical approach is best, so that is the approach we will use. First, let's look at the Bode plot. Create an m-file with the following code:

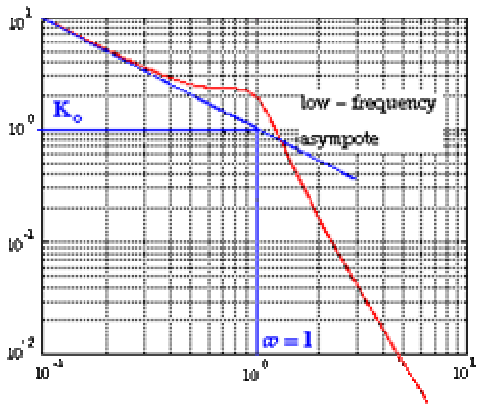
```
num = 10;
den = [1.25,1];
bode(num, den)
```



There are several several characteristics of the system that can be read directly from this Bode plot. First of all, we can see that the bandwidth frequency is around 10 rad/sec. Since the bandwidth frequency is roughly the same as the natural frequency (for a second order system of this type), the rise time is 1.8/BW=1.8/10=1.8 seconds. This is a rough estimate, so we will say the rise time is about 2 seconds.

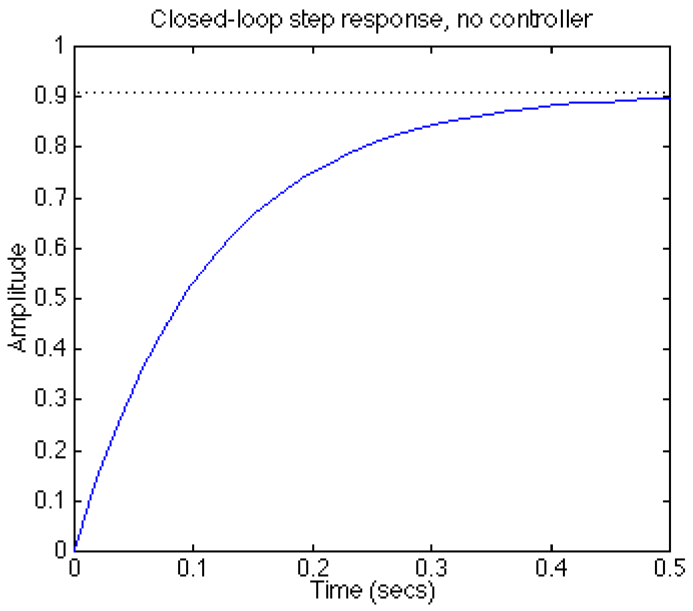
The phase margin for this system is approximately 95 degrees. This corresponds to a damping of PM/100=95/100=0.95. Plugging in this value into the equation relating overshoot and the damping ratio (or consulting a plot of this relation), we find that the damping ratio corresponding to this overshoot is approximately 1%. The system will be close to being overdamped.

The last major point of interest is steady-state error. The steady-state error can be read directly off the Bode plot as well. The constant (Kp, Kv, or Ka) are located at the intersection of the low frequency asymptote with the w=1 line. Just extend the low frequency line to the w=1 line. The magnitude at this point is the constant. Since the Bode plot of this system is a horizontal line at low frequencies (slope = 0), we know this system is of type zero. Therefore, the intersection is easy to find. The gain is 20dB (magnitude 10). What this means is that the constant for the error function it 10. Click here to see the table of system types and error functions. The steady-state error is 1/(1+Kp)=1/(1+10)=0.091. If our system was type one instead of type zero, the constant for the steady-state error would be found in a manner similar to the following



Let's check our predictions by looking at a step response plot. This can be done by adding the following two lines of code into the Matlab command window.

```
[numc,denc] = cloop(num,den,-1);
step(numc,denc)
```



As you can see, our predictions were very good. The system has a rise time of about 2 seconds, is overdamped, and has a steady-state error of about 9%. Now we need to choose a controller that will allow us to meet the design criteria. We choose a PI controller because it will yield zero steady state error for a step input. Also, the PI controller has a zero, which we can place. This gives us additional design flexibility to help us meet our criteria. Recall that a PI controller is given by:

```
            K*(s+a)
Gc(s)  =  -------
```

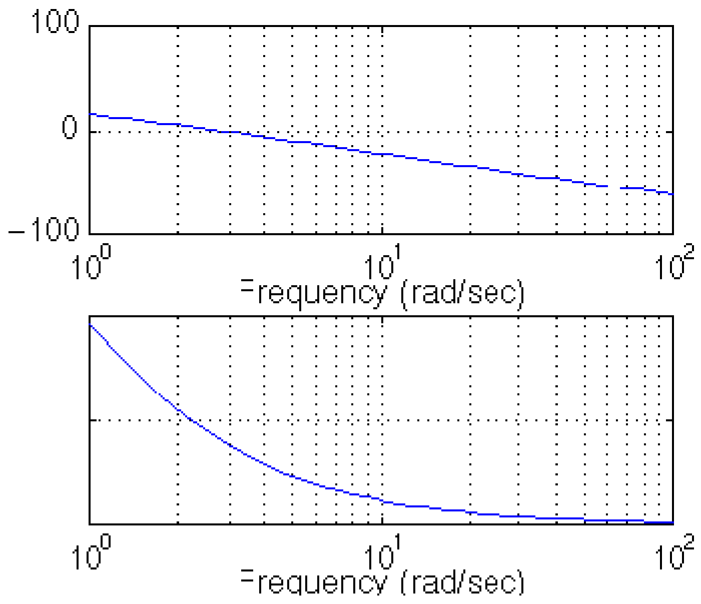
s

The first thing we need to find is the damping ratio corresponding to a percent overshoot of 40%. Plugging in this value into the equation relating overshoot and damping ratio (or consulting a plot of this relation), we find that the damping ratio corresponding to this overshoot is approximately 0.28. Therefore, our phase margin should be approximately 30 degrees. From our Ts*Wbw vs damping ratio plot, we find that Ts*Wbw ~ 21. We must have a bandwidth frequency greater than or equal to 12 if we want our settling time to be less than 1.75 seconds which meets the design specs.

Now that we know our desired phase margin and bandwidth frequency, we can start our design. Remember that we are looking at the open-loop Bode plots. Therefore, our bandwidth frequency will be the frequency corresponding to a gain of approximately -7 dB.

