

Scilab Manual for  
Digital Control System  
by Prof Deepti Khimani  
Instrumentation Engineering  
VESIT<sup>1</sup>

Solutions provided by  
Prof Deepti Khimani  
Instrumentation Engineering  
Mumbai/ V.E.S. Institute of Technology

January 31, 2025

<sup>1</sup>Funded by a grant from the National Mission on Education through ICT, <http://spoken-tutorial.org/NMEICT-Intro>. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website <http://scilab.in>



# Contents

List of Scilab Solutions	4
1 Represent the given discrete-time (sampled data) sytem using pulse transfer function and state space forms.	6
2 Discretize the given continuous-time signal using Zero Order Hold and First Order Hold techniques.	11
3 Discretize the given continuous-time system using Bilinear Transformation.	14
4 Map constant attenuation loci from s-plane to z-plane.	18
5 Map constant frequency loci from s-plane to z-plane.	22
6 Map constant constant damping ratio loci from s-plane to z-plane.	26
7 Transform the discrete-time state model into canonical forms.	30
8 Check controllability & observability of a given system.	33
9 Design state feedback controller for a given system to achieve desired dynamic characteristics.	35
10 Design dead beat controller for a given system.	40
11 Design full state observer for a given system.	45

12 Determine stability of a given system by using Lyapunov stability analysis.	49
13 Construct root loci and bode plots for a given discrete-time system.	51

# List of Experiments

Solution 1.01	Lab01	.....	6
Solution 2.02	Lab02	.....	11
Solution 3.03	Lab03	.....	14
Solution 4.04	Lab04	.....	18
Solution 5.05	Lab05	.....	22
Solution 6.06	Lab06	.....	26
Solution 7.07	Lab07	.....	30
Solution 8.08	Lab08	.....	33
Solution 9.09	Lab09	.....	35
Solution 10.10	Lab10	.....	40
Solution 11.11	Lab11	.....	45
Solution 12.12	Lab12	.....	49
Solution 13.13	Lab13	.....	51

# List of Figures

1.1	Lab01	9
1.2	Lab01	10
2.1	Lab02	12
2.2	Lab02	13
3.1	Lab03	17
4.1	Lab04	20
4.2	Lab04	21
5.1	Lab05	24
5.2	Lab05	25
6.1	Lab06	28
6.2	Lab06	29
9.1	Lab09	36
9.2	Lab09	37
10.1	Lab10	41
10.2	Lab10	42
11.1	Lab11	48
13.1	Lab13	53
13.2	Lab13	54

# Experiment: 1

Represent the given discrete-time (sampled data) system using pulse transfer function and state space forms.

Scilab code Solution 1.01 Lab01

```
1 // Lab01: (A) Represent the given discrete-time (
  sampled data) system using
2 //           pulse transfer function and state
  space forms.
3 //           (B) Observe the responses of continuous-
  time and sampled data system.
4
5 //           scilab - 5.5.1
6 //           Operating System : Windows 7, 32-bit
7 //
  *****

8 //Clean the environment
9 close;
10 clear;
```

```

11 clc;
12 //
    *****

13 //A. Representation of discrete-time models
14 //
    *****

15
16 //State space representation
17 A=[0 1;-0.2 -0.1];
18 B=[0 1]';
19 C=[1 0];
20 D=0;
21 sysd=syslin('d',A,B,C,D);
22
23 // Pulse transfer function representation
24 Num=1;
25 Den=poly([0.2 0.1 1],'z','coeff');
26 Gz1=syslin('d',Num,Den)
27
28 //Pulse transfer function from ss model
29 Gz2=ss2tf(sysd)
30
31 //Response of the system with sampling time Ts=0.5
    sec.
32
33 //sampling Time
34 Ts=0.5;
35 t=0:Ts:10;
36 u=ones(1,length(t));
37
38 y=flts(u,sysd);
39 plot2d2(t,y,2)
40 zoom_rect([0 0 10 1.2])
41 xgrid(35)
42 title('Response of discrete time system','fontsize'
    ,3)

```



```

43 xlabel('kT','fontsize',2)
44 ylabel('y(kT)','fontsize',2)
45
46 //
    *****

47 // (B) Responses of the continuous-time and discrete
    -time model of the given
48 //      system.
49 //
    *****

50
51 //poles of contiuous time system are at -1, -2 and
    -3;
52 den=poly([-1,-2,-3],'s','roots');
53 num=1;
54 g=num ./den
55 g=syslin('c',g)
56 tc=0:0.2:10;
57 yc=csim("step",tc,g)
58
59 //Discrete-time respresentation with Ts=0.5;
60 sysz=dscr(g,Ts);
61 gz=ss2tf(sysz)
62 yd=flts(u,sysz);
63
64 //Responses
65 figure,
66 plot(tc,yc,'blue') //continuous time system
67 plot2d2(t,yd,5) // Discrete time system
68 title('Responses of continuous and discrete time
    system','fontsize',3)
69 xlabel('t','fontsize',2)
70 ylabel('y(t), (y(kT))','fontsize',2)
71 f=get("current_figure") //Current figure handle
72 f.background=8
73 xgrid(36)

```

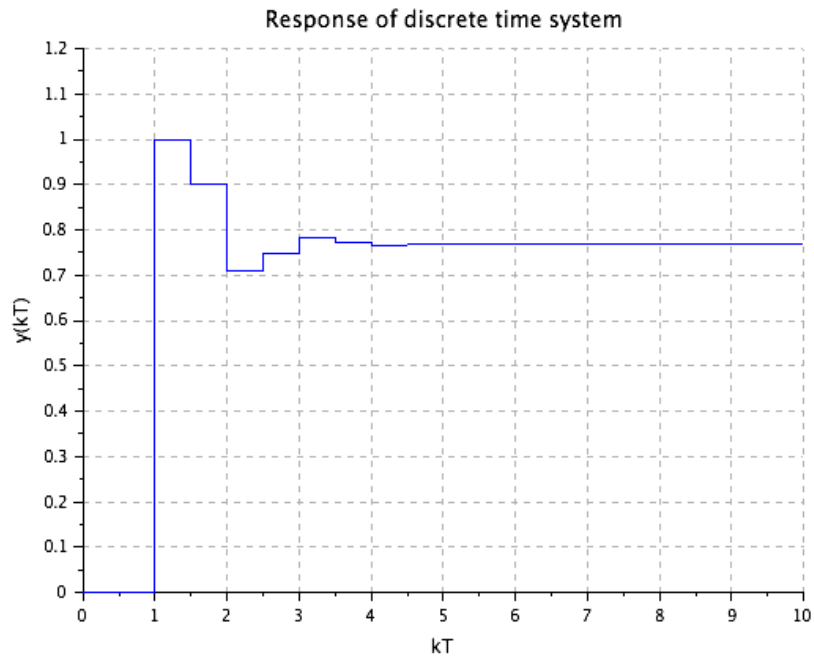


Figure 1.1: Lab01

```
74 h=legend('y(t)', 'y(kT)')  
75 h.legend_location = "in_lower_right"
```

