

Automatic Control
Dr. Anil Kumar
Department of Mechanical & Industrial Engineering
Indian Institute of Technology, Roorkee

Lecture – 37
Steady State Error, Root Locus

So, welcome to the lecture on application of MATLAB in automatic control. So, we discussed in previous lecture how to find the transfer function, find the poles zeros and how to get the response like step response, impulse response or some arbitrary input, response under some arbitrary input.

So, in this lecture we will continue the response of first order and second order system under step input and then we will discuss about some steady state errors and root locus. So, here we have this some of these problems like how to plot the response of the following transfer functions.

(Refer Slide Time: 01:09)

PROBLEM

Plot the response of the following transfer functions.

a) $R(s) = \frac{1}{s}$ \rightarrow $G(s) = \frac{9}{s^2 + 9s + 9}$ \rightarrow $C(s)$

c) $R(s) = \frac{1}{s}$ \rightarrow $G(s) = \frac{9}{s^2 + 9}$ \rightarrow $C(s)$

b) $R(s) = \frac{1}{s}$ \rightarrow $G(s) = \frac{9}{s^2 + 2s + 9}$ \rightarrow $C(s)$

d) $R(s) = \frac{1}{s}$ \rightarrow $G(s) = \frac{9}{s^2 + 6s + 9}$ \rightarrow $C(s)$

e) $G(s) = \frac{50}{s+50}$

Ref - Norman S. Nise, Control Systems Engineering, Wiley, 2013

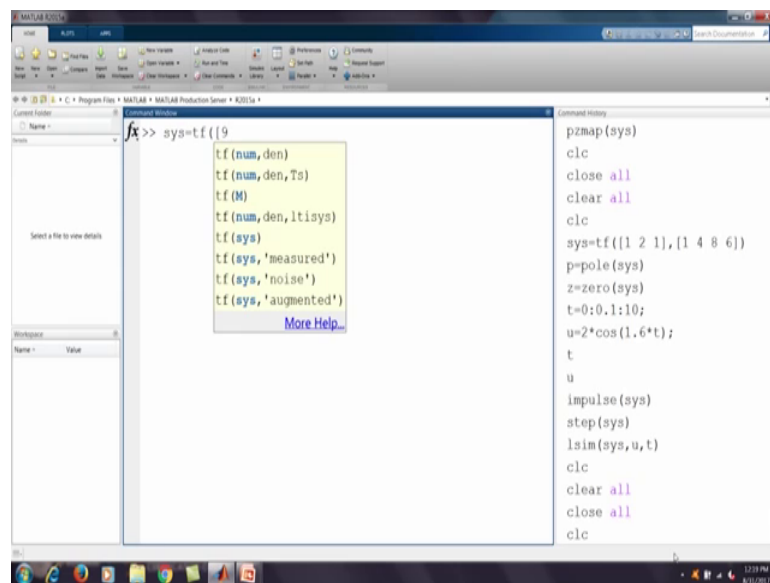
IIT ROORKEE
 NITEL ONLINE CERTIFICATION COURSE
 2

So, we have these transfer functions like 9 upon s square plus 9 is plus 9 and there is some step input because the R s of the step input unit step input is 1 by s.

So, how to find it and transfer function; So, here we can see that in the numerator there is the polynomial which has only one coefficient 9 and in the denominator we have coefficient 1 9 9. So, we have to first find the system using t of the transfer function. So, if we know that t f command we use t f num comma den in the parenthesis. So, here num

we will write only 9 and then here 1 9 9 and then we can find the response. So, here we will first find the system transfer function.

(Refer Slide Time: 02:16)

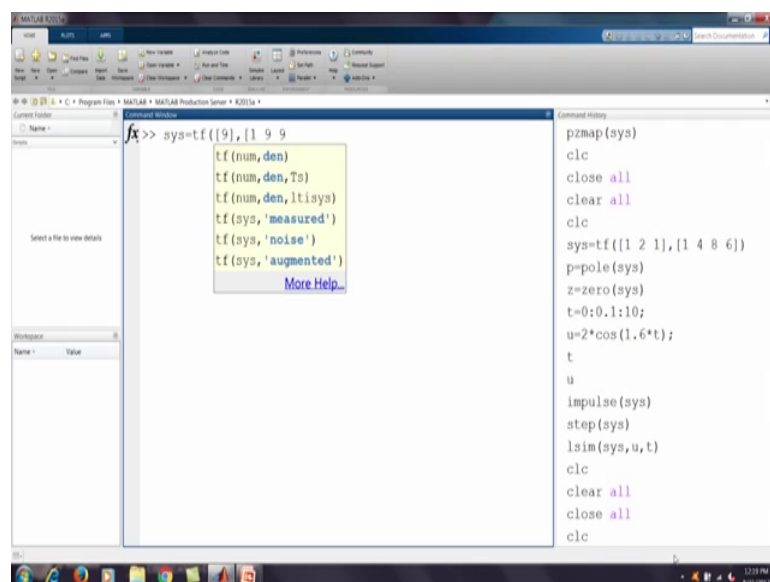


```
sys=tf(9  
tf(num,den)  
tf(num,den,Ts)  
tf(M)  
tf(num,den,ltsys)  
tf(sys)  
tf(sys,'measured')  
tf(sys,'noise')  
tf(sys,'augmented')  
More Help...
```

```
pzmap(sys)  
clc  
close all  
clear all  
clc  
sys=tf([1 2 1],[1 4 8 6])  
p=pole(sys)  
z=zero(sys)  
t=0:0.1:10;  
u=2*cos(1.6*t);  
t  
u  
impulse(sys)  
step(sys)  
lsim(sys,u,t)  
clc  
clear all  
close all  
clc
```

So, system equal to t f. Now, we define the numerator that is only one coefficient 9 and then comma the denominator.

