实验四　利用根轨迹法控制的MATLAB实现

一.　实验目的

1.　熟悉并掌握MATLAB的工作环境。

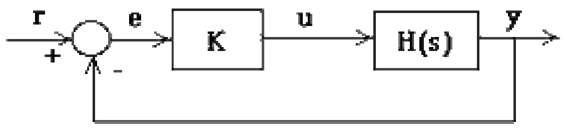
2.　了解根轨迹控制技术的基本理论。

3.　在MATLAB工作环境下，选择适当的例子，实现根轨迹法控制，讨论控制效果。

二.　实验内容

Closed-Loop Poles

The root locus of an (open-loop) transfer function H(s) is a plot of the locations (locus) of all possible closed loop poles with proportional gain k and unity feedback:



The closed-loop transfer function is:

$$\frac{Y(s)}{R(s)} = \frac{KH(s)}{1+KH(s)}$$

and thus the poles of the closed loop system are values of s such that 1 + K H(s) = 0.

If we write H(s) = b(s)/a(s), then this equation has the form:

$$a(s) + K\,b(s) = 0$$
$$\frac{a(s)}{K} + b(s) = 0$$

Let n = order of a(s) and m = order of b(s) [the order of a polynomial is the highest power of s that appears in it].

We will consider all positive values of k. In the limit as k -> 0, the poles of the closed-loop system are a(s) = 0 or the poles of H(s). In the limit as k -> infinity, the poles of the closed-loop system are b(s) = 0 or the zeros of H(s).

No matter what we pick k to be, **the closed-loop system must always have n poles**, where n is the number of poles of H(s). **The root locus must have n branches**, each branch starts at a pole of H(s) and goes to a zero of H(s). If H(s) has more poles than zeros (as is often the case), m < n and we say that H(s) has **zeros at infinity**. In this case, the limit of H(s) as s -> infinity is zero. The number of zeros at infinity is n-m, the number of poles minus the number of zeros, and is the number of branches of the root locus that go to infinity (asymptotes).

Since the root locus is actually the locations of all possible closed loop poles, from the root locus we can select a gain such that our closed-loop system will perform the way we want. If any of the selected poles are on the right half plane, the closed-loop system will be unstable. The poles that are closest to the imaginary axis have the greatest influence on the closed-loop response, so even though the system has three or four poles, it may still act like a second or even first order system depending on the location(s) of the dominant pole(s).

Plotting the root locus of a transfer function

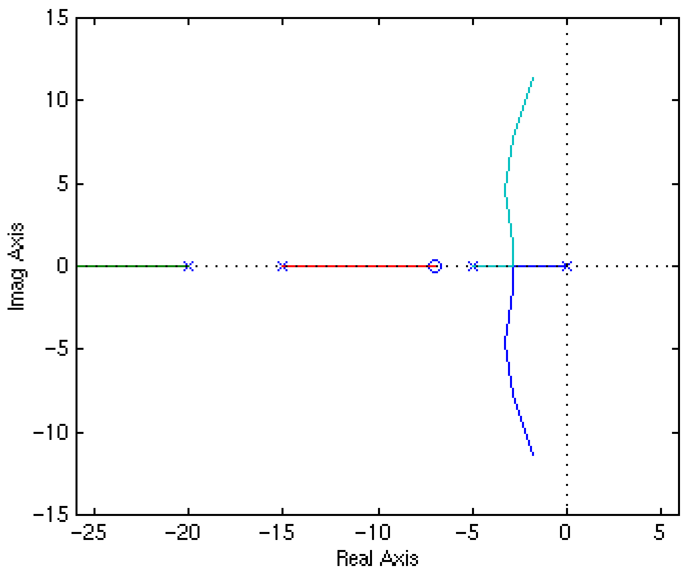
Consider an open loop system which has a transfer function of

$$H(s) = \frac{Y(s)}{U(s)} = \frac{s+7}{s(s+5)(s+15)(s+20)}$$

How do we design a feed-back controller for the system by using the root locus method? Say our design criteria are 5% overshoot and 1 second rise time. Make a Matlab file called rl.m. Enter the transfer function, and the command to plot the root locus:

```
num=[1 7];
```