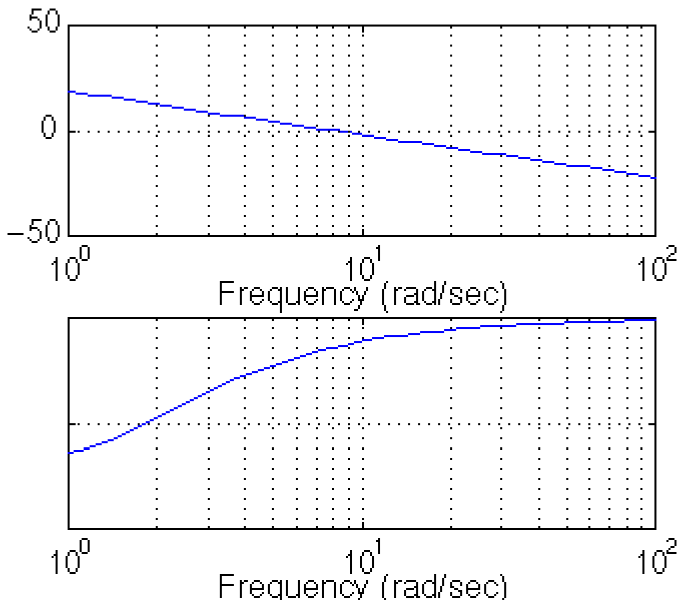
Let's see how the integrator portion of the PI or affects our response. Change your m-file to look like the following (this adds an integral term but no proportional term):

```
num = [10];
den = [1.25, 1];
numPI = [1];
denPI = [1 0];
newnum = conv(num,numPI);
newden = conv(den,denPI);
bode(newnum, newden, logspace(0,2))
```



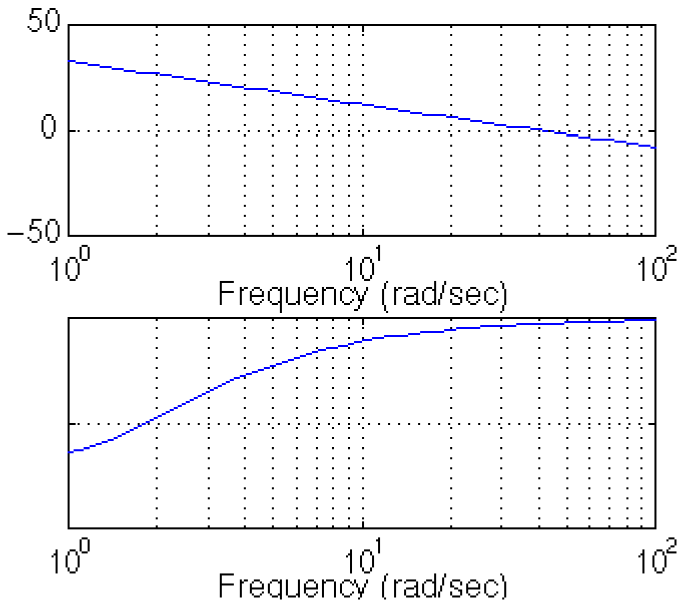
Our phase margin and bandwidth frequency are too small. We will add gain and phase with a zero. Let's place the zero at 1 for now and see what happens. Change your m-file to look like the following:

```
num = [10];
den = [1.25, 1];
numPI = [1 1];
denPI = [1 0];
newnum = conv(num,numPI);
newden = conv(den,denPI);
bode(newnum, newden, logspace(0,2))
```



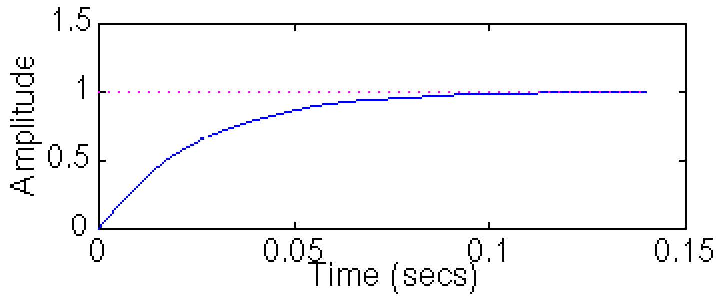
It turns out that the zero at 1 with a unit gain gives us a satisfactory answer. Our phase margin is greater than 60 degrees (even less overshoot than expected) and our bandwidth frequency is approximately 11 rad/s, which will give us a satisfactory response. Although satisfactory, the response is not quite as good as we would like. Therefore, let's try to get a higher bandwidth frequency without changing the phase margin too much. Let's try to increase the gain to 5 and see what happens . This will make the gain shift and the phase will remain the same.

```
num = [10];
den = [1.25, 1];
numPI = 5*[1 1];
denPI = [1 0];
newnum = conv(num,numPI);
newden = conv(den,denPI);
bode(newnum, newden, logspace(0,2))
```



That looks really good. Let's look at our step response and verify our results. Add the following two lines to your m-file:

```
[clnum,clden] =cloop(newnum,newden,-1);
step(clnum,clden)
```



As you can see, our response is better than we had hoped for. However, we are not always quite as lucky and usually have to play around with the gain and the position of the poles and/or zeros in order to achieve our design requirements.

This tutorial is continued on the Nyquist page (the link is after the feedback form).

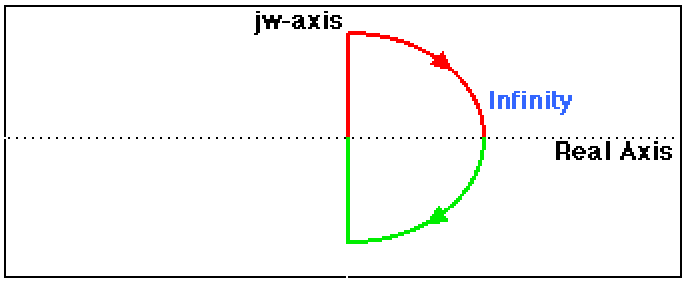
Frequency Response II: The Nyquist Diagram

The Nyquist Diagram

The Nyquist plot allows us to predict the stability and performance of a closed-loop system by observing its open-loop behavior. The Nyquist criterion can be used for design purposes regardless of open-loop stability (remember that the Bode design methods assume that the system is stable in open loop). Therefore, we use this criterion to determine closed-loop stability when the Bode plots display confusing information. The following movie will help you visualize the relationship between the Bode plot and the Nyquist diagram.

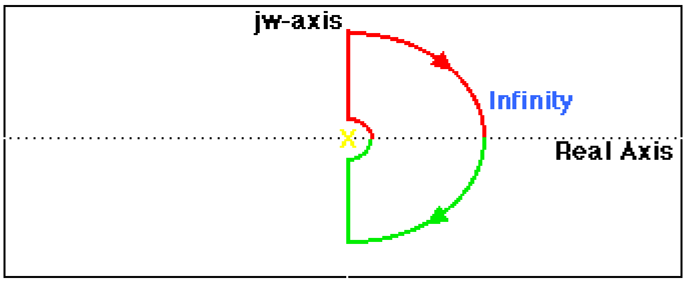
Note: The Matlab nyquist command does not provide an adequate representation for systems that have open-loop poles in the jw-axis. Therefore, we suggest that you copy the nyquist1.m file as a new m-file. This m-file creates more accurate Nyquist plots, since it take into account poles and zeros on the jw-axis.