---

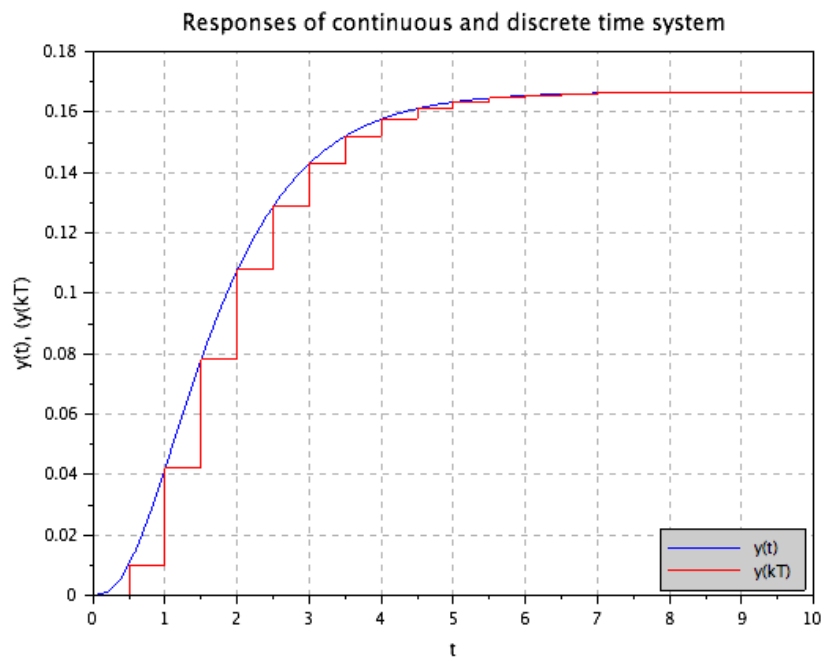


Figure 1.2: Lab01

## Experiment: 2

Discretize the given continuous-time signal using Zero Order Hold and First Order Hold techniques.

Scilab code Solution 2.02 Lab02

```
1 // Lab02: Discretize the given continuous-time
  signal using Zero Order Hold
2 //           and First Order Hold techniques.
3
4 //           scilab - 5.5.1
5 //           Operating System : Windows 7, 32-bit
6 //
  *****

7 //Clean the environment
8 close;
9 clear;
10 clc;
11
12 //ramp generator during sampling period
```

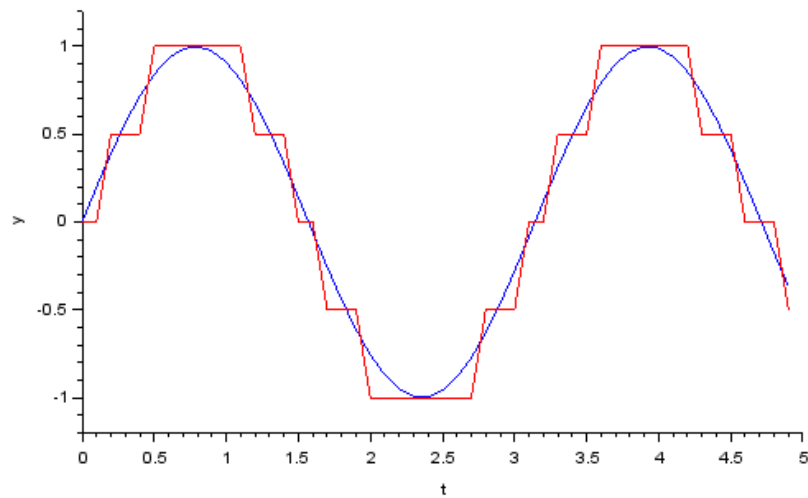


Figure 2.1: Lab02

```

13 tau.time = (0:0.5:1)';
14 tau.values = (0:0.5:1)';
15 importXcosDiagram('Lab02model.xcos')
16 typeof(scs_m) //The diagram data structure
17 xcos_simulate(scs_m, 4);

```

---

This code can be downloaded from the website [www.scilab.in](http://www.scilab.in)