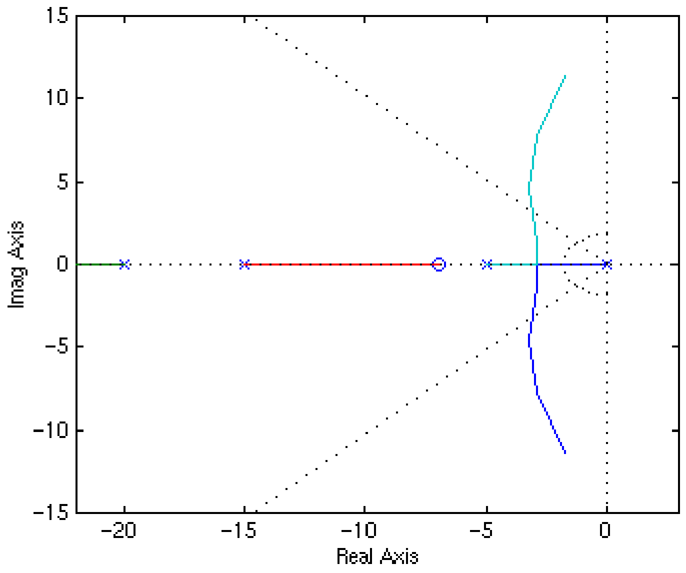
```
den=conv(conv([1 0],[1 5]),conv([1 15],[1 20]));
rlocus(num,den)
axis([-22 3 -15 15])
```



Choosing a value of K from the root locus

The plot above shows all possible closed-loop pole locations for a pure proportional controller. Obviously not all of those closed-loop poles will satisfy our design criteria. To determine what part of the locus is acceptable, we can use the command sgrid(Zeta,Wn) to plot lines of constant damping ratio and natural frequency. Its two arguments are the damping ratio (Zeta) and natural frequency (Wn) [these may be vectors if you want to look at a range of acceptable values]. In our problem, we need an overshoot less than 5% (which means a damping ratio Zeta of greater than 0.7) and a rise time of 1 second (which means a natural frequency Wn greater than 1.8). Enter in the Matlab command window:

```
zeta=0.7;
Wn=1.8;
sgrid(zeta, Wn)
```



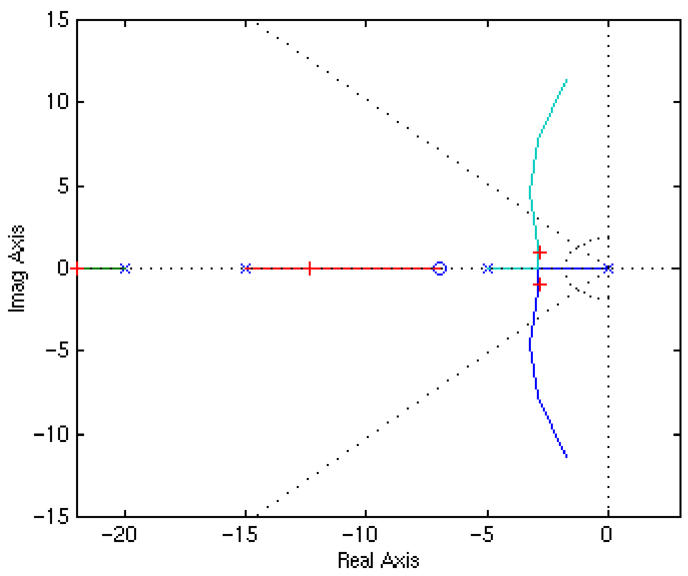
On the plot above, the two white dotted lines at about a 45 degree angle indicate pole locations with Zeta = 0.7; in between these lines, the poles will have Zeta > 0.7 and outside of the lines Zeta < 0.7. The semicircle indicates pole locations with a natural frequency Wn = 1.8; inside the circle, Wn < 1.8 and outside the circle Wn > 1.8.

Going back to our problem, to make the overshoot less than 5%, the poles have to be in between the two white dotted lines, and to make the rise time shorter than 1 second, the poles have to be outside of the white dotted semicircle. So now we know only the part of the locus outside of the semicircle and in between the two lines are acceptable. All the poles in this location are in the left-half plane, so the closed-loop system will be stable.