The Nyquist diagram is basically a plot of G(j* w) where G(s) is the open-loop transfer function and w is a vector of frequencies which encloses the entire right-half plane. In drawing the Nyquist diagram, both positive and negative frequencies (from zero to infinity) are taken into account. We will represent positive frequencies in red and negative frequencies in green. The frequency vector used in plotting the Nyquist diagram usually looks like this (if you can imagine the plot stretching out to infinity):



In order to see how the frequency vector contributes to the Nyquist diagram more clearly, you can view our movie.

However, if we have open-loop poles or zeros on the jw axis, G(s) will not be defined at those points, and we must loop around them when we are plotting the contour. Such a contour would look as follows:



Please note that the contour loops around the pole on the jw axis. As we mentioned before, the Matlab nyquist command does not take poles or zeros on the jw axis into account and therefore produces an incorrect plot. To correct this, please download and use nyquist1.m. If we have a pole on the jw axis, we have to use nyquist1. If there are no poles or zeros on the jw-axis, or if we have pole-zero cancellation, we can use either the nyquist command or nyquist1.m.

The Cauchy criterion

The Cauchy criterion (from complex analysis) states that when taking a closed contour in the complex plane, and mapping it through a complex function G(s), the number of times that the plot of G(s) encircles the origin is equal to the number of zeros of G(s) enclosed by the frequency contour minus the number of poles of G(s) enclosed by the frequency contour. Encirclements of the origin are counted as positive if they are in the same direction as the original closed contour or negative if they are in the opposite direction.

When studying feedback controls, we are not as interested in G(s) as in the closed-loop transfer function:

```
  G(s)
---------
1 + G(s)
```

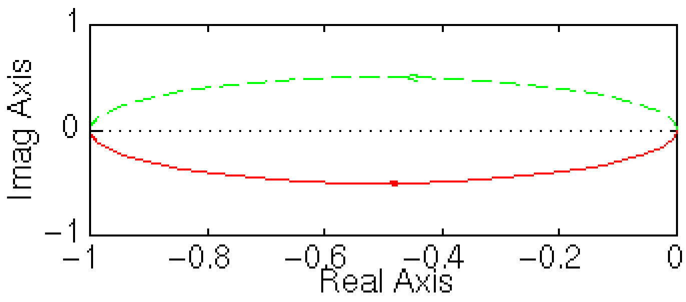
If 1+ G(s) encircles the origin, then G(s) will enclose the point -1. Since we are interested in the closed-loop stability, we want to know if there are any closed-loop poles (zeros of 1 + G(s)) in the right-half plane. More details on how to determine this will come later.

Therefore, the behavior of the Nyquist diagram around the -1 point in the real axis is very important; however, the axis on the standard nyquist diagram might make it hard to see what's happening around this point. To correct this, you can add the lnyquist1.m function to your files. The lnyquist1.m command plots the Nyquist diagram using a logarithmic scale and preserves the characteristics of the -1 point.

To view a simple Nyquist plot using Matlab, we will define the following transfer function and view the Nyquist plot:

$$\frac{0.5}{s - 0.5}$$

```
        nyquist (0.5,[1 -0.5])
```

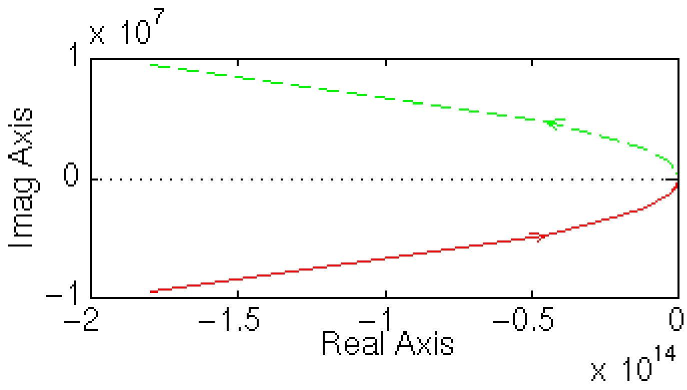


Now we will look at the Nyquist diagram for the following transfer function:

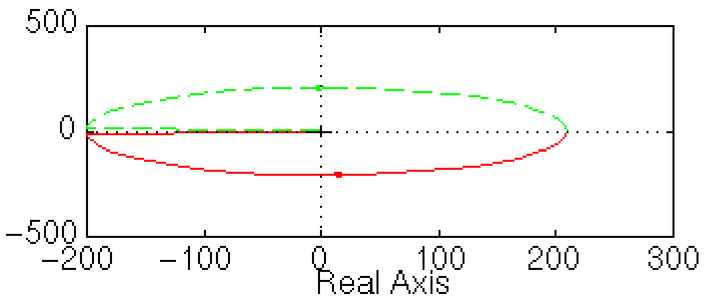
$$\frac{s + 2}{s^2}$$

Note that this function has a pole at the origin. We will see the difference between using the nyquist, nyquist1, and lnyquist1 commands with this particular function.

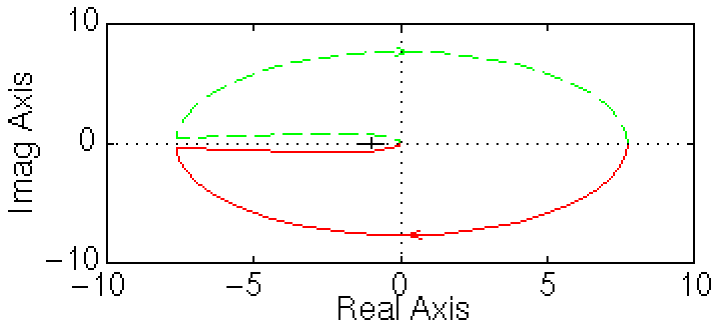
```
        nyquist([1 2], [1 0 0])
```



```
        nyquist1([1 2], [1 0 0])
```



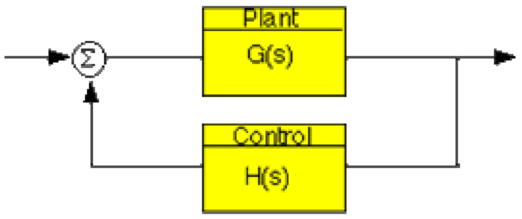
`lnyquist1([1 2], [1 0 0])`



Note that the nyquist plot is not the correct one, the nyquist1 plot is correct, but it's hard to see what happens close to the −1 point, and the lnyquist1 plot is correct and has an appropriate scale.