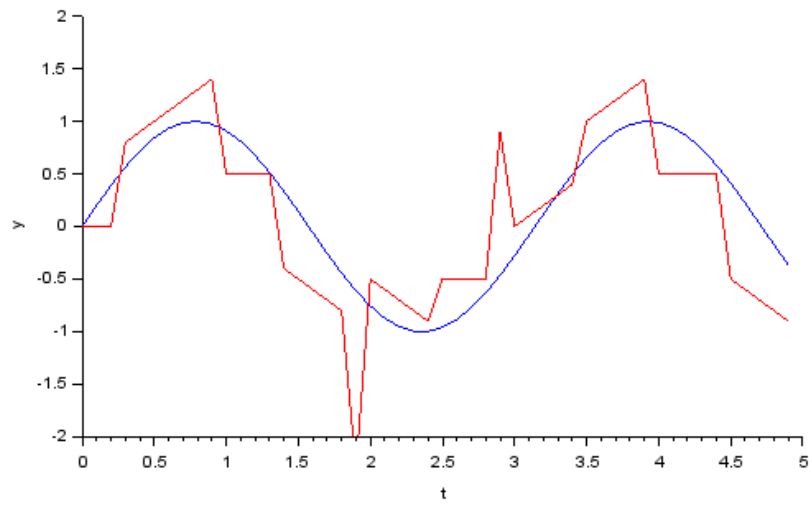


Figure 2.2: Lab02

## Experiment: 3

# Discretize the given continuous-time system using Bilinear Transformation.

Scilab code Solution 3.03 Lab03

```
1 // Lab03: Discretize the given continuous-time
  system
2 //      using Bilinear Transformation.
3
4 //      scilab - 5.5.1
5 //      Operating System : Windows 7, 32-bit
6 //
  *****

7 //Clean the environment
8 close;
9 clear;
10 clc;
11
12 //
  *****
```

```

13 // State space representation of continuous time
    system
14 //
    *****

15 //
16 A=[0 1;-6 -5];
17 B=[0 1]';
18 C=[1 0];
19 D=0;
20 sysc=syslin('c',A,B,C,D);
21 //transfer function
22 Gs=ss2tf(sysc)
23 disp('Gs=')
24 disp(Gs)
25 //Response of the system
26 tc=0:0.1:10;
27 yc=csim("step",tc,sysc);
28
29 //
    *****

30 // Discretization of the system using bilinear
    transformation at
31 // sampling time Ts=0.5 sec
32 //
    *****

33 Ts=0.5;
34 sysd=cls2dls(sysc,Ts);
35 //Pulse transfer function
36 Gz=ss2tf(sysd)
37 disp('Gz=')
38 disp('Gz=',Gz)
39 //Response of the discrete system
40 td=0:Ts:10;
41 ud=ones(1,length(td));
42 yd=flts(ud,sysd);

```



```

43
44 //
    *****

45 // Ploting the responses
46 //
    *****

47 plot2d(tc,yc,5)//continuous time
48 plot2d2(td,yd,2) // Discrete time
49 xgrid(35)
50 title('Responses of continous and Tustin transformed
        discrete time system','fontsize',3)
51 xlabel('kT','fontsize',2)
52 ylabel('y(kT)','fontsize',2)
53 h=legend('y(t)', 'y(kT)')
54 h.legend_location = "in_lower_right";

```

---

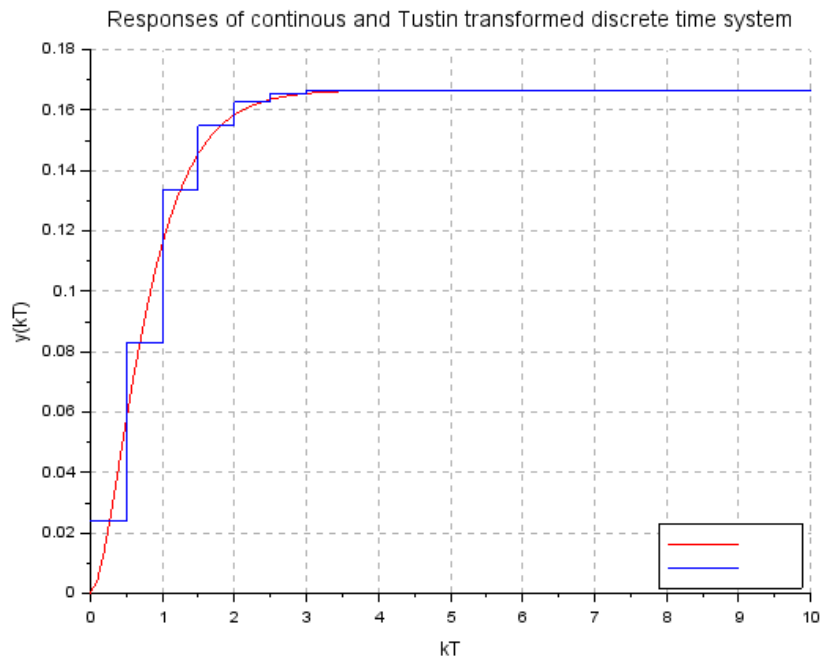


Figure 3.1: Lab03

# Experiment: 4

## Map constant attenuation loci from s-plane to z-plane.

Scilab code Solution 4.04 Lab04

```
1 //Lab. 04: Map constant attenuation loci from s-
  plane to z-plane
2
3 //          scilab - 5.5.1
4 //          Operating System : Windows 7, 32-bit
5 //
  *****

6 //Clean the environment
7 close;
8 clear;
9 clc;
10
11 //// System model
12 s=poly(0, 's');
13 Ts=0.2;
14 //
  *****
```

```

15 // S-plane pole-zero map for the continuous time
    system
16 //
    *****

17 for sigma=[-1 1];
18 for w=0:0.2:16
19     num=1;
20     den=poly([-sigma+%i*w, -sigma-%i*w], 's');
21     g=syslin('c', num./den);
22     plzr(g)
23 end
24 end
25 sgrid()
26 title('S-plane pole-zero map for the constant
    attenuation loci', 'fontsize', 3)
27
28 // Zoom axes for clarity
29 zoom_rect([-5 -20 5 20])
30
31 //
    *****

32 //Z-plane pole-zero map for the sample data system
33 //
    *****

34     figure;
35 for sigma=[-1 1]
36 for w=0:0.2:16
37     num=1;
38     den=poly([-sigma+%i*w, -sigma-%i*w], 's');
39     g=syslin('c', num./den);
40     gz=dscr(g, Ts)
41     plzr(gz);
42 end
43 end
44 zgrid();

```

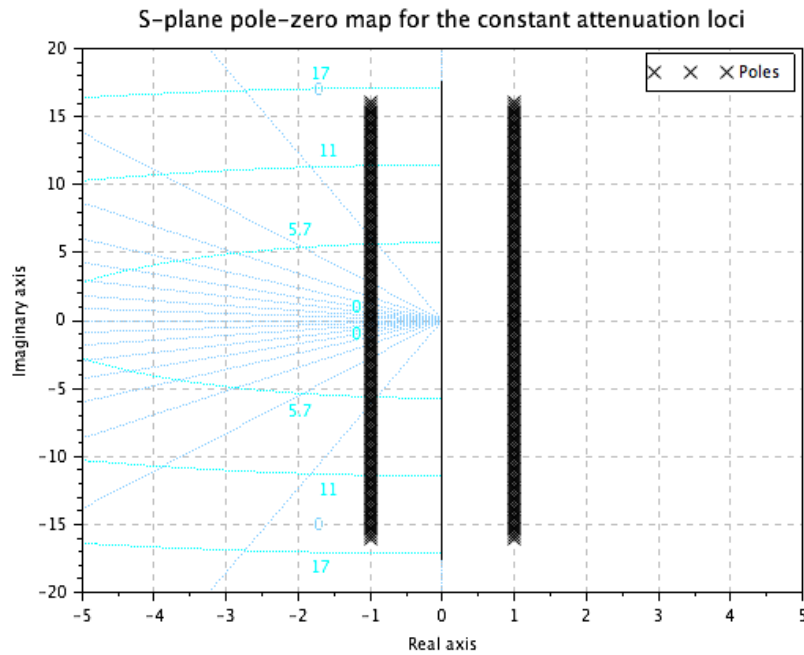


Figure 4.1: Lab04

```

45 f=get("current_figure") //Current figure handle
46 f.background=8
47 title('Z-plane pole-zero map for the constant
         attenuation loci','fontsize',3)
48
49 // Zoom axes for clarity
50 zoom_rect([-1.5 -1.5 1.5 1.5])