(Refer Slide Time: 02:26)

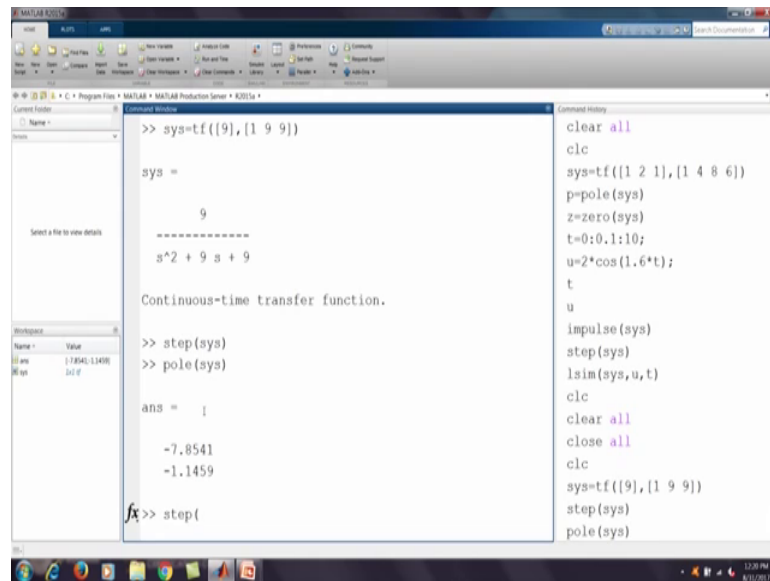


```
sys=tf([9],[1 9 9]  
tf(num,den)  
tf(num,den,Ts)  
tf(num,den,ltsys)  
tf(sys,'measured')  
tf(sys,'noise')  
tf(sys,'augmented')  
More Help...
```

```
pzmap(sys)  
clc  
close all  
clear all  
clc  
sys=tf([1 2 1],[1 4 8 6])  
p=pole(sys)  
z=zero(sys)  
t=0:0.1:10;  
u=2*cos(1.6*t);  
t  
u  
impulse(sys)  
step(sys)  
lsim(sys,u,t)  
clc  
clear all  
close all  
clc
```

So, we will have 1 9 9; So, 1 9 9. So, and then parentheses close.

(Refer Slide Time: 02:36)



```
>> sys=tf([9],[1 9 9])

sys =

      9
-----
s^2 + 9 s + 9

Continuous-time transfer function.

>> step(sys)
>> pole(sys)

ans =

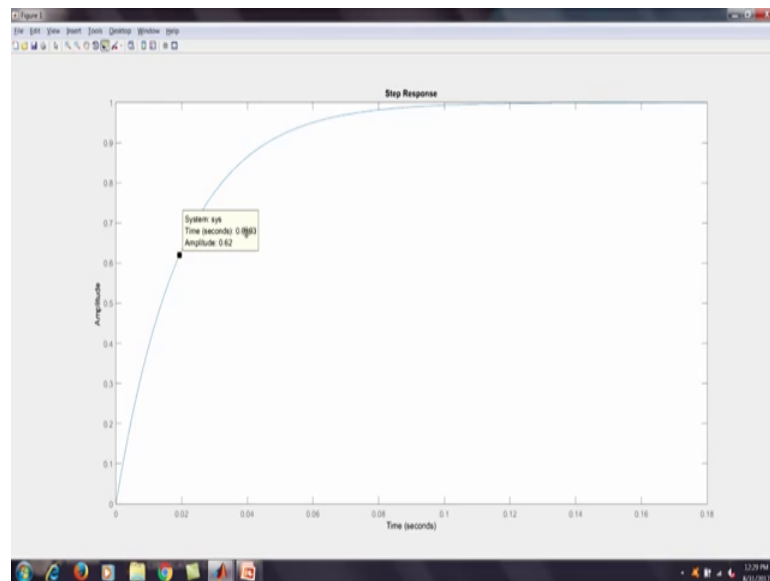
     1
    -7.8541
    -1.1459

fx>> step(
```

```
clear all
clc
sys=tf([1 2 1],[1 4 8 6])
p=pole(sys)
z=zero(sys)
t=0:0.1:10;
u=2*cos(1.6*t);
t
u
impulse(sys)
step(sys)
lsim(sys,u,t)
clc
clear all
close all
clc
sys=tf([9],[1 9 9])
step(sys)
pole(sys)
```

So, here we have got this transfer function 9 upon s square plus 9 s plus 9.

(Refer Slide Time: 02:53)

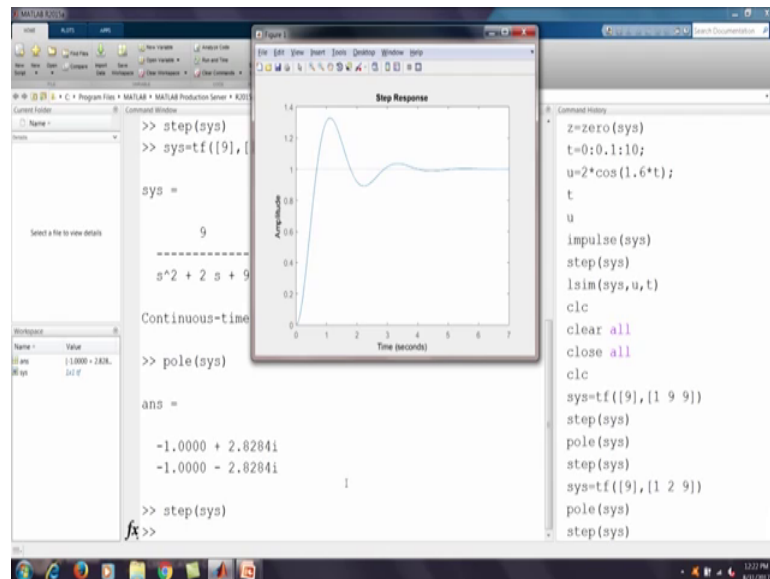


So, if you want to see the step response. So, we say step says. So, here we can see the step response of this system. So, there is unity step. So, you can see here this is the step input 1 and the system response is reaching towards the step input. Now, we want the pole of the system. So, we can write pole sys and we can get the pole minus 7.8 and minus 1.1459.

So, we see the poles are distinct and because this is a second order system and the poles are distinct we will get a response like what we got here step sys. So, this is the response that we will get.

Now, let us go for the second system that is here 9 upon s square plus $2s$ plus 9 . So, here we have in the denominator $1 \ 2 \ 9$. So, here we put the system we define sys; So, here $1 \ 2 \ 9$. So, this is a system now we get the pole of sys.