Closed Loop Stability

Consider the negative feedback system:



Remember from the Cauchy criterion that the number N of times that the plot of G(s)H(s) encircles −1 is equal to the number Z of zeros of 1 + G(s)H(s) enclosed by the frequency contour minus the number P of poles of 1 + G(s)H(s) enclosed by the frequency contour (N = Z − P). Keeping careful track of open- and closed-loop transfer functions, as well as numerators and denominators, you should convince yourself that:

- the zeros of 1 + G(s)H(s) are the poles of the closed-loop transfer function

- the poles of 1 + G(s)H(s) are the poles of the open-loop transfer function.

The Nyquist criterion then states that:

- P = the number of open-loop (unstable) poles of G(s)H(s)
- N = the number of times the Nyquist diagram encircles -1
- clockwise encirclements of -1 count as positive encirclements
- counter-clockwise (or anti-clockwise) encirclements of -1 count as negative encirclements
- Z = the number of right half-plane (positive, real) poles of the closed-loop system

The important equation which relates these three quantities is:

$$Z = P + N$$

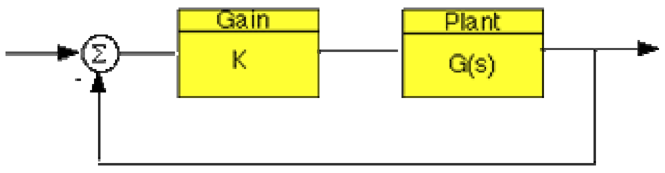
Note: This is only one convention for the Nyquist criterion. Another convention states that a positive N counts the counter-clockwise or anti-clockwise encirclements of -1. The P and Z variables remain the same. In this case the equation becomes Z = P - N. Throughout these tutorials, we will use a positive sign for clockwise encirclements.

It is very important (and somewhat tricky) to learn how to count the number of times that the diagram encircles -1. Therefore, we will go into some detail to help you visualize this. You can view this movie as an example.

Another way of looking at it is to imagine you are standing on top of the -1 point and are following the diagram from beginning to end. Now ask yourself: How many times did I turn my head a full 360 degrees? Again, if the motion was clockwise, N is positive, and if the motion is anti-clockwise, N is negative.

Knowing the number of right-half plane (unstable) poles in open loop (P), and the number of encirclements of -1 made by the Nyquist diagram (N), we can determine the closed-loop stability of the system. If Z = P + N is a positive, nonzero number, the closed-loop system is unstable.

We can also use the Nyquist diagram to find the range of gains for a closed-loop unity feedback system to be stable. The system we will test looks like this:



where G(s) is :

```
s^2 + 10 s + 24
---------------
 s^2 - 8 s + 15
```

This system has a gain K which can be varied in order to modify the response of the closed-loop system. However, we will see that we can only vary this gain within certain limits, since we have to make sure that our closed-loop system will be stable. This is what we will be looking for: the range of gains that will make this system stable in the closed loop.

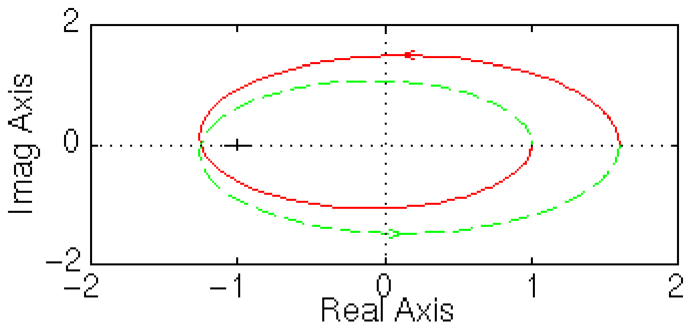
The first thing we need to do is find the number of positive real poles in our open-loop transfer function:

```
          roots([1 -8 15])

                ans =
                        5
                        3
```

The poles of the open-loop transfer function are both positive. Therefore, we need two anti-clockwise (N = -2) encirclements of the Nyquist diagram in order to have a stable closed-loop system (Z = P + N). If the number of encirclements is less than two or the encirclements are not anti-clockwise, our system will be unstable.

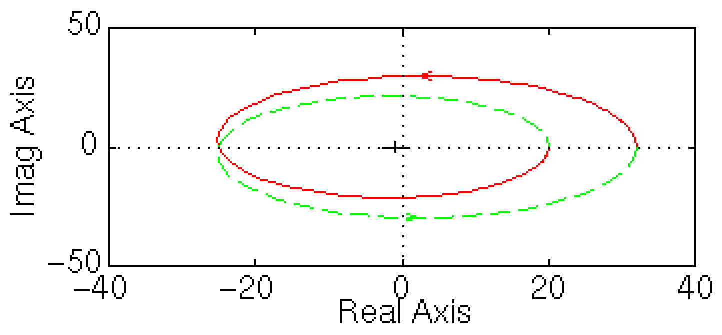
Let's look at our Nyquist diagram for a gain of 1:

```
nyquist([ 1 10 24], [ 1 -8 15])
```



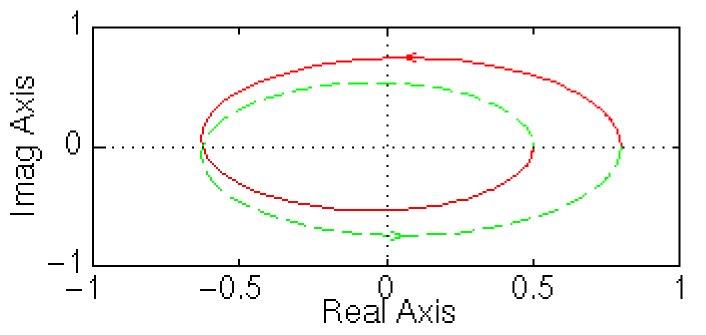
There are two anti-clockwise encirclements of -1. Therefore, the system is stable for a gain of 1. Now we will see how the system behaves if we increase the gain to 20:

```
nyquist(20*[ 1 10 24], [ 1 -8 15])
```



The diagram expanded. Therefore, we know that the system will be stable no matter how much we increase the gain. However, if we decrease the gain, the diagram will contract and the system might become unstable. Let's see what happens for a gain of 0.5:

```
nyquist(0.5*[ 1 10 24], [ 1 -8 15])
```



The system is now unstable. By trial and error we find that this system will become unstable for gains less than 0.80. We can verify our answers by zooming in on the Nyquist plots as well as by looking at the closed-loop steps responses for gains of 0.79, 0.80, and 0.81.