```

---

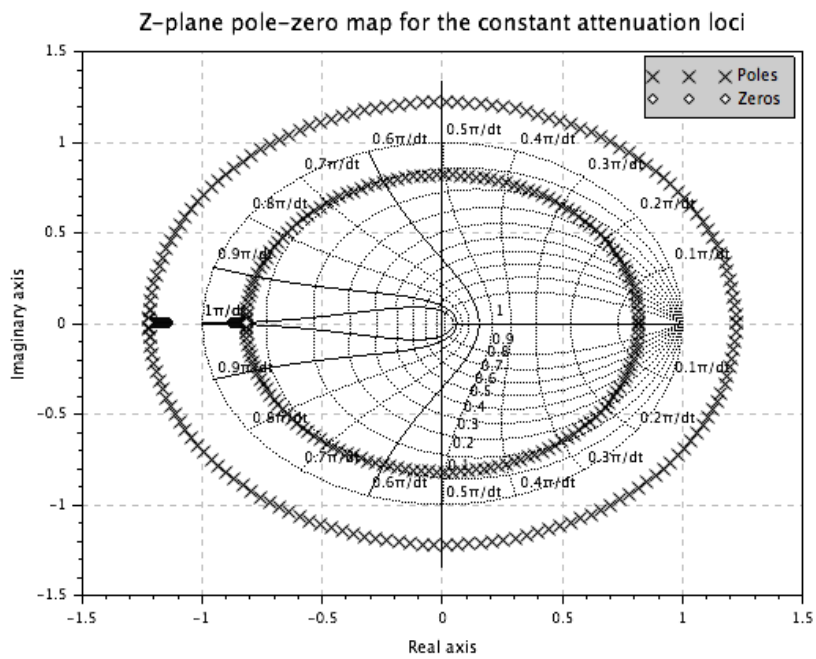


Figure 4.2: Lab04

# Experiment: 5

## Map constant frequency loci from s-plane to z-plane.

Scilab code Solution 5.05 Lab05

```
1 //Lab. 05: Map constant frequency loci from s-plane
  to z-plane
2
3 //          scilab - 5.5.1
4 //          Operating System : Windows 7, 32-bit
5 //
  *****

6 //Clean the environment
7 close;
8 clear;
9 clc;
10
11 // System model
12 s=poly(0, 's');
13 Ts=0.2;
14 w=1.5;
15
16 //
```

```

*****

17 // S-plane pole-zero map for the continuous time
    system
18 //
    *****

19 for sigma=-1.1:0.2:20
20     num=1;
21     den=poly([-sigma+%i*w, -sigma-%i*w], 's');
22     g=syslin('c',num./den);
23
24     plzr(g)
25 end
26 sgrid()
27 title('S-plane pole-zero map for the constant
        frequency loci','fontsize',3)
28 // Zoom axes for clarity
29 zoom_rect([-21 -10 5 10])
30
31 //
    *****

32 //Z-plane pole-zero map for the sample data system
33 //
    *****

34     figure;
35 for sigma=-1.1:0.2:20
36     num=1;
37     den=poly([-sigma+%i*w, -sigma-%i*w], 's');
38     g=syslin('c',num./den);
39     gz=dscr(g,Ts)
40     plzr(gz);
41 end
42 zgrid()
43 f=get("current_figure") //Current figure handle
44 f.background=8

```





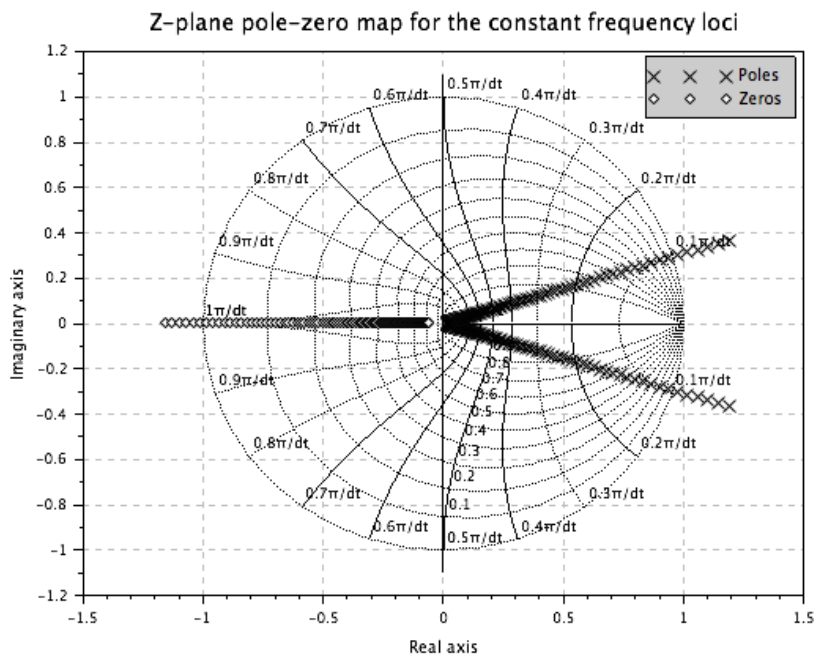


Figure 5.2: Lab05

# Experiment: 6

## Map constant damping ratio loci from s-plane to z-plane.

Scilab code Solution 6.06 Lab06

```
1 //Lab. 06: Map constant damping ratio loci from s-
   plane to z-plane
2
3 //           scilab - 5.5.1
4 //           Operating System : Windows 7, 32-bit
5 //
   *****

6 //Clean the environment
7 close;
8 clear;
9 clc;
10
11 // System model
12 s=poly(0, 's');
13 Ts=0.2;
14 //
```

```

*****

15 // S-plane pole-zero map for the continuous time
    system
16 //
    *****

17 for w=0:0.2:19
18     sigma=w;
19     num=1;
20     den=poly([-sigma+%i*w, -sigma-%i*w], 's');
21     g=syslin('c',num./den);
22     plzr(g)
23 end
24 sgrid()
25 title('S-plane pole-zero map for the constant
        damping ratio loci','fontsize',3)
26
27 // Zoom axes for clarity
28
29 zoom_rect([-20 -22 20 22])
30
31 //
    *****

32 //Z-plane pole-zero map for the sample data system
33 //
    *****

34 figure;
35 for w=0:0.2:19
36     sigma=w;
37     num=1;
38     den=poly([-sigma+%i*w, -sigma-%i*w], 's');
39     g=syslin('c',num./den);
40     gz=dscr(g,Ts)
41     plzr(gz);
42 end