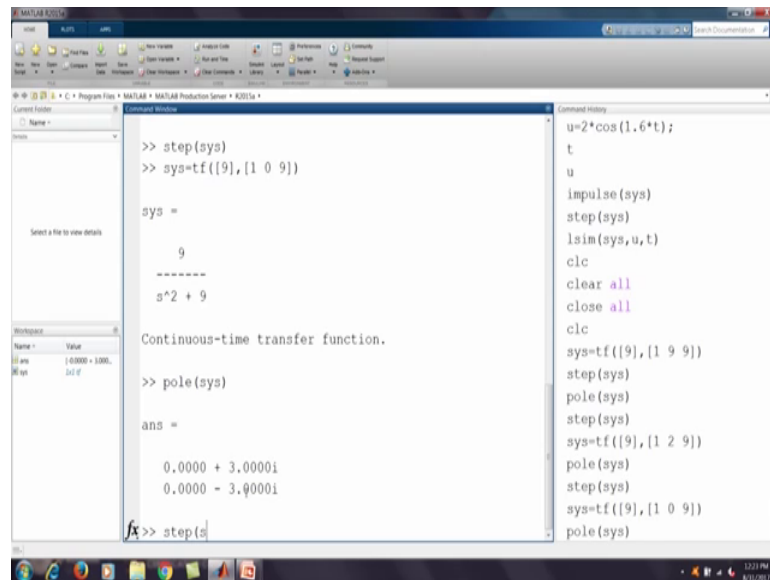
(Refer Slide Time: 04:26)



Here we can see that these are the complex pair poles complex poles. So, they are minus 1 plus $2.8i$ and minus 1 minus 2.8 . So, they are complex poles and if we have complex poles then this system responds again step sys we can find and you can see here when the poles are complex we have under damped system. So, damping is less than between less than 1 so, between 0 and 1 .

Now, the earlier system when the roots where the poles were distinct and real the system was over damped system. So, here we have under damped system and we can see that this amplitude is decaying with time and there are some oscillations. Now, we take the third example here 9 by s square plus 9 . So, here we have in the denominator $1 \ 0 \ 9$. So, here if I define the system I will say $1 \ 0 \ 9$. So, this is the system and transfer function is 9 upon s square plus 9 .

(Refer Slide Time: 05:55)



```
>> step(sys)
>> sys=tf([9],[1 0 9])

sys =

      9
  -----
 s^2 + 9

Continuous-time transfer function.

>> pole(sys)

ans =

    0.0000 + 3.0000i
    0.0000 - 3.0000i
```

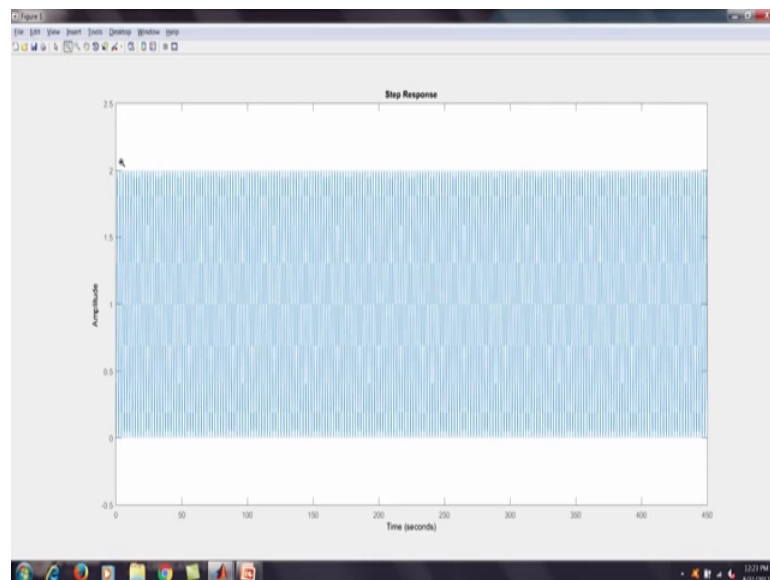
The image shows a MATLAB Command Window with the following code and output:

```
u=2*cos(1.6*t);
t
u
impulse(sys)
step(sys)
lsim(sys,u,t)
clc
clear all
close all
clc
sys=tf([9],[1 9 9])
step(sys)
pole(sys)
step(sys)
sys=tf([9],[1 2 9])
pole(sys)
step(sys)
sys=tf([9],[1 0 9])
pole(sys)
```

The system is defined as $\text{sys} = \frac{9}{s^2 + 9}$. The poles are calculated as $0.0000 + 3.0000i$ and $0.0000 - 3.0000i$.

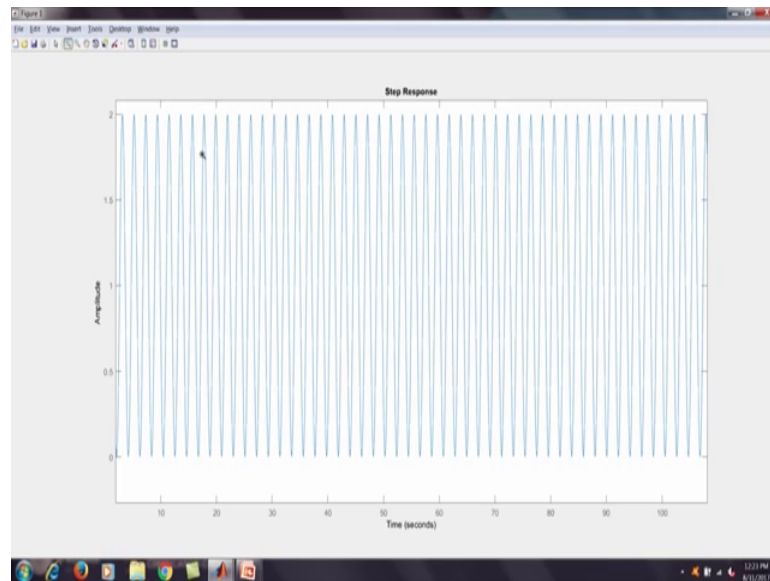
Now, if I get the pole of the system pole sys I will get that here we have real part is 0 and the poles are plus minus 3 i. So, these poles are on the j omega axis and. So, this is the case of critical damping and this we will see the response of this system against step input.