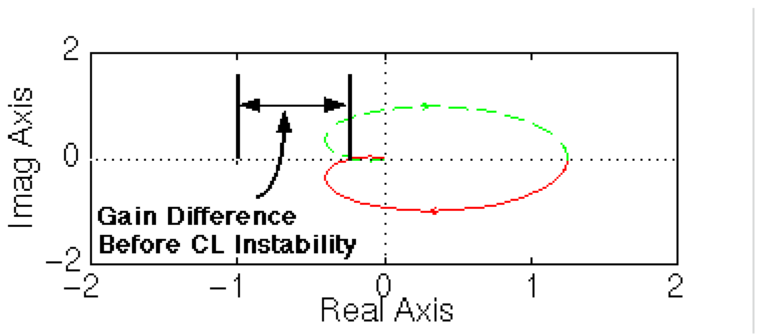
If you are having trouble counting the Nyquist encirclements, we suggest you try using nyquist1. The number of anti-clockwise encirclements will be displayed on your screen (remember that this number actually represents negative N, i.e. if nyquist1 shows 2, N = -2) as well as the number of open and closed-loop positive real poles.

Gain Margin

We already defined the gain margin as the change in open-loop gain expressed in decibels (dB), required at 180 degrees of phase shift to make the system unstable. Now we are going to find out where this comes from. First of all, let's say that we have a system that is stable if there are no Nyquist encirclements of -1, such as :

50 ---------------------- s^3 + 9 s^2 + 30 s + 40

Looking at the roots, we find that we have no open loop poles in the right half plane and therefore no closed-loop poles in the right half plane if there are no Nyquist encirclements of -1. Now, how much can we vary the gain before this system becomes unstable in closed loop? Let's look at the following figure:



The open-loop system represented by this plot will become unstable in closed loop if the gain is increased past a certain boundary. The negative real axis area between -1/a (defined as the point where the 180 degree phase shift occurs...that is, where the diagram crosses the real axis) and -1 represents the amount of increase in gain that can be tolerated before closed-loop instability.

If we think about it, we realize that if the gain is equal to a, the diagram will touch the -1 point:

G(jw) = -1/a = a*G(jw) = a* -1/a => a*G(jw) = -1

Therefore, we say that the gain margin is 'a' units. However, we mentioned before that the gain margin is usually measured in decibels. Hence, the gain margin is :

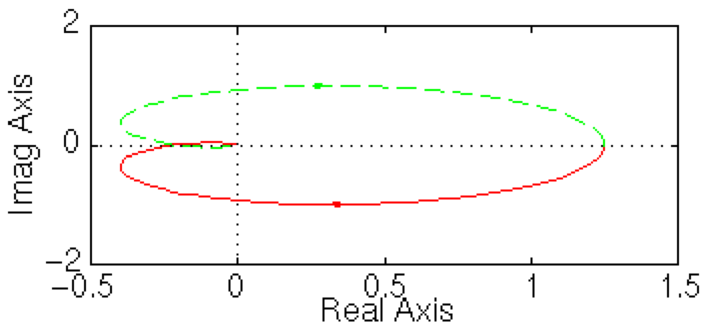
GM = 20*log10(a) [dB]

We will now find the gain margin of the stable, open-loop transfer function we viewed before. Recall that the function is:

```
50 ----------------------- s^3 + 9 s^2 + 30 s + 40
```

and that the Nyquist diagram can be viewed by typing:

```
nyquist (50, [1 9 30 40 ])
```



As we discussed before, all that we need to do to find the gain margin is

find 'a', as defined in the preceding figure. To do this, we need to find

the point where there is exactly 180 degrees of phase shift. This means

that the transfer function at this point is real (has no imaginary part).

The numerator is already real, so we just need to look at the denominator.

When s = j*w, the only terms in the denominator that will have imaginary parts are those which are odd powers of s. Therefore, for G(j*w) to be real, we must have:

$$-j\ w^3 + 30\ j\ w = 0$$

which means w=0 (this is the rightmost point in the Nyquist diagram) or w=sqrt(30). We can then find the value of G(j*w) at this point using polyval:

```
polyval(50,j*w)/polyval([1 9 30 40],j*w)
```

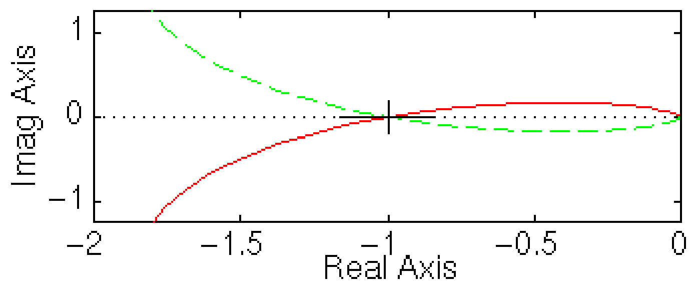
Our answer is: -0.2174 + 0i. The imaginary part is zero, so we know that our answer is correct. We can also verify by looking at the Nyquist plot again. The real part also makes sense. Now we can
proceed to find the gain margin.

We found that the 180 degrees phase shift occurs at -0.2174 + 0i. This point was previously defined as -1/a. Therefore, we now have 'a', which is the gain margin. However, we need to express the gain margin in decibels,

$$-1/a = -0.2174 \Rightarrow a = 4.6 \Rightarrow GM = 20*\log10(\ 4.6) = 13.26\ \text{dB}$$

We now have our gain margin. Let's see how accurate it is by using a gain of a = 4.6 and zooming in on the Nyquist plot:

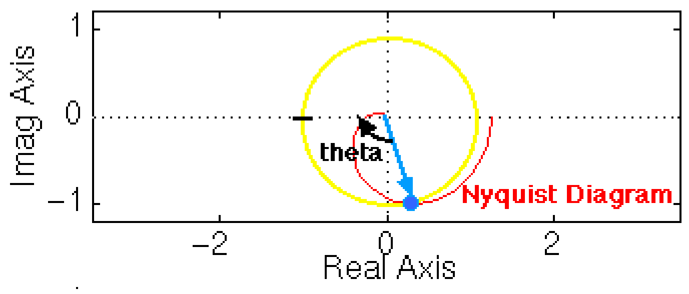
```
a = 4.6
nyquist(a*50,[1 9 30 40])
```



The plot appears to go right through the −1 point. We will now verify the accuracy of our results by viewing the zoomed Nyquist diagrams and step responses for gains of 4.5, 4.6, and 4.7.

Phase Margin

We have already discussed the importance of the phase margin. Therefore, we will only talk about where this concept comes from. We have defined the phase margin as the change in open-loop phase shift required at unity gain to make a closed-loop system unstable. Let's look at the following graphical definition of this concept to get a better idea of what we are talking about.



Let's analyze the previous plot and think about what is happening. From our previous example we know that this particular system will be unstable in closed loop if the Nyquist diagram encircles the -1 point. However, we

must also realize that if the diagram is shifted by theta degrees, it will then touch the -1 point at the negative real axis, making the system marginally stable in closed loop. Therefore, the angle required to make this system marginally stable in closed loop is called the phase margin (measured in degrees). In order to find the point we measure this angle from, we draw a circle with radius of 1, find the point in the Nyquist diagram with a magnitude of 1 (gain of zero dB), and measure the phase shift needed for this point to be at an angle of 180 deg.