```

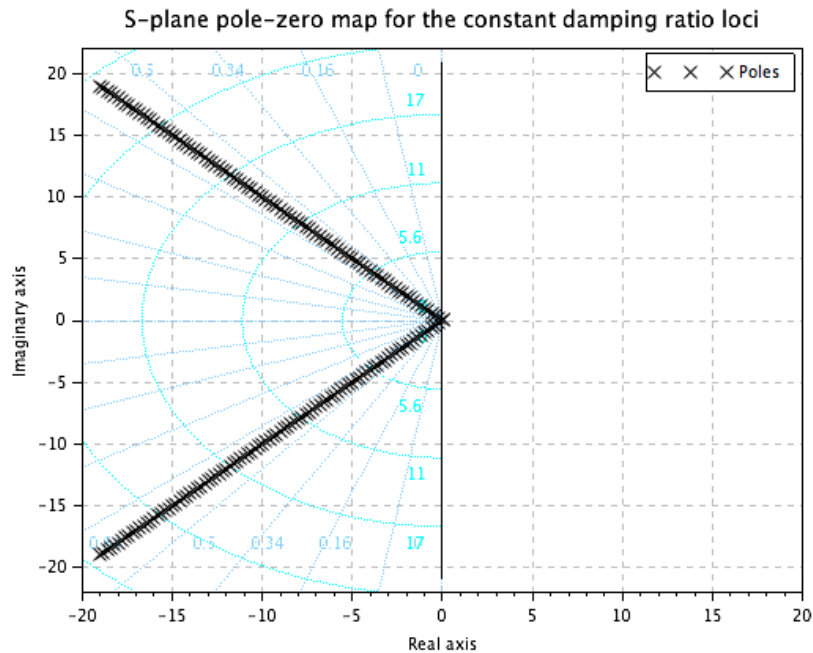


Figure 6.1: Lab06

```

43 zgrid();
44 f=get("current_figure") //Current figure handle
45 f.background=8
46 title('Z-plane pole-zero map for the constant
         damping ratio loci','fontsize',3)
47
48 // Zoom axes for clarity
49 zoom_rect([-1.2 -1.2 1.2 1.2])

```

---

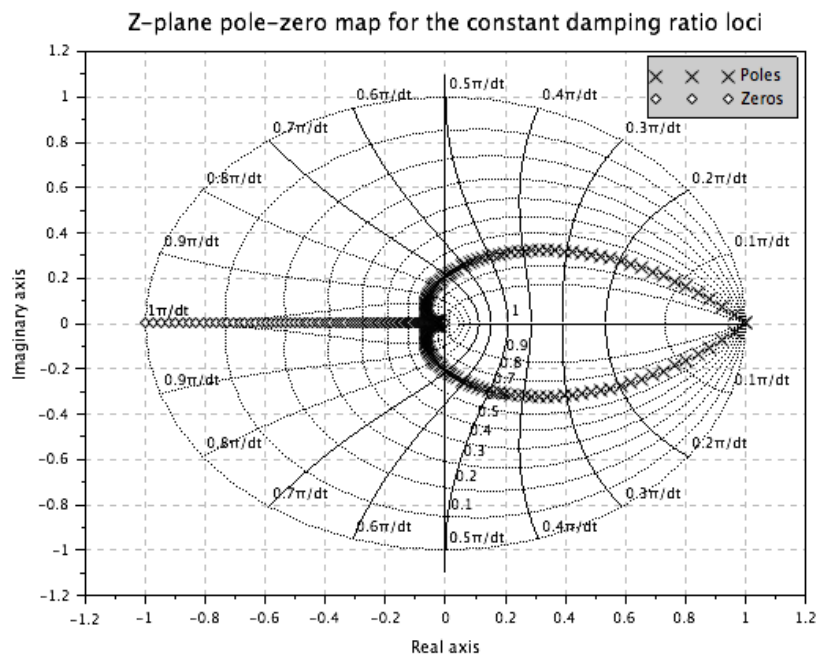


Figure 6.2: Lab06

# Experiment: 7

## Transform the discrete-time state model into canonical forms.

Scilab code Solution 7.07 Lab07

```
1 //Lab. 07: Transform the discrete-time state model
   into canonical forms.
2
3 //           scilab - 5.5.1
4 //           Operating System : Windows 7, 32-bit
5 //
   *****

6 //Clean the environment
7 close;
8 clear;
9 clc;
10
11 // State space model
12 A=[0 -0.4; 1 -1.3];
13 B=[0;1];
14 C=[0 1];
```

```

15 D=0;
16
17 sys=syslin('d',A,B,C,D)
18 mprintf('State space representation of the given
    discrete system is ')
19 disp(sys)
20
21 // Transfer function model
22 systf=ss2tf(sys)
23 mprintf('Transfer function of the given discrete
    system is ')
24 disp(systf)
25
26
27 // Eigen values of system matrix
28 eig_val=spec(A)
29 mprintf('Eigen values of the system matrix are ')
30 disp(eig_val)
31
32 // Controllable canonical form
33 [Phi , Gamma ,T]=canon(A,B)
34 T=flipdim(T,2);
35 Phi=T\A*T;
36 Gamma=T\B;
37 C=C*T;
38 D=D;
39 sysd=syslin('d',Phi ,Gamma ,C,D)
40 mprintf('State space representation of the given
    discrete system ')
41 disp('in controllable canonical form is ')
42 disp(sysd)
43
44 // Diagonal form
45 [Phid M]=bdiag(A);
46 Gammad=M\B;
47 Cd=C*M;
48 Dd=D;
49 sysd=syslin('d',Phid ,Gammad ,Cd ,Dd)