(Refer Slide Time: 06:26)



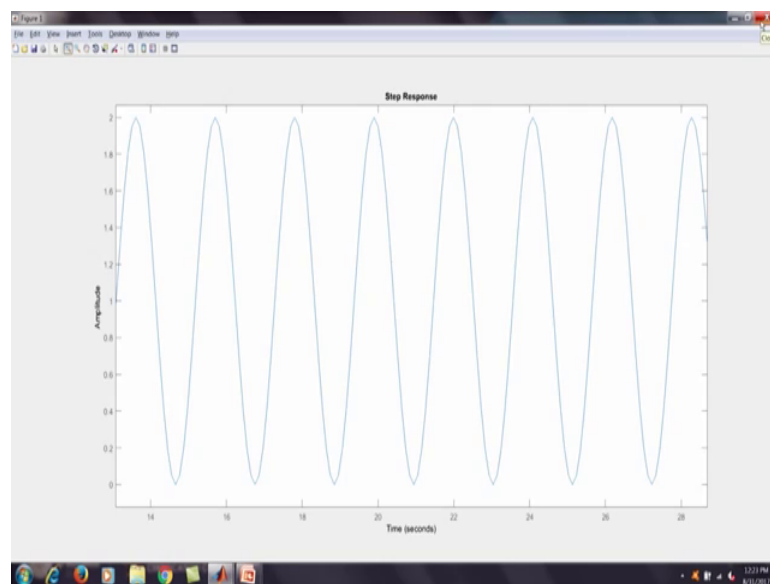
And, here know we know that when the poles are on the j omega axis there they have the oscillations.

(Refer Slide Time: 06:33)



And, we can see here the oscillations.

(Refer Slide Time: 06:35)



And, because they are plus minus 3, so, omega is 3 here; So, these are the oscillations because the poles are on the $j\omega$ axis. So, here we find this $s^2 + 9$ case.

Now, we have the another problem that is 9 upon $s^2 + 6s + 9$. No. So, so this case that is 9 upon $s^2 + 9$ is undamped undamped case.

$$G(s) = \frac{b}{s^2 + as + b}$$

$$\zeta = \frac{a}{2\sqrt{b}}$$

$$\omega_n = \sqrt{b}$$

$$\frac{9}{s^2 + 9}$$

$$a = 0, b = 9$$

$$\zeta = 0, \omega_n = \sqrt{9} = \underline{\underline{3}}$$

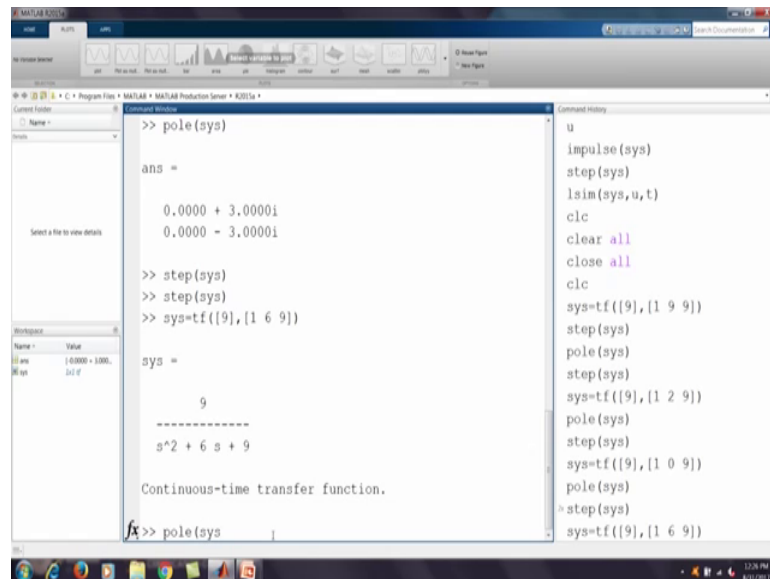
$$G(s) = \frac{50}{s + 50}$$

Because, here we have the damping that is damping is defined if we have a system like b upon s square plus a s plus b this is our G s . So, our damping a equal to is a by 2 root b and ω is root b .

So, here in this case when we treat the case 9 upon s square plus 9 , 9 upon s square plus 9 . So, we had here a equal to 0 and b equal to 9 . So, a is 0 . So, damping is 0 and b equal to 9 . So, ω_n equal to root 9 . So, we were getting 3 . So, this is the natural frequency of the system and this is undamped response of the system. So, what we saw here, this this is the undamped response. There is no damping in the system. So, this is the natural response of the system with the natural frequency that is ω_n equal to 3 .

Now, we come to the last case here this one; So, 9 upon s square plus 6 s plus 9 . So, here s is 6 here s coefficient.

(Refer Slide Time: 09:25)



```
>> pole(sys)

ans =

    0.0000 + 3.0000i
    0.0000 - 3.0000i

>> step(sys)
>> step(sys)
>> sys=tf([9],[1 6 9])

sys =

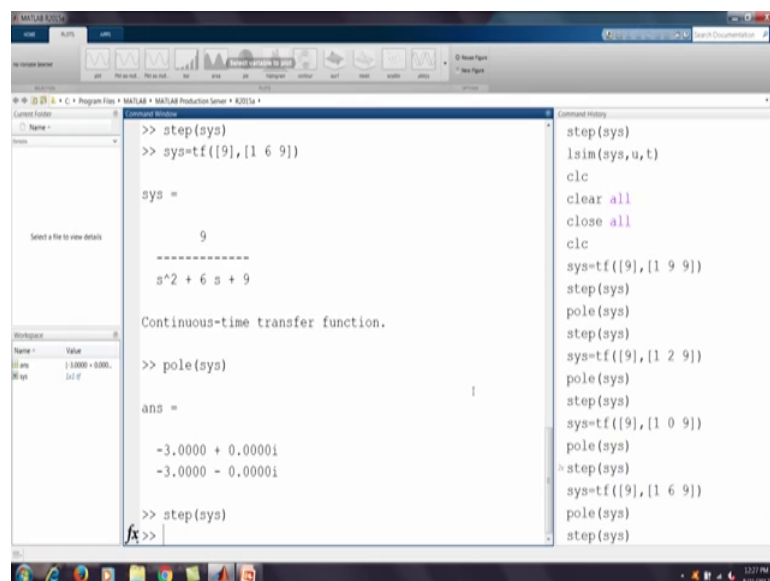
     9
-----
s^2 + 6 s + 9

Continuous-time transfer function.

fx>> pole(sys)
```

So, we can write the system here is 6. Now, we have 9 upon s square plus 6 s plus 9. So, we get the poles of the system. So, we get the pole. So, we get here the poles the imaginary part is 0, both the poles are on the real axis and they are equal.