三. 实验步骤

选择如下示例，按步骤进行试验：

Example: Solution to the Cruise Control Problem Using Frequency Response

The open-loop transfer function for this problem is :

$$\frac{Y(s)}{U(s)} = \frac{1}{ms + b}$$

- m=1000
- b=50
- U(s)=10
- Y(s)=Velocity output

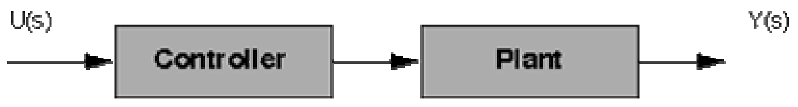
The design criteria are:

Rise time < 5 sec
Overshoot < 10%
Steady state error < 2%

To see the original problem set, see the Cruise Control Modeling page.

Bode plot and open-loop response

The first step in solving this problem using frequency response is to determine what open-loop transfer function to use. Just like for the Root-Locus design method, we will only use a proportional controller to solve the problem. The block diagram and the open-loop transfer function are shown below.

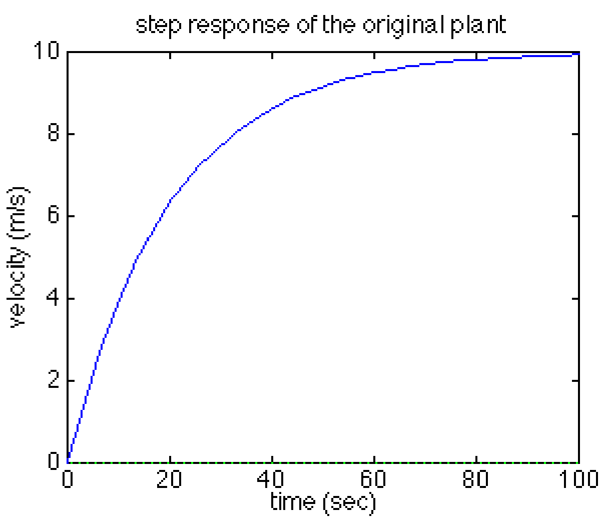


$$\frac{Y(s)}{U(s)} = \frac{K_p}{ms + b}$$

In order to use a Bode plot, the open-loop response must be stable. Let Kp equals 1 for now and see how the open-loop response looks like. Create an new m-file and enter the following commands.

```
m = 1000;
b = 50;
u = 500;
Kp=1;

numo=[Kp];
deno=[m b];

step (u*numo,deno)
```

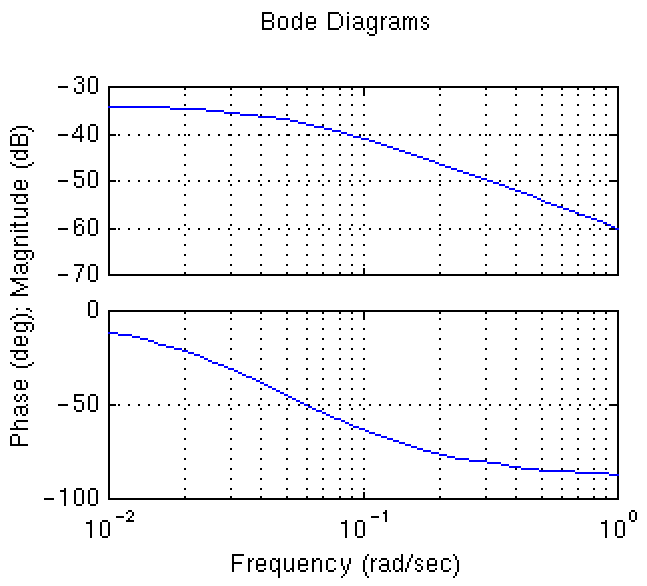
Running this m-file in the Matlab command window should give you the following plot.



As you can see, the open-loop system is stable; thus, we can go ahead and generate the Bode plot. Change the above m-file by deleting step command and add in the following command.

```
bode(numo,deno)
```

Running this new m-file should give you the following Bode plot.



Proportional controller

Let's see what system characteristics we can determine from the above Bode plot. Recall from the Root-Locus Tutorial, the bandwidth frequency (BW) (the frequency at the gain M(dB)=-6~-7.5dB) is roughly equals to the natural frequency (Wn). Using the equation,

$$Tr = \frac{1.8}{Wn} = \frac{1.8}{BW}$$

the rise time (Tr) for our system can be determined to be extremely long since the gain shown above do not reach -6~-7.5dB. Moreover, we see from the Root-Locus Tutorial that the damping ratio is roughly equals to the phase margin (in degrees) divided by 100.

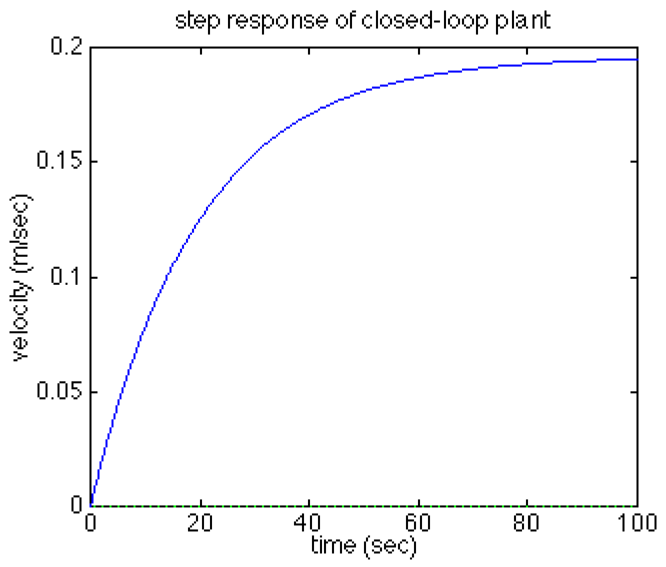
$$\xi = \frac{PM(deg)}{100}$$

Since our phase margin is approximately 155 degrees, the damping ratio will be 1.55. Thus, we know that the system is overdamped (since the damping ratio is greater than 1). Finally, the steady-state error can be found from the following equation:

$$ss-error = \frac{1}{1+(magnitude)} \times 100\%$$

$$10^{(M\,dB)/20} = Magnitude$$

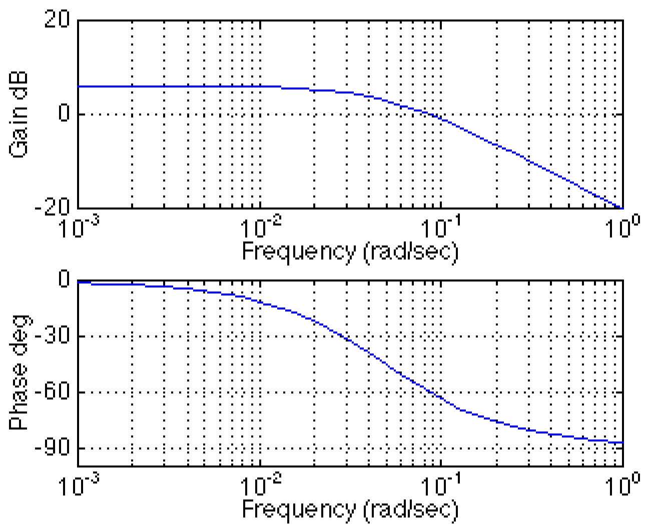
For our system, since the low frequency gain M(dB) is approximately -35dB, the steady-state error should be 98%. We can confirm these by generating a closed-loop step response.