```



```
50 mprintf('State space representation of the given')
51 disp('discrete system in diagonal form is')
52 disp(sysd)
```

---

# Experiment: 8

## Check controllability & observability of a given system.

Scilab code Solution 8.08 Lab08

```
1 //Lab. 08: Check controllability & observability of
  a given system.
2
3 //          scilab - 5.5.1
4 //          Operating System : Windows 7, 32-bit
5 //
  *****

6 //Clean the environment
7 close;
8 clear;
9 clc;
10
11 // State space representation
12 A=[-0.2 0;0 -0.8];
13 B=[1 1]';
14 C=[4/3 -1/3];
15 D=0;
16 sys=syslin('d',A,B,C,D)
```

```

17
18 // Controllability test
19
20 n=cont_mat(sys)
21 mprintf('Controllability matrix is ')
22 disp(n)
23
24 if rank(n)==2 then
25     disp('Rank of controllability matrix is full,')
26     disp('therefore system is controllable')
27 else
28     disp('Rank of controllability matrix is not full,')
29     disp('therefore system is uncontrollable')
30 end
31
32 disp('')
33
34 // Observability test
35 m=obsv_mat(sys)
36 mprintf(' Observability matrix is ')
37 disp(m)
38
39 if rank(m)==2 then
40     disp('Rank of observability matrix is full,')
41     disp('therefore system is observable')
42 else
43     disp('Rank of observability matrix is not full,')
44     disp('therefore system is unobservable')
45 end

```

---

# Experiment: 9

Design state feedback controller for a given system to achieve desired dynamic characteristics.

Scilab code Solution 9.09 Lab09

```
1 //  
-----  
2 // Lab. 09: Design state feedback controller for a  
   given system to achieve desired dynamic  
   characteristics.  
3 //           scilab - 5.5.1  
4 //           Operating System : Windows 7, 32-bit  
5 //  
   *****  
6 //Clean the environment  
7 close;
```

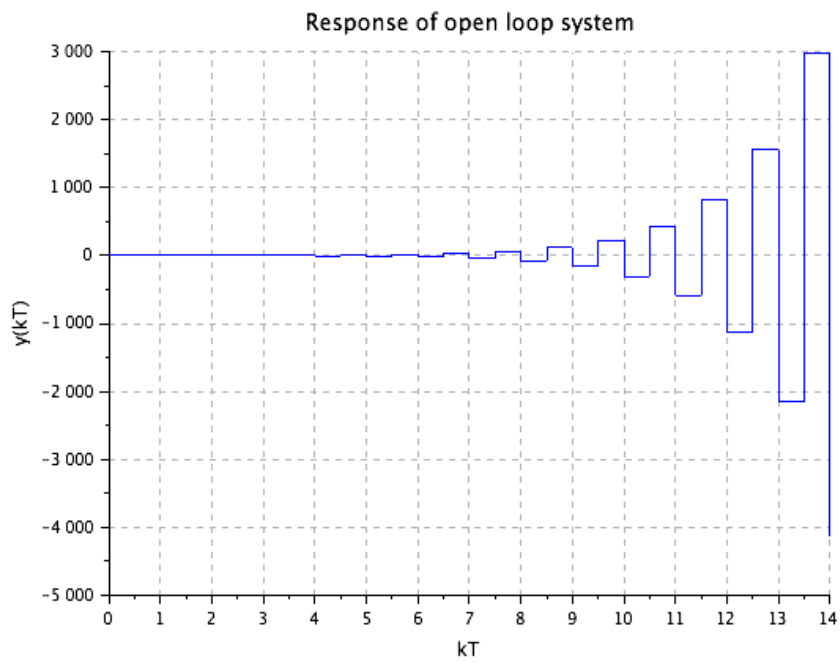


Figure 9.1: Lab09

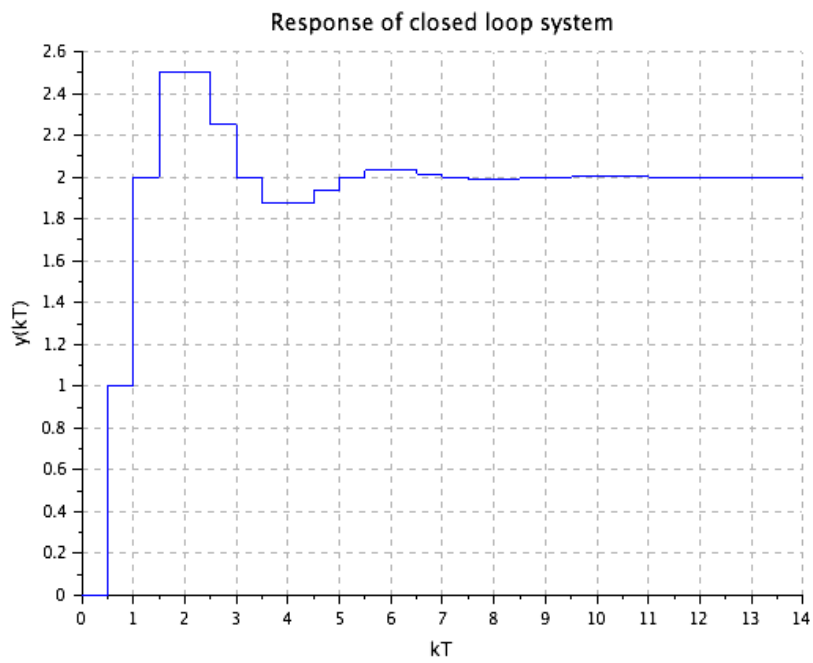


Figure 9.2: Lab09

```

8  clear;
9  clc;
10
11 // State space representation
12 A=[0 1;-0.16 -1.5];
13 B=[0 1]';
14 C=[0 1];
15 D=0;
16
17 sys1=syslin('d',A,B,C,D)
18
19 // Desired poles
20 Pd=[0.5+0.5*%i 0.5-0.5*%i];
21
22 // State feedback gain matrix
23 K=ppol(A,B,Pd)
24 mprintf('State feedback gain matrix is K=')
25 disp(K)
26
27 //Closed loop system
28 sys=syslin('d',A-B*K,B,C,D)
29
30 // Sampling Time
31 Ts=0.5;
32 t=0:Ts:14;
33 u=ones(1,length(t));
34
35 //Response of open loop system
36 y1=flts(u,sys1);
37 plot2d2(t,y1,2)
38 xgrid(35)
39 title('Response of open loop system','fontsize',3)
40 xlabel('kT','fontsize',2)
41 ylabel('y(kT)','fontsize',2)
42
43 //Response of closed loop system
44 y=flts(u,sys);
45 figure,

```

```
46 plot2d2(t,y,2)
47 f=get("current_figure") //Current figure handle
48 f.background=8
49 xgrid(36)
50 title('Response of closed loop system','fontsize',3)
51 xlabel('kT','fontsize',2)
52 ylabel('y(kT)','fontsize',2)
```

---



# Experiment: 10

Design dead beat controller for a given system.