(Refer Slide Time: 09:40)



```
>> step(sys)
>> sys=tf([9],[1 6 9])

sys =

     9
-----
s^2 + 6 s + 9

Continuous-time transfer function.

>> pole(sys)

ans =

 -3.0000 + 0.0000i
 -3.0000 - 0.0000i

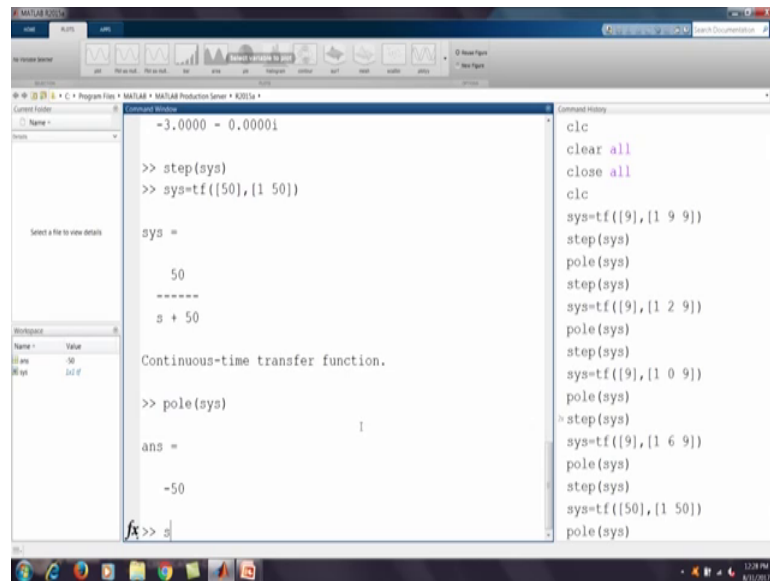
>> step(sys)

fx>>
```

So, this is the critical damping case and if we see the response of the system step sys, so, we will get this response. So, this is the critical damping response.

Now, we come to the last case that is G s equal to 50 upon s plus 50. So, we have G s equal to 50 upon s plus 50. So, we can see that this is a first order system because here the power of s is 1.

(Refer Slide Time: 10:43)



The screenshot shows the MATLAB Command Window with the following content:

```
-3.0000 - 0.0000i
>> step(sys)
>> sys=tf([50],[1 50])

sys =

    50
-----
s + 50

Continuous-time transfer function.

>> pole(sys)

ans =

    -50
```

The Command History window on the right shows the following commands:

```
clc
clear all
close all
clc
sys=tf([9],[1 9 9])
step(sys)
pole(sys)
step(sys)
sys=tf([9],[1 2 9])
pole(sys)
step(sys)
sys=tf([9],[1 0 9])
pole(sys)
step(sys)
sys=tf([9],[1 6 9])
pole(sys)
step(sys)
sys=tf([50],[1 50])
pole(sys)
```

So, now, if we want to write this in MATLAB, so, we define the system. So, $t f$ is 50 here in the numerator and here we will have only s plus 50, so, 1 and this is 50 and there will not be the this term. So, we have only 1 and 50 now, this is the system; So, 50 upon s plus 50. Now, we define the pole, so, pole sys that is equal to minus 50 now. We get the step response. So, we get it.

So, we can see that this is a first order response here and this response is leading to the unity step input and if we take this cursor here. So, we can see here amplitude is 0.62 and we define in this as it is happening at time equal to 0.0193 or is about 0.02. So, when from this curve we can find the time constant that is the response reaches to 63 percent and so, we can reach it the time constant we can get about 0.02 second. So, this is the response of the first order system. So, here we can obtain the response of first order and second order system in MATLAB.

(Refer Slide Time: 12:33)

PROBLEM

1. For the given system find the static error constants and find the expected error for the standard step, ramp and parabolic inputs using MATLAB.

Ref. Norman S. Nise, Control Systems Engineering, Wiley, 2013

IIT ROORKEE NITEL ONLINE CERTIFICATION COURSE 3

Now, come to the another problem that is the steady state error and steady static error constants. So, we have this unity feedback system and for the given system we have to find the static error constants and we have to find the transfer sorry error steady state error, for the standard input that is step ramp and parabolic input.

(Refer Slide Time: 13:14)

$$G(s) = \frac{500(s+2)(s+5)(s+6)}{s(s+8)(s+10)(s+12)}$$
$$K_p = \lim_{s \rightarrow 0} G(s) \Rightarrow e(\infty) = \frac{1}{1+K_p}$$
$$K_v = \lim_{s \rightarrow 0} s G(s) \Rightarrow e(\infty) = \frac{1}{K_v}$$
$$K_a = \lim_{s \rightarrow 0} s^2 G(s) \Rightarrow e(\infty) = \frac{1}{K_a}$$

So, here we have this function, that is, 500 so, s plus 2, s plus 5, s plus 6 and then s, s plus 8, s plus 10, s plus 12; so, this is the G s of the unitary feedback system and we have to find the static error constant and steady state error for step ramp and parabolic inputs. So, we know that this system is type one system and we know that if we the static error constants we have already defined. So, K p equal to G s; s tends to 0, then K v equal to s

G_s when limit s tends to 0 and K_a equal to $s^2 G_s$. So, these are the and from here we find static error equal to $\frac{1}{1 + K_p}$ we have already defined from here in finite equal to $\frac{1}{K_v}$ and here in finite equal to $\frac{1}{K_a}$. So, these formulas we have already discussed.

