MATLAB MODULE 6

Stability Analysis on Root Locus Plots

Stability Analysis in State Space

There are several useful MATLAB commands related to analysis and design in state space. Commands useful for stability analysis in state space are:

ss; step; initial; eig; tf; minreal

The MATLAB program below forms the matrices **A,b,c** and *d* for the system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t)$$
$$y(t) = \mathbf{c}\mathbf{x}(t) + du(t)$$

```
>> A = [0 1 0; 0 0 1; 0 -2 -3];

>> b = [0; 0; 1];

>> c = [1 0 0 ];

>> d = 0;
```

The output generated in the MATLAB Command Window by executing these commands is

```
A =

0         1          0
0         0          1
0        -2         -3


b =

0
0
1
```

```
c =

1          0            0


d =

0
```

The command **ss (A,b,c,d)** creates the state space model; let us call it **stmod** . The MATLAB dialogue is given below.

```
>> stmod = ss(A,b,c,d)

a =

     x1  x2 x3

x1   0   1    0

x2   0   0    1

x3   0  -2   -3


b =

      u1

x1    0

x2    0

x3    1


c =

     x1 x2 x3

y1   1   0    0


d =
      u 1

y1    0

Continuous-time model.
```

The system **stmod** can then be used as an argument for many of the MATLAB commands we have already discussed. For instance, the command **step(stmod)**generates the plot of unit-step response *y*(*t*) (with zero initial conditions).

The command **initial(stmod, x0)** creates the free response (with input *u*(*t*) = 0) to initial condition vector **x**0**.**

Typing **eig(stmod)** in the command window produces the dialogue

```
>> eig(stmod)

ans =

  0
-1
-2
```

Thus the eigenvalues of the plant matrix **A** are

$$\lambda = 0, \lambda = -1, \text{ and } \lambda = -2.$$

MATLAB has the means to perform model conversions. Given the state model stmod, the syntax for conversion to transfer function model is

```
tfmod = tf(stmod)
```

Common pole-zero factors of the transfer function model must be cancelled before we can claim that we have the transfer function representation of the system. To assist us in pole-zero cancellation, MATLAB has **mineral (tfmod)** function.

Asymptotic stability is given by the command **eig(A)**. If conversion of the state model to transfer function gives a model with no pole-zero cancellations, the system is both controllable and observable; asymptotic stability implies BIBO stability and *vice versa*. Stability analysis can be carried out using transfer functions under the assumption of controllability and observability of the corresponding dynamic system.

The command **pole(tfmod)** finds the poles of a transfer function. The command**pzmap(tfmod)** plots the poles and zeros of the transfer function. A MATLAB code for the Routh stability criterion can easily be written for stability analysis. However, our focus in this course is on root locus for stability analysis. The focus in this MATLAB module will therefore be on root locus analysis.

MATLAB MODULE 6

Stability Analysis on Root Locus Plots

Root Locus Analysis

In the following, we provide brief description of powerful MATLAB commands for root locus analysis. The reader may wonder why the instructors emphasize learning of hand calculations when powerful MATLAB commands are available. For a given set of open-loop poles and zeros, MATLAB immediately plots the root loci. Any changes made in the poles and zeros, immediately result in new root loci and so on.

Depending on our background and aptitude, we may, after a while, begin to make some sense of the patterns. May be we finally begin to formulate a set of rules that enables us to quickly make a mental sketch of the root locus the moment the poles and zeros appear. In other words, by trial-and-error, we find the rules of the root locus.

Through the systematic formulation of set of rules of the root locus, we look for the clearest, and simplest explanation of the dynamic phenomena of the system. The rules of the root locus give us a clear and precise understanding of the endless patterns that can be created by an infinite set of characteristic equations. We could eventually learn to design without these rules, but our level of skill would never be as high or our understanding as great. This is true of other analysis techniques also such as Bode plots, Nyquist plots, Nichols charts, and so on, covered later in the course.