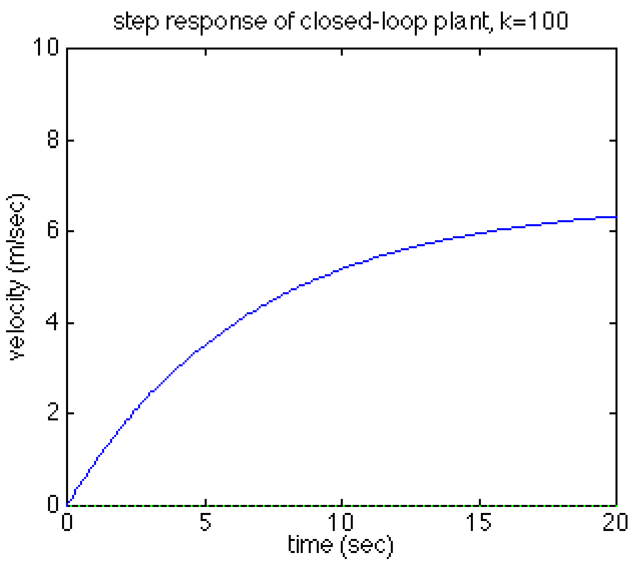
In terms of the Bode plot, if we can shift the gain upward so that both the bandwidth frequency and the low frequency gain increase, then both the rise time and the steady-state error will improve. We can do that by increasing the proportional gain (Kp). Let's increase the proportional gain (Kp) to ,say, 100 and see what happens. Change the m-file to the following.

```
m = 1000;
b = 50;
u = 10;
Kp=100;

numo=[Kp];
deno=[m b];

bode(numo,deno)
```

Running this m-file in the Matlab command window should give you the following Bode plot.

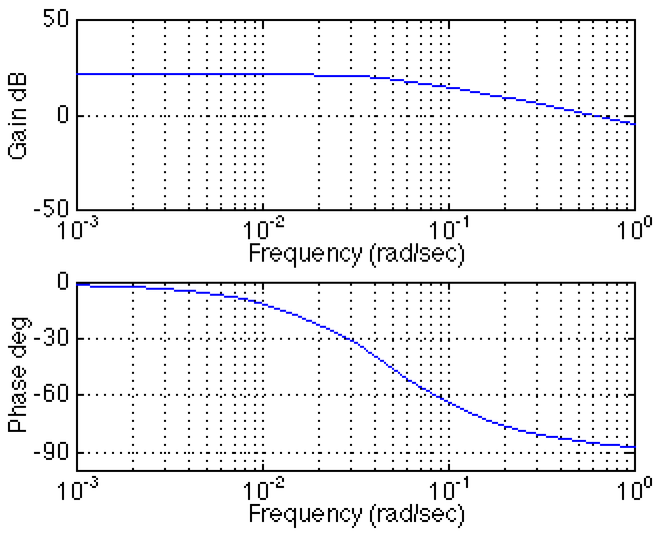


Now, the low frequency gain is about 6dB (magnitude 2) which predicts the steady-state error of 33%. The bandwidth frequency is about 0.1 rad/sec so that the rise time should be around 18 seconds. Let's take a look at the closed-loop step response and confirm these.

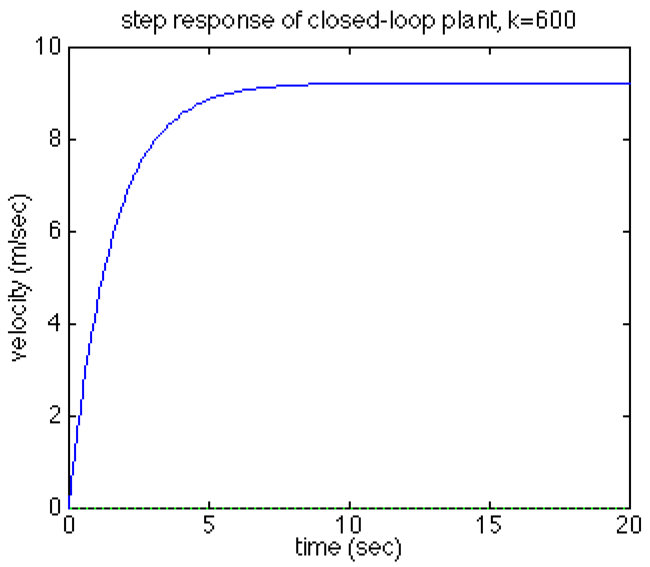


As we predicted, both the steady-state error and the rise time have improved. Let's increase the proportional gain even higher and see what happens. Change the m-file to the following and rerun it. You should get the following Bode plot.

```
m = 1000;
b = 50;
u = 10;
Kp=100;

numo=[Kp/b];
deno=[m/b 1];

bode(numo,deno)
```



Now, the low frequency gain is approximately 20dB (magnitude 10) that predicts the steady-state error of 9%, and the bandwidth frequency is around 0.6 that predicts the rise time of 3 sec **(within the desired value)**. Thus, both the steady-state error and the rise time should have been improved. Again, let's confirm these by generating the closed-loop step response.



If you noticed, the steady-state error will eventually reach the desired value by increasing the proportional gain even higher. However, by the time the steady-state error reaches the desired value, the rise time becomes too fast (unrealistic for the real physical system). Thus, let's leave the Kp as it is and implement a lag controller to handle the steady-state error problem.