So, let us first define these numerator and denominator. So, we have. So, first we clean this screen, clc.

(Refer Slide Time: 15:20)

A screenshot of the MATLAB Command Window. The Command History on the right shows a sequence of commands: `sys=tf([9],[1 9 9])`, `step(sys)`, `pole(sys)`, `step(sys)`, `sys=tf([9],[1 2 9])`, `pole(sys)`, `step(sys)`, `sys=tf([9],[1 0 9])`, `pole(sys)`, `step(sys)`, `sys=tf([9],[1 6 9])`, `pole(sys)`, `step(sys)`, `sys=tf([50],[1 50])`, `pole(sys)`, `step(sys)`, `clc`, `close all`, and `clear all`. The Command Window shows the execution of `close all` and `clear all`.

Close all and clear all. So, we clear the workspace and closed all the figures and then again we clean this screen.

(Refer Slide Time: 15:33)

A screenshot of the MATLAB Command Window. The Command History on the right shows: `step(sys)`, `sys=tf([9],[1 2 9])`, `pole(sys)`, `step(sys)`, `sys=tf([9],[1 0 9])`, `pole(sys)`, `step(sys)`, `sys=tf([9],[1 6 9])`, `pole(sys)`, `step(sys)`, `sys=tf([50],[1 50])`, `pole(sys)`, `step(sys)`, `clc`, `close all`, `clear all`, `clc`, `numg=500*poly([-2 -5 -6])`, and `deng=poly([0 -8 -10 -12])`. The Command Window shows the execution of `numg=500*poly([-2 -5 -6])`, `deng=poly([0 -8 -10 -12])`, and `G=tf(numg,deng)`. A tooltip for the `tf` function is visible, listing options like `tf(num,den)`, `tf(num,den,Ts)`, `tf(num,den,lt,sys)`, `tf(sys,'measured')`, `tf(sys,'noise')`, and `tf(sys,'augmented')`.

So, now, we let us define num g equal to 500 the numerator we are defining with 500 into the polynomial minus 2 minus 5 minus 6. So, we define this polynomial for numerator. So, minus 2 minus 5 minus 6 into 500, then denominator; So, deng equal to, so, polynomial 0 minus 8 minus 10 minus 12. So, this is we have defined the denominator.

Now, we can get the transfer function G equal to t f num g comma deng.

(Refer Slide Time: 16:47)

```

>> numg=500*poly([-2 -5 -6]);
>> deng=poly([0 -8 -10 -12]);
>> G=tf(numg,deng);
>> G

G =

      500 s^3 + 6500 s^2 + 26000 s + 30000
-----
      s^4 + 30 s^3 + 296 s^2 + 960 s

Continuous-time transfer function.

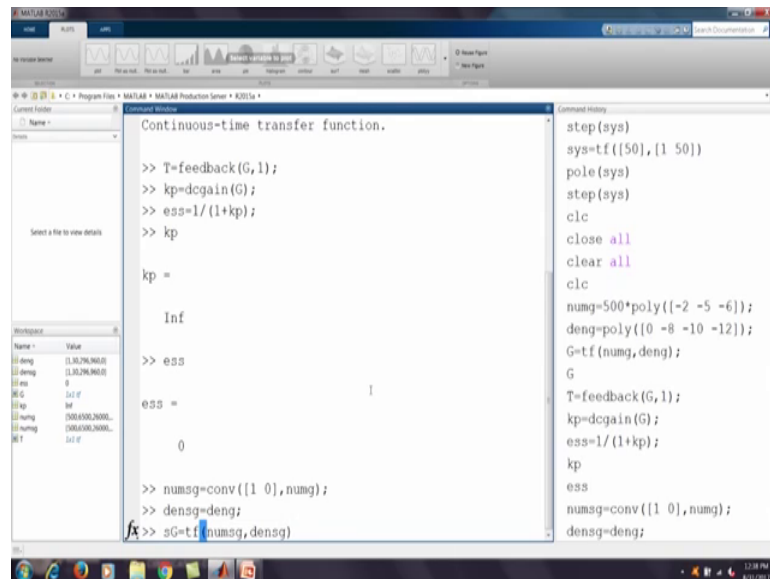
>> T=feedback(G,1);
>> kp=dcgain(G);
>> ess=1/(1+kp);
fx>>
  
```

So, we have obtained this transfer function. So, we can see here G if we enter G we have got 500 s cube plus 500 into 500. So, we have got this transfer function. Now, we have to find the equivalent transfer function we can get that is T equal to with unity feedback, so, T equal to feedback. Feedback we get G comma 1. So, unity feedback and we have got this transfer function. Now, for the step input we want to calculate K p.

So, here K p equal to dc gain dc gain G. So, what we are doing here dc gain evaluate the K p of G for s equal to 0. So, dc gain command sys putting just s equal to 0, because here K p define s tends to 0 for G s. So, this will be evaluated for s, when s is tending to 0. So, we will get K p.

Now, we will calculate the steady state error equal to 1 by 1 plus K p, 1 by 1 plus K K p. So, ok. So, we can see what is K p.

(Refer Slide Time: 18:44)



The image shows a MATLAB Command Window and Workspace. The Command Window displays the following code and output:

```
Continuous-time transfer function.
>> T=feedback(G,1);
>> kp=dcgain(G);
>> ess=1/(1+kp);
>> kp
kp =
    Inf
>> ess
ess =
     0
>> numsg=conv([1 0],numg);
>> densg=deng;
>> sG=tf(numsg,densg)
```

The Workspace shows the following variables and their values:

Name	Value
deng	(1.0e+05) [1 50]
densg	(1.0e+05) [1 50]
ess	0
G	1/10000000000
kp	Inf
numg	(5.0e+02) [500 0]
numsg	(5.0e+02) [500 0]
T	1/10000000000

The Command History shows the following commands:

```
step(sys)
sys=tf([50],[1 50])
pole(sys)
step(sys)
clc
close all
clear all
clc
numg=500*poly([-2 -5 -6]);
deng=poly([0 -8 -10 -12]);
G=tf(numg,deng);
T=feedback(G,1);
kp=dcgain(G);
ess=1/(1+kp);
kp
ess
numsg=conv([1 0],numg);
densg=deng;
sG=tf(numsg,densg)
```

So, K_p we have bought in finite because this is a type one system. So, if we give the step input we must get and ess equal to 0. So, it steady state error 0. Now, come to the ramp a ramp input. So, our for ramp input we have to get $s G s$. So, we have to multiply s in the numerator. So, we are doing this num numerator we are redefining num numsg equal to conv 1 0. So, this is a polynomial that is s and 0. So, that is the just s 1 0 and we are earlier num numg.