MATLAB allows root locus for the characteristic equation

$$1 + G(s)H(s) = 0$$

to be plotted with the **rlocus(GH)** command. Points on the root loci can be selected interactively (placing the cross-hair at the appropriate place) using the **[K,p] = rlocfind(GH)** command. MATLAB then yields the gain **K** at that point as well as all the poles **p** that have that gain. The root locus can be drawn over a grid generated using the**sgrid (zeta, wn)** command, that allows constant damping ratio **zeta** and constant natural frequency **wn** curves. The command **rlocus (GH, K)** allows us to specify the range of gain **K** for plotting the root locus. Also study the commands **[p,K]=rlocus(GH)** and**[p]=rlocus(GH,K)** using MATLAB online help.

Example M6.1

Consider the system shown in the block diagram of Fig. M6.1.

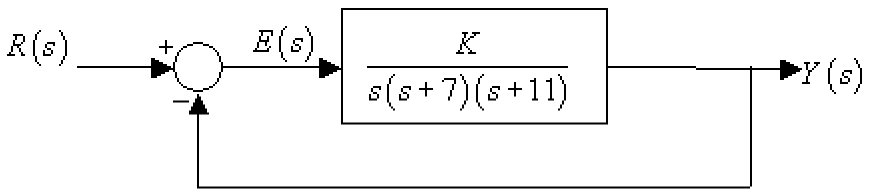

Fig. M6.1

The characteristic equation of the system is

$$1 + G(s) = 0$$

with

$$G(s) = \frac{K}{s(s+7)(s+11)}$$

The following MATLAB script plots the root loci for $0 \le K < \infty$.

```
s = tf('s');

G = 1/(s*(s+7)*(s+11));
```

```
rlocus(G);

axis equal;
```

Clicking at the point of intersection of the root locus with the imaginary axis gives the data shown in Fig. M6.2. We find that the closed-loop system is stable for $K < 1360$; and unstable for $K > 1360$.

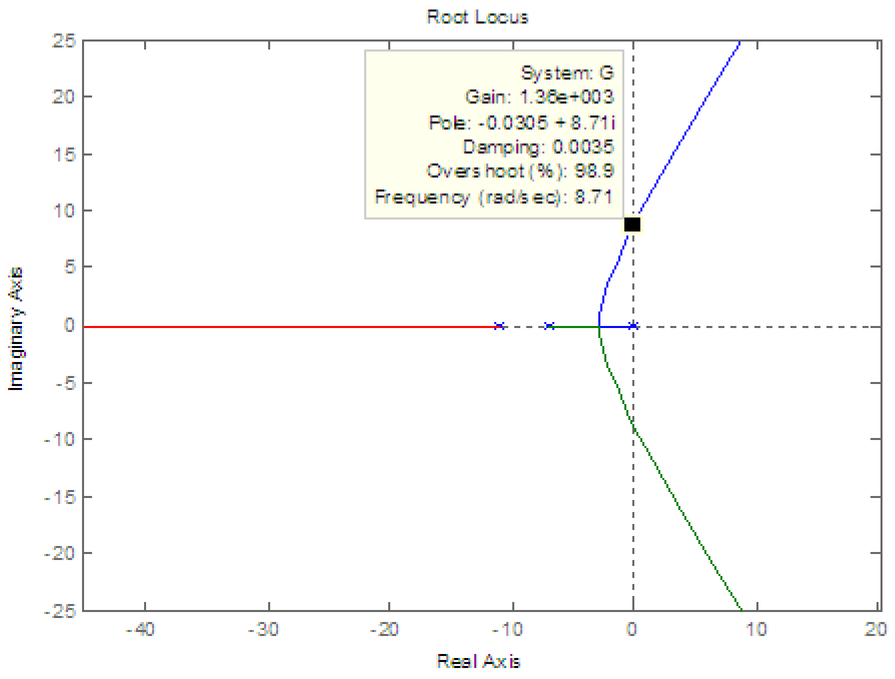

Fig. M6.2

Fig. M6.3 shows step responses for two values of K .

```
>> K = 860;

>> step(feedback(K*G,1),5)

>> hold;

% Current plot held

>> K = 1460;
>> step(feedback(K*G,1),5)
```