From the plot above we see that there is part of the root locus inside the desired region. So in this case we need only a proportional controller to move the poles to the desired region. You can use rlocfind command in Matlab to choose the desired poles on the locus:

```
    [kd,poles] = rlocfind(num,den)
```

Click on the plot the point where you want the closed-loop pole to be. You may want to select the points indicated in the plot below to satisfy the design criteria.



Note that since the root locus may has more than one branch, when you select a pole, you may want to find out where the other pole (poles) are. Remember they will affect the response too. From the plot above we see that all the poles selected (all the white "+") are at reasonable positions. We can go ahead and use the chosen kd as our proportional controller.

Closed-loop response

In order to find the step response, you need to know the closed-loop transfer function. You could compute this using the rules of block diagrams, or let Matlab do it for you:

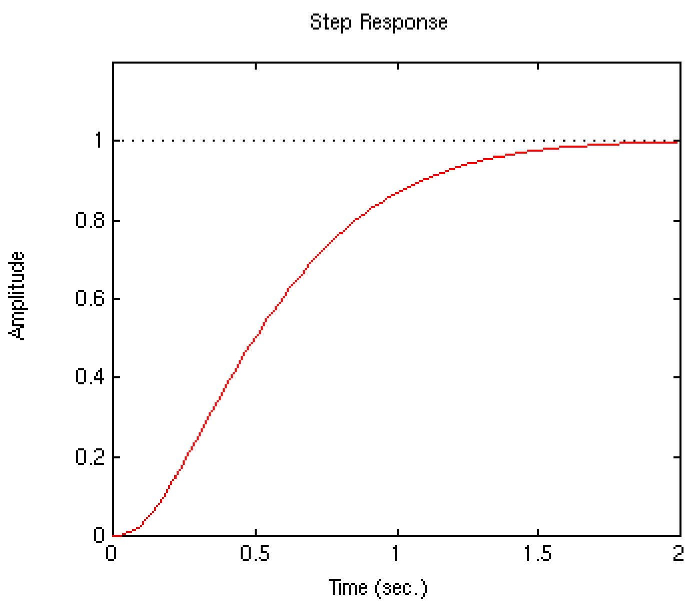
```
    [numCL, denCL] = cloop((kd)*num, den)
```

The two arguments to the function cloop are the numerator and denominator of the open-loop system. You need to include the proportional gain that you have chosen. Unity feedback is assumed.

> If you have a non-unity feedback situation, look at the help file for the Matlab function feedback, which can find the closed-loop transfer function with a gain in the feedback loop.

Check out the step response of your closed-loop system:

```
    step(numCL,denCL)
```



As we expected, this response has an overshoot less than 5% and a rise time less than 1 second.

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 Root Locus Method

Proportional Controller

Lag controller

The open-loop transfer function for this problem is:

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

where

- 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 setup, refer to Cruise Control Modeling page.

Proportional controller

Recall from the Root-Locus Tutorial page, the root-locus plot shows the locations of all possible closed-loop poles when a **single gain** is varied from zero to infinity. Thus, only a proportional controller (Kp) will be considered to solve this problem. Then, the closed-loop transfer function becomes:

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

Also, from the Root-Locus Tutorial, we know that the Matlab command called sgrid should be used to find an acceptable region of the root-locus plot. To use the sgrid, both the damping ratio (zeta) and the natural frequency (Wn) need to be determined first. The following two equations will be used to find the damping ratio and the natural frequency:

$$\omega_n \geq \frac{1.8}{Tr}$$

$$\xi \geq \sqrt{\frac{\left(\ln Mp/\pi\right)^2}{1 + \left(\ln Mp/\pi\right)^2}}$$

where

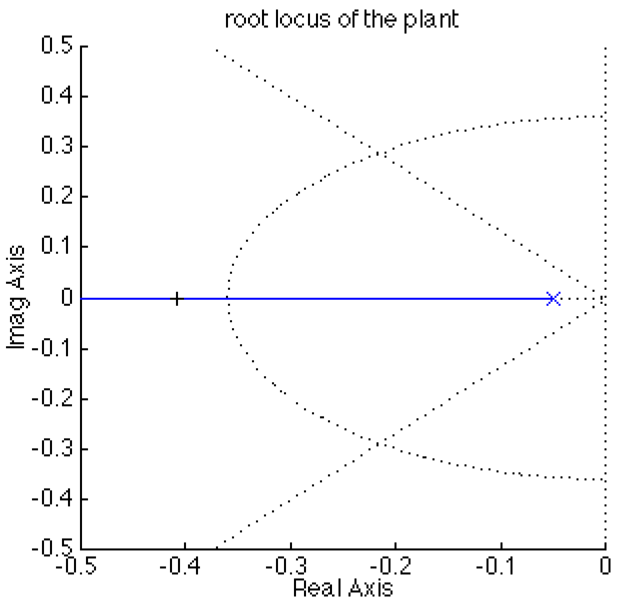
Wn=Natural frequency
zeta=Damping ratio
Tr=Rise time
Mp=Maximum overshoot