So, here we have conv, so, we have got s into the numerator of this $G s$ and densg equal to same as earlier denominator because we are not changing. So, we can say den s g equal to deng, deng because it is the same, but we have just renamed another variable also with the same denominator. So, densg equal to deng, so, we have got.

Now, we want $s G$ equal to $t f$ transfer function num sg comma den sg.

(Refer Slide Time: 21:04)

The screenshot shows a MATLAB Command Window with the following code and output:

```

>> numsg=conv([1 0],numg);
>> densg=deng;
>> sG=tf(numsg,densg);
>> kv=dcgain(sG);
>> ess=1/kv;
>> kv

kv =

    31.2500
tf(num,den)
>> ess
    0.0320
tf(num,den,ltisys)
ess =
    0.0320
tf(sys,'measured')
    0.0320
tf(sys,'noise')
tf(sys,'augmented')
>> nums2g=tf(DynamicSystem object...)
>> dens2g=
>> s2G=tf(nums2g,

```

Command History:

```

clc
numg=500*poly([-2 -5 -6]);
deng=poly([0 -8 -10 -12]);
G=tf(numg,deng);
G
T=feedback(G,1);
kp=dcgain(G);
ess=1/(1+kp);
kp
ess
numsg=conv([1 0],numg);
densg=deng;
sG=tf(numsg,densg);
kv=dcgain(sG);
ess=1/kv;
kv
ess
nums2g=conv([1 0 0],numg);
dens2g=deng;

```

Workspace:

Name	Value
deng	1x3 double
dens2g	1x3 double
densg	1x3 double
ess	0.0320
G	tf
kp	31
kv	31.2500
numg	500x3 double
numsg	500x3 double
num2g	500x3 double
s2G	tf
sG	tf

So, we have defined s into G . So, now, we have obtained s into G . Now, we have to put the limit s equal to 0 so, by using dc gain command. So, to get kv . So, kv equal to dc gain sG . So, kv equal to dc gain sG . So, we have got and ess now equal to 1 by kv . So, we can see what is kv . So, kv equal to 31.25. So, this is type one system we are getting a finite kv value and we will get the finite error.

So, we are getting 0.032 steady state error for this system now, we go for parabolic input. So, parabolic input we have to put s square G we have to get s square G . So, we will repeat this num num 2 g equal to $conv$ $conv$. So, here we have one 00. So, it is just s square plus 0 s plus 0. So, s square and you multiply $conv$ function multiplies this with the num G .

So, we are getting the numerator for the parabolic input and then $dens$ $dens$ 2 G equal to same as because this is the same. So, we can say $deng$. So, we got this now s square G or s 2 G we say s 2 G equal to transfer function that is num 2 num 2 G comma $dens$ 2 G and. So, now, we will use here we can also use one more command s 2 G equal to mean real s 2 G . So, that we forgot to use in the last part. So, here mean real command will cancel the common terms in the numerator and denominator because we do not need those terms they will if they are in the polynomial in the numerator there is suppose s plus 2 and in the denominator also s plus 2 they will be cancelled.

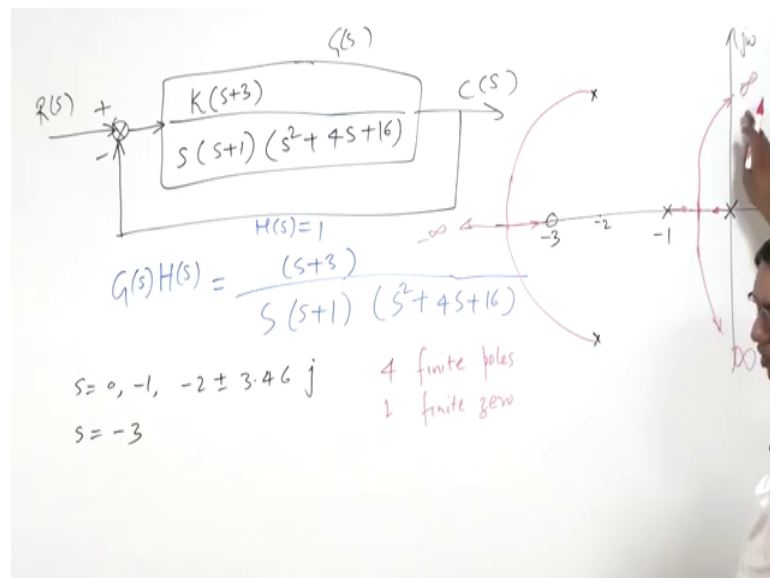
So, this command we have to use mean real s 2 G and once this we do then we can get k a. So, k a equal to dc gain s 2 G . So, we get k a and then ess steady state error one by k a.

So, you can here we can see what is ka? ka we are getting 0 and ess we are getting infinite. So, this is how we have obtained this steady state error and static error constants for given system.

Now, we come to the next problem that is find the root locus. So, we have to find the root locus for this. So, we know that we have already discussed about root locus techniques. So, root locus represents the movement of closed loop poles when we vary the gain. So, we discussed several rules that may help us to identify the branches of the root locus and to catch the root locus.

However, in MATLAB there is a very simple command that can plot the root locus. So, we will see we have this system.

(Refer Slide Time: 26:07)



That is K s plus 3 upon s s plus 1. So, this is the. So, this is our input and this is the G s and this is unity feedback. So, this is our system and we have to plot the root locus. So, in order to plot the root locus, ok; So, first we see here this G s H s, open loop transfer function what is this G s H s?