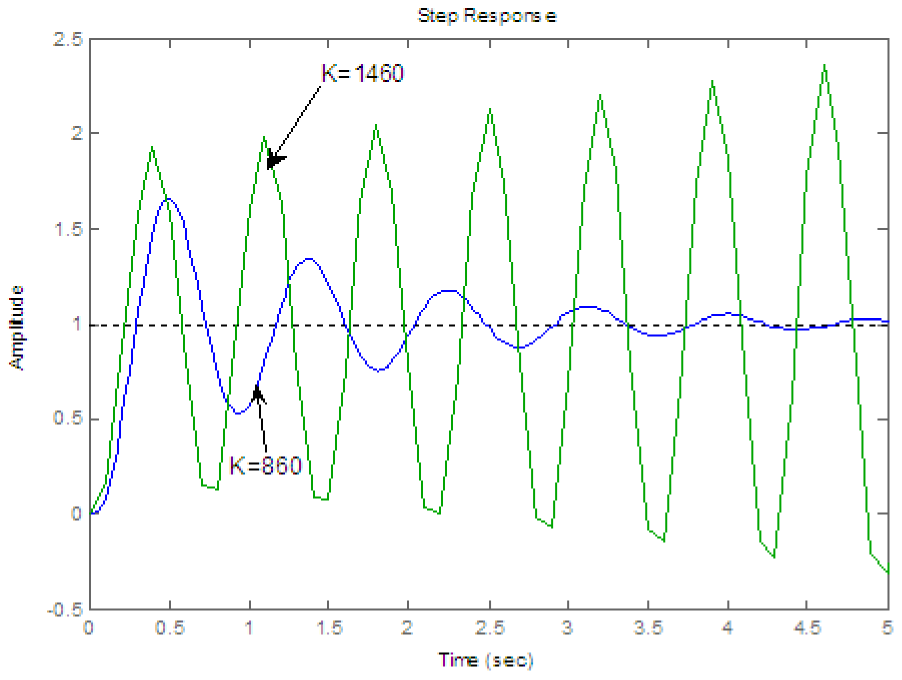

Fig. M6.3

Example M6.2

Consider the system shown in Fig. M6.4.

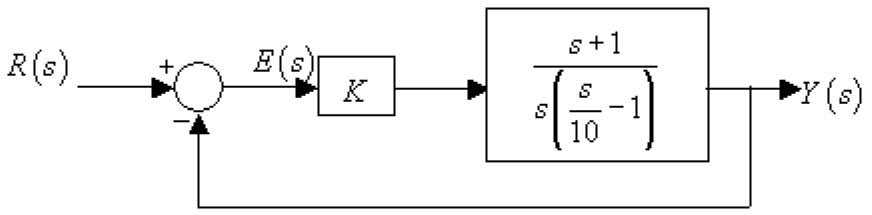

Fig M6.4

The plant transfer function *G*(*s*) is given as as

$$G(s) = \frac{s+1}{s(0.1s-1)}$$

The following MATLAB script plots the root locus for the closed-loop system.

```
clear all;

close all;

s = tf('s');

G = (s+1)/(s*(0.1*s-1));

rlocus(G);

axis equal;

sgrid;

title('Root locus for (s+1)/s(0.1s-1)');

[K,p]=rlocfind(G)
```