One of our design criteria is to have a rise time of less than 5 seconds. From the first equation, we see that the natural frequency must be greater than 0.36. Also using the second equation, we see that the damping ratio must be greater than 0.6, since the maximum overshoot must be less than 10%.

Now, we are ready to generate a root-locus plot and use the sgrid to find an acceptable region on the root-locus. Create an new m-file and enter the following commands.

```
hold off;
m = 1000;
b = 50;
u = 10;
numo=[1];
deno=[m b];

figure
hold;
axis([-0.6 0 -0.6 0.6]);
rlocus (numo,deno)
sgrid(0.6, 0.36)
[Kp, poles]=rlocfind(numo,deno)

figure
hold;
numc=[Kp];
denc=[m (b+Kp)];
t=0:0.1:20;
step (u*numc,denc,t)
axis ([0 20 0 10])
```

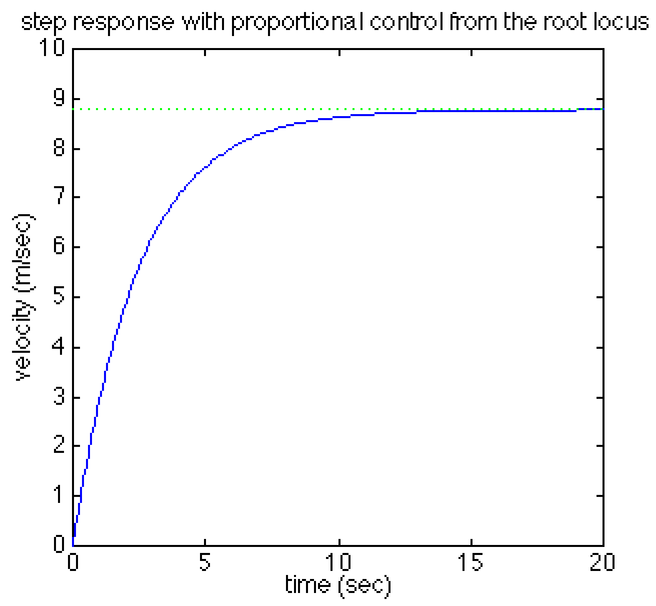
Running this m-file should give you the following root-locus plot.



The two dotted lines in an angle indicate the locations of constant damping ratio (zeta=0.6); the damping ratio is greater than 0.6 in between these lines and less than 0.6 outside the lines. The semi-ellipse indicates the locations of constant natural frequency (Wn=0.36); the natural frequency is greater than 0.36 outside the semi-ellipse, and smaller than 0.36 inside.

If you look at the Matlab command window, you should see a prompt asking you to pick a point on the root-locus plot. Since you want to pick a point in between dotted lines (zeta>0.6) and outside the semi-ellipse (Wn>0.36), click on the real axis just outside the semi-ellipse (around -0.4).

You should see the gain value (Kp) and pole locations in the Matlab command window. Also you should see the closed-loop step response similar to the one shown below.



With the specified gain Kp (the one you just picked), the rise time and the overshoot criteria have been met; however, the steady-state error of more than 10% remained.