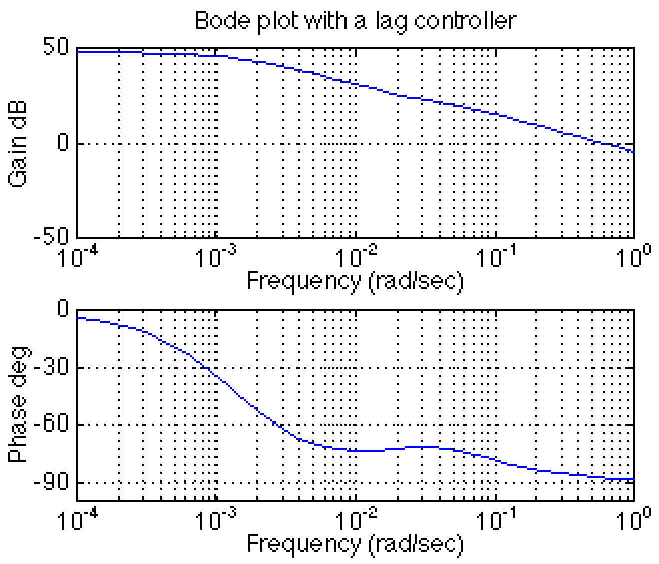
Lag controller

If you take a look at the "Lag or Phase-Lag Compensator using Frequency Response" section of the Lead and Lag Compensator page, the lag controller adds gain at the low freqencies while keeping the bandwidth frequency at the same place. This is actually what we need: Larger low frequency gain to reduce the steady-state error and keep the same bandwidth frequency to maintain the desired rise time. The transfer function of the lag controller is shown below.

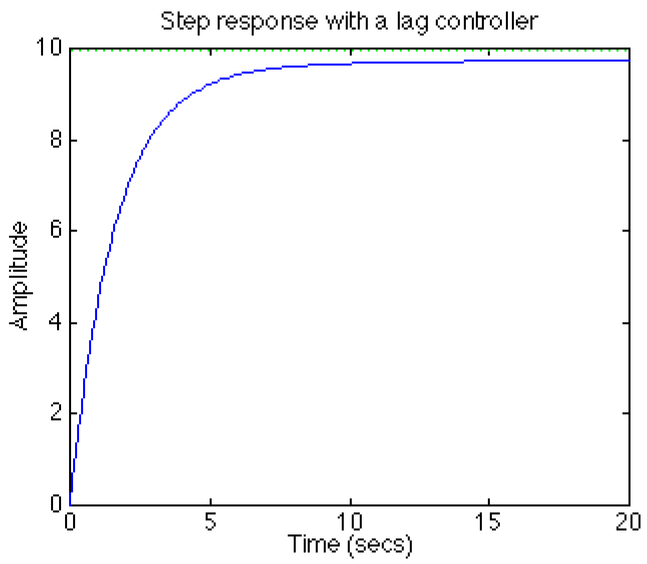
$$G(s)=\frac{1}{a}\left(\frac{1+aTs}{1+Ts}\right) \quad (a<1)$$

Now, we need to choose a value for **a** and **T**. Recall from the "Lag or Phase-Lag Compensator using Frequency Response" page, the steady-state error will decrease by a factor of **a**. The value **T** should be chosen so that two corner frequencies will not be placed close together because transient response gets worse. So let **a** equals 0.05 and **T** equals 700 and see what happens. Copy the following m-file and run it in the Matlab command window. You should see the following Bode plot.

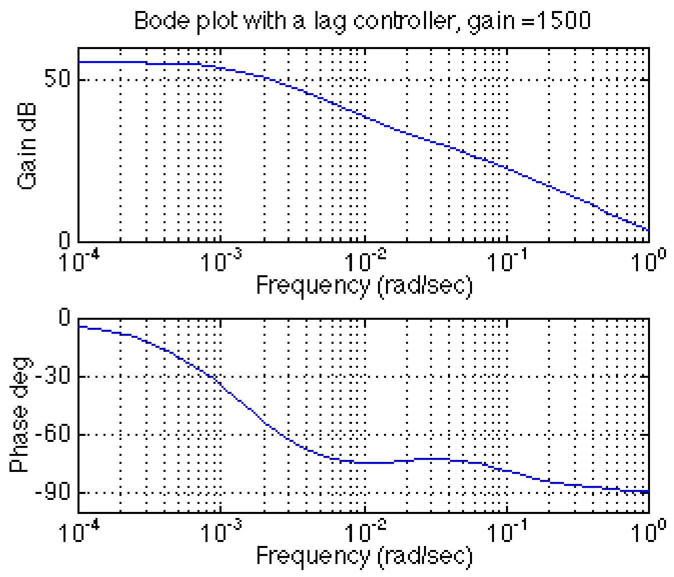
```
m = 1000;
b = 50;
u = 10;
Kp=600;

numo=[Kp/b];
deno=[m/b 1];

a = 0.05;
T=700;
numlag = [a*T 1];
denlag = a*[T 1];
newnum = conv(numo,numlag);
newden = conv(deno,denlag);

bode(newnum,newden)

figure
[numc,denc]=cloop(newnum,newden);
step (u*numc,denc)
```

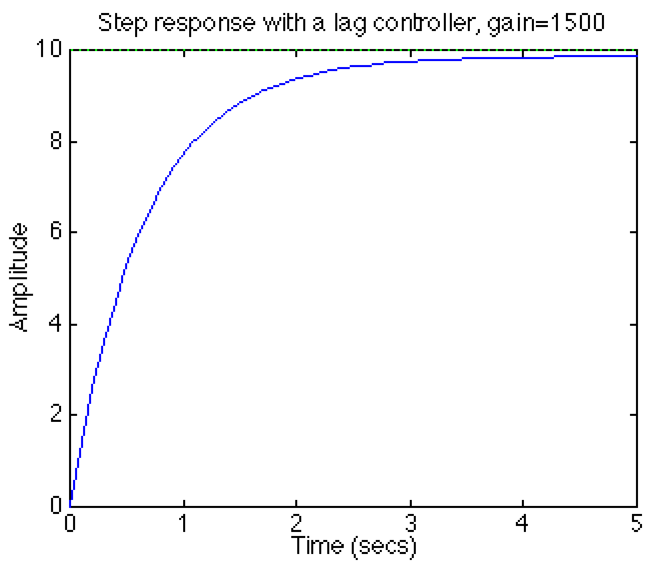


Since the low frequency gain has increased while the bandwidth frequency stayed the same, the steady-state error should be reduced and the rise time should stay the same. Let's confirm this by generating a closed-loop step response.



It may be hard to see, but there should be a green, dotted line across just below 10. This line shows the steady-state value of the step, and we can see that the steady-state error has been met. However, the settling time is too long. To fix this, raise the proportional gain to Kp=1500. This gain was chosen from trial-and-error that will not be described here in the interest of length. With this change made, the following Bode and step response plots can be generated.





As you can see, the overshoot is in fact zero, the steady state error is close to zero, the rise time is about 2 seconds, and the settling time is less than 3.5 seconds. The system has now met all of the design requirements. No more iteration is needed.

四. 实验报告

1. 综述频率响应法控制的理论原理；

2. 画出示例程序中频率响应法控制的曲线；

3. 选择其它的示例实现利用频率响应控制。