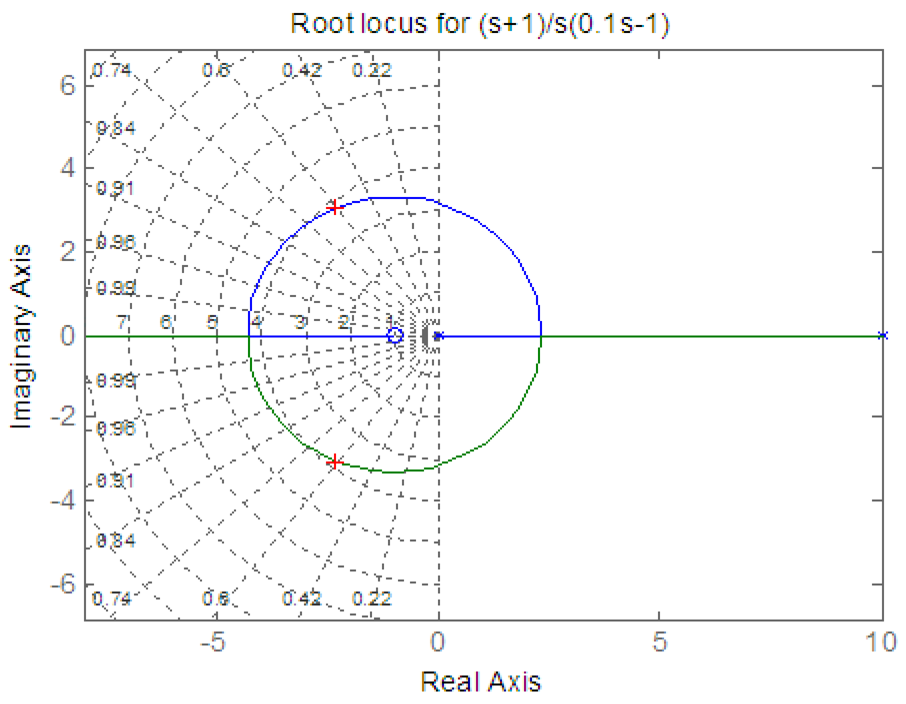

Fig M6.5

```
selected_point =

-2.2204 + 3.0099i
```

```
K =

1.4494

p =

-2.2468 + 3.0734i

-2.2468 - 3.0734i
```

Example M6.3

For a unity feedback system with open-loop transfer function

$$G(s) = \frac{K(s^2 - 4s + 20)}{(s+2)(s+4)}$$

a root locus plot shown in Fig. M6.6 has been generated using the following MATLAB code.

```
s = tf('s');

G = (s^2-4*s+20)/((s+2)*(s+4));

rlocus(G);

zeta = 0.45;

wn = 0;

sgrid(zeta,wn);
```

Properly redefine the axes of the root locus using **Right click --> Properties --> Limits.**

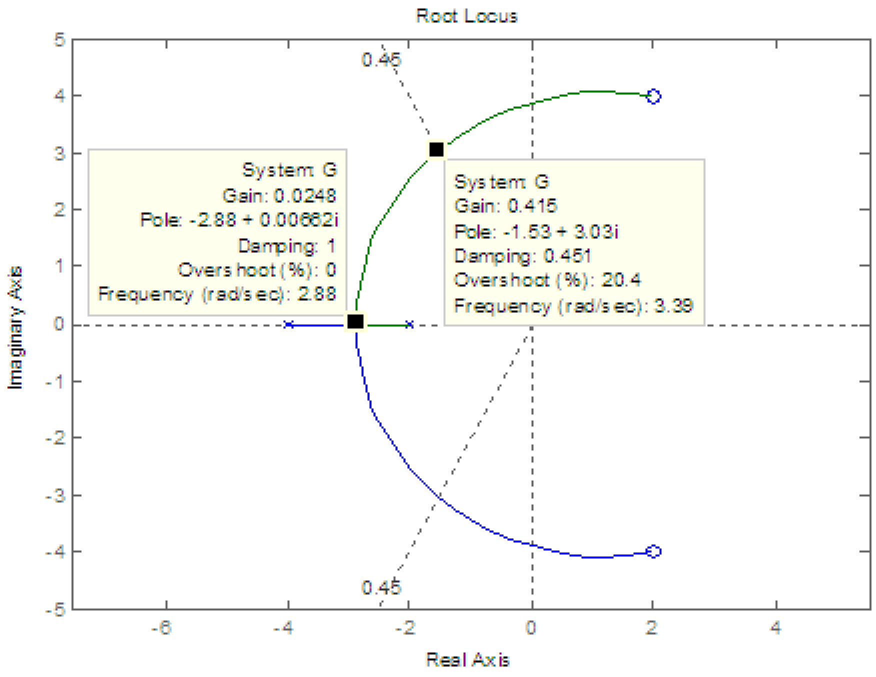

Fig. M6.6

Clicking on the intersection of the root locus with **zeta=0.45** line gives the system gain K = 0.415 that corresponds to closed-loop poles with $\zeta = 0.45$. Clicking on the intersection of the root locus with the real axis gives the breakaway point and the gain at that point.