Lag controller

To reduce the steady-state error, a lag controller will be added to the system. The transfer function of the lag controller is:

$$G_l(s) = \frac{s + Z_o}{s + P_o}$$

The open-loop transfer function (not including Kp) now becomes:

$$\frac{Y(s)}{U(s)} = \frac{s + Z_o}{ms^2 + (b + mP_o)s + bP_o}$$

Finally, the closed-loop transfer function becomes:

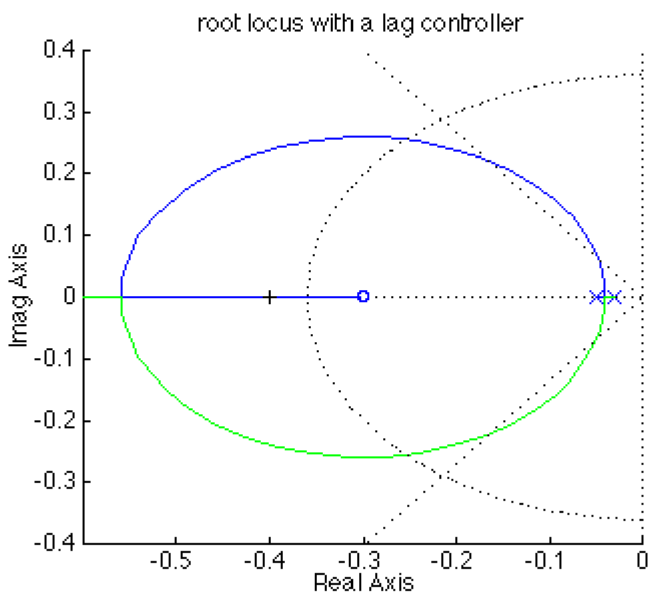
$$\frac{Y(s)}{U(s)} = \frac{K_p s + K_p Z_o}{ms^2 + (b + mP_o + K_p)s + (bP_o + K_p Z_o)}$$

If you read the "Lag or Phase-Lag Compensator using Root-Locus" section in Lead and Lag Compensator page, the pole and the zero of a lag controller need to be placed close together. Also, it states that the steady-state error will be reduce by a factor of Zo/Po. For these reasons, let Zo equals -0.3 and Po equals -0.03.

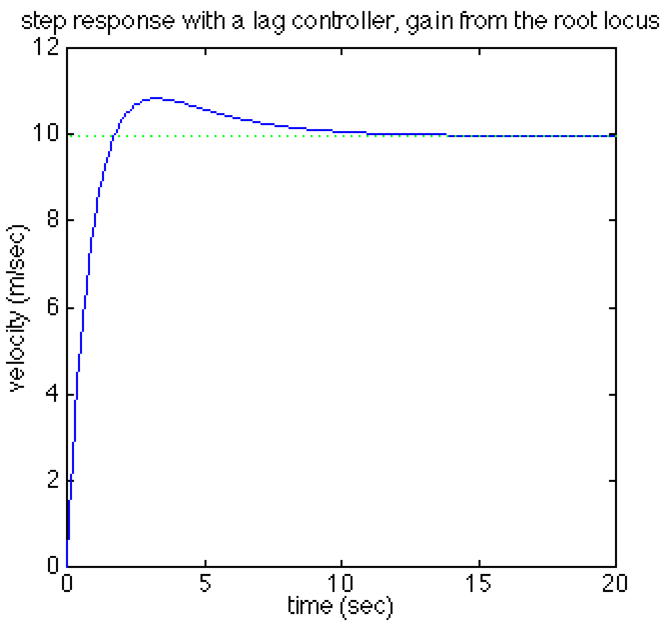
Create an new m-file, and enter the following commands.

```
hold off;
m = 1000;
b = 50;
u = 10;
Zo=0.3;
Po=0.03;
numo=[1 Zo];
deno=[m b+m*Po b*Po];

figure
hold;
axis ([-0.6 0 -0.4 0.4])
rlocus(numo,deno)
sgrid(0.6,0.36)
[Kp, poles]=rlocfind(numo,deno)

figure
t=0:0.1:20;
numc=[Kp Kp*Zo];
denc=[m b+m*Po+Kp b*Po+Kp*Zo];
axis ([0 20 0 12])
step (u*numc,denc,t)
```

Running this m-file should give you the root-locus plot similar to the following:



In the Matlab command window, you should see the prompt asking you to select a point on the root-locus plot. Once again, click on the real axis around -0.4. You should have the following response.



As you can see, the steady-state error has been reduced to near zero. Slight overshoot is a result of the zero added in the lag controller.

Now all of the design criteria have been met and no further iterations will be needed.

四．实验报告

1．综述根轨迹控制的理论原理；

2．画出示例程序中根轨迹控制结构图，并简述控制效果；

3．选择其它的示例实现根轨迹控制。