Scilab code Solution 10.10 Lab10

```
1 //  
-----  
2 // Lab.10: Design dead beat controller for a given  
   system.  
3 //           scilab – 5.5.1  
4 //           Operating System : Windows 7, 32-bit  
5 //  
   *****  
6 //Clean the environment  
7 close;  
8 clear;  
9 clc;  
10
```

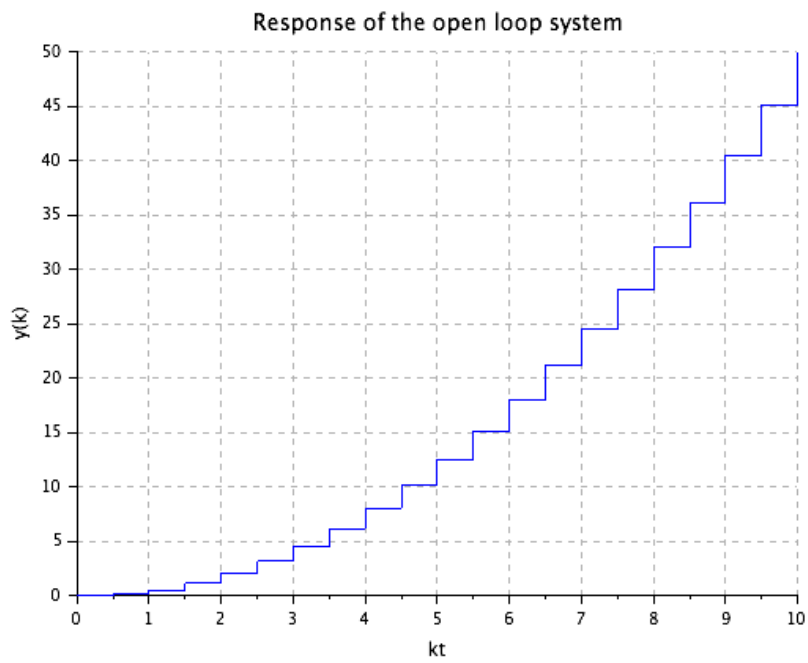


Figure 10.1: Lab10

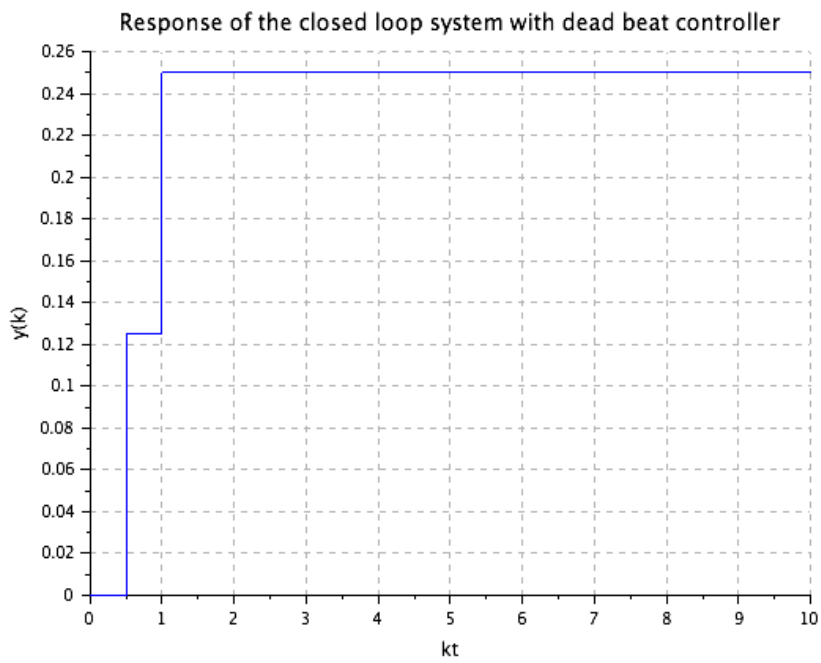


Figure 10.2: Lab10

```

11 //Sampling time
12 Ts=0.5;
13
14 // State space representation of the continuous time
    system
15 A=[0 1;0 0];
16 B=[0 1]';
17 C=[1 0];
18 D=0;
19
20 sys1=syslin('c',A,B,C,D)
21
22 //Discrete model of the system
23
24 sys=dscr(sys1,Ts)
25 mprintf('Discrete model of the system is sys=')
26 disp(sys)
27
28 // Desired poles
29 Pd=[0 0];
30
31 // State feedback gain matrix
32 K=ppol(sys.A,sys.B,Pd)
33 mprintf('State feedback gain matrix is K=')
34 disp(K)
35
36 //Responses
37 t=0:Ts:10;
38 u=ones(1,length(t));
39
40 //Response of open loop system
41 y1=flts(u,sys);
42 plot2d2(t,y1,2)
43 xgrid(35)
44 title('Response of the open loop system','fontsize'
    ,3)
45 xlabel('kt','fontsize',2)
46 ylabel('y(k)','fontsize',2)

```

```
47
48
49 //Response of closed loop system
50 syscl=syslin('d',sys.A-sys.B*K,sys.B,sys.C,0)
51 y=flts(u,syscl);
52 figure,
53 plot2d2(t,y,2)
54 f=get("current_figure") //Current figure handle
55 f.background=8
56 xgrid(36)
57 title('Response of the closed loop system with dead
        beat controller','fontsize',3)
58 xlabel('kt','fontsize',2)
59 ylabel('y(k)','fontsize',2)
```