(Refer Slide Time: 27:07)

```
>> num=[1 3];
>> den=conv([1 1 0],[1 4 16]);
>> sys=tf(num,den)

sys =

      s + 3
-----
s^4 + 5 s^3 + 20 s^2 + 16 s + 1
Continuous-time transfer function.

fx>> pole(sys)
```

The screenshot shows the MATLAB Command Window with the following code and output:

```
>> num=[1 3];
>> den=conv([1 1 0],[1 4 16]);
>> sys=tf(num,den)

sys =

      s + 3
-----
s^4 + 5 s^3 + 20 s^2 + 16 s + 1
Continuous-time transfer function.

fx>> pole(sys)
```

The Command History window shows the following commands:

```
kv=dcgain(sG);
ess=1/kv;
kv
ess
num2g=conv([1 0 0],numg);
dens2g=deng;
s2G=tf(num2g,dens2g);
s2G=minreal(s2G);
ka=dcgain(s2G);
ess=1/ka;
ka
ess
clc
close all
clear all
clc
num=[1 3];
den=conv([1 1 0],[1 4 16]);
sys=tf(num,den)
```

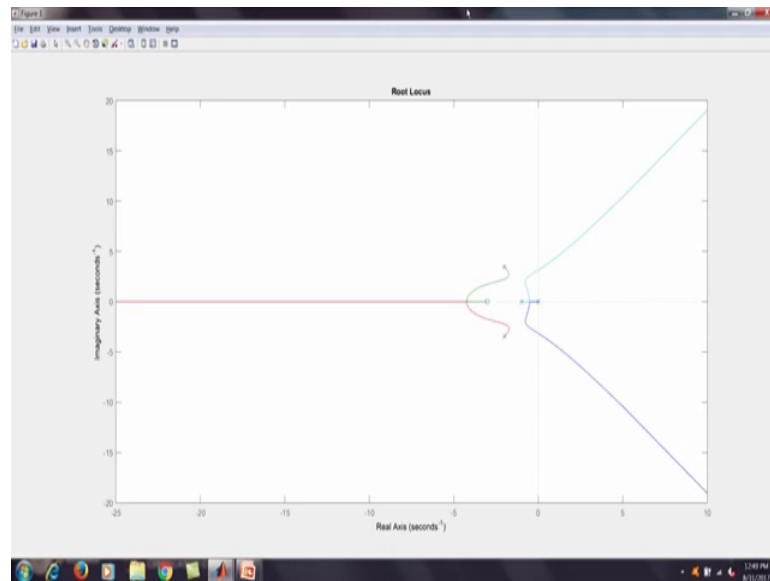
So, $H(s)$ is 1 here. So, $G(s)H(s)$ is the same as this. So, that is $s + 3$ upon $s^4 + 5s^3 + 20s^2 + 16s + 1$. So, this is $G(s)H(s)$. Now, for this open loop transfer function this is called open loop transfer function we have. So, we have this open loop transfer function and. So, first we find the transfer function. So, we define the num equal to

So, 1 and 3 because this $s + 3$ the coefficient are 1 and 3 then denominator den equal to. So, denominator is equal to we have some factor. So, convolution we will use the multiplication of two polynomials. So, first we take this $s^2 + s$. So, we will have 1, 1, 0, 1, 1, 0 first polynomial, then we will have 1, 4, 16 the second polynomial. So, s^2 coefficient is 1 then s coefficient is 4 and then the constant term that is 16. So, 1, 4, 16.

. So, we have now we can define the system that is equal to tf transfer function num comma den. So, this is the transfer function we define. So, we obtain this transformer $s + 3$ upon $s^4 + 5s^3 + 20s^2 + 16s + 1$, this is the transfer function. Now, if we calculate the pole of the system then we get these poles.

So, here we have 0 minus 1 and these two complex poles and zeros. So, 0 sys. So, open loop poles and open loop zeros. So, that is minus 3. Now, we if we want to plot the root locus. So, we just simple command `rlocus`, `rlocus num comma den`. So, this simple command will plot the root locus and we can see here this root locus.

(Refer Slide Time: 30:40)



So, now let us I connect this with the theory. So, we know that first we calculate this poles and these poles we obtained here we can see here these poles 0 minus 1 and then minus 2 plus minus 3.46 j. So, these are the 4 poles and the 0 was at s equal to minus 3. So, now we can focus on this root locus. So, I said that on the root locus we must first plot this σ j ω axis we plot the open loop poles; So, here 0 minus 1 then here minus 2; So, plus minus 3. So, these are here these two poles and there is one 0 at minus 3. So, here at minus 3 this is minus 2 and here is minus 3, there is a 0.

So, we have 4 poles and we know that here the root locus will lie in this region. So, the root locus will start from here from here and it should break away at the some point and then root locus will not lie in this region. So, odd it will lie to the left of the odd. So, 1 and then here we have to hear it or not, but here 1 2 3 4 here also not lie, but it will lie here it will reach. So, we have we can see we have four finite poles and one finite 0. So, the one locus will reach to this minus 3, but others 3 will reach to the infinite.

So, how it will happen; So, these are going to break away and let us they are going like this these two poles they will start and they will meet because they will they will move in similar way because the 2 poles will be in pair and one will reach to this and other will reach to the infinite and they will reach to the infinite am and following certain asymptotes, we can find that asymptotes and we can also find where they are cutting the j ω axis.

So, here we can see this thing. So, these poles going here then leading to the unstable region and we can find what is the this unstable region we can see here that here we can find the breakaway point. So, we can see that at gain 1.43 they are breaking away at minus 0.524 and the real axis. Now, we can see here is breaking point because they are getting here at breaking and we can see that at minus 4.18 and at gain 189 they are breaking to the real axis.

Now, we can have we can see the this is $j\omega$ axis crossing. So, here at gain 33.9 and here this would be 0, then about 3.16 of ω , they are crossing to the $j\omega$ axis and they are going to the left. So, the gain limit is 33.9 we cannot keep the gain more than 33.9, otherwise they will reach to the unstable region.

So, here we understood how to find the response of second order system, first order system, how to find the steady state error under different inputs that is step ramp and parabolic of a given feedback system, then how to plot the root locus using MATLAB and how we can relate these plots with theory that is already discussed in the lectures and we can give some get some insight into the quick insight into the problem.

So, I stop this lecture here and we will continue in the next lecture.

Thank you.