---

# Experiment: 11

## Design full state observer for a given system.

Scilab code Solution 11.11 Lab11

```
1 //  


---

  
2 // Lab. 11: Design a full state observer for the  
  system.  
3 //           scilab – 5.5.1  
4 //           Operating System : Windows 7, 32-bit  
5 //  
  *****  
  
6 //Clean the environment  
7 close;  
8 clear;  
9 clc;  
10  
11 //State space model  
12 A=[0 1 0; 0 0 1; 0.6 0.7 0.2 ];  
13 B=[0 0 1]';  
14 C=[1 0 0];
```

```

15 D=0;
16
17 // Stabilizer design
18 // Desired poles
19 Pd=[0.2, -0.5+0.5*i, -0.5-0.5*i];
20
21 // State feedback gain matrix
22 K=ppol(A,B,Pd)
23
24 // Computation of gain for dead beat observer
    response
25 obsr_pol=[0 0 0];
26 L=ppol(A',C',obsr_pol)
27
28 // Augmented system
29 temp=size(A);
30 Aa=[A-B*K    B*K; zeros(temp(1),temp(2))    A-L*C
    ];
31 temp=size(Aa);
32 Ba=zeros(temp(1),1);
33 Ca=eye(6,6);
34 sys=syslin('d',Aa,Ba,Ca,zeros(6,1))
35
36 // Observer error
37 t=0:0.2:4.5;
38 u=zeros(1,length(t));
39 //y=flts(u,sys);
40 x=ltitr(Aa,Ba,u,[0.1 0.1 0.1 -0.5 -0.5 -0.5]')
41
42 temp=size(Aa);
43 temp=temp(1,1)/2;
44
45 col=[1 2 5]; //specifying plot colors
46 for i=1:temp
47     subplot(2,1,1), plot2d2(t,x(i,:),col(1,i))
48     if i==temp then
49         title('States of the system','fontsize',2);
50         xlabel('$t$', 'fontsize',2);

```

```

51         ylabel('$x(kT)$', 'fontsize', 2);
52         h1=legend('$x_1(kT)$', '$x_2(kT)$', '$x_3(kT)$
           ');
53         h1.legend_location = "in_lower_right";
54         xgrid(35)
55     end
56     subplot(2,1,2), plot2d2(t,x(i+3,:),col(1,i))
57     if i==temp then
58         title('Observer error', 'fontsize', 2)
59         xlabel('$t$', 'fontsize', 2)
60         ylabel('$e(kT)=x(kT)-\hat{x}(kT)$', 'fontsize',
           2)
61         h2=legend('$e_1(kT)$', '$e_2(kT)$', '$e_3(kT)$
           ')
62         h2.legend_location = "in_lower_right"
63         xgrid(35)
64     end
65 end

```

---



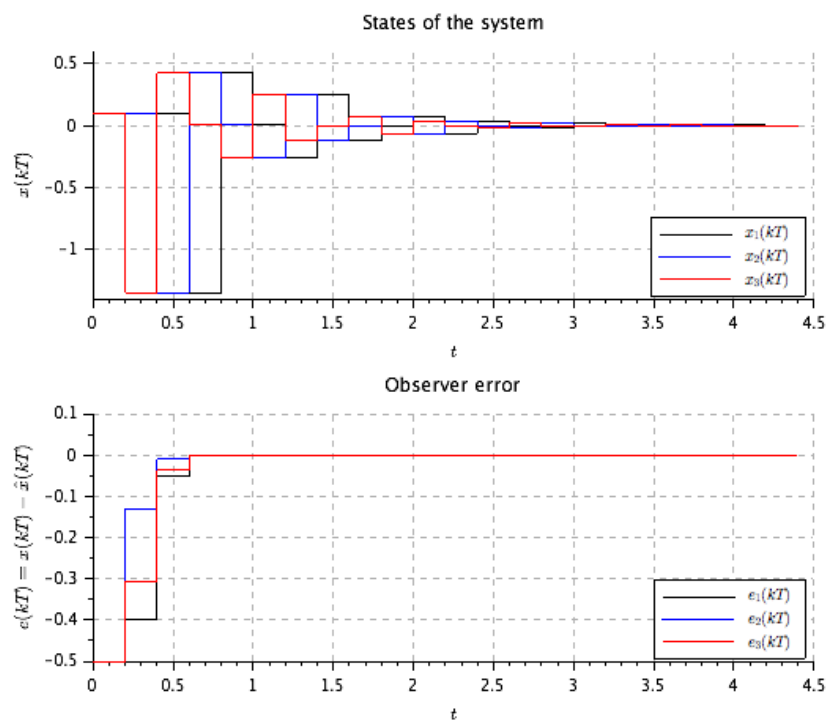


Figure 11.1: Lab11

## Experiment: 12

Determine stability of a given system by using Lyapunov stability analysis.

Scilab code Solution 12.12 Lab12

```
1 //Lab. 12: Determine stability of a given system by
  using Lyapunov stability analysis
2 //          scilab - 5.5.1
3 //          Operating System : Windows 7, 32-bit
4 //
  *****

5 //Clean the environment
6 close;
7 clear;
8 clc;
9
10 //System matrix
11 a=[0 1; -0.5 -1];
12 q=-eye(2,2);
13 p=lyap(a,q,'d');
14
```

```
15 // For a stable system matrix, p should be positive
    definite for
16 //which all the principle minors or all eigen values
    of the matrix p
17 // should be positive
18
19 eig_val=spec(p);
20 m=length(eig_val);
21 stable=0;
22 for i=1:m;
23     if real(eig_val(i))>0 then
24         stable=stable+1;
25     end
26 end
27 if stable==m then
28     disp('The system is asymptotically stable')
29 else
30     disp('The system is unstable or critically
        stable')
31 end
```

---

# Experiment: 13

## Construct root loci and bode plots for a given discrete-time system.

Scilab code Solution 13.13 Lab13

```
1 //Lab. 13: Construct root loci and bode plots for a
   given discrete-time system.
2
3 //          scilab - 5.5.1
4 //          Operating System : Windows 7, 32-bit
5 //
   *****
6 //Clean the environment
7 close;
8 clear;
9 clc;
10
11 //Sampling time
12 Ts=0.4;
13
14 //System transfer function.
```

```
15 s=poly(0, 's');
16 g=1/(s*(s+1));
17 g=syslin('c',g);
18 gz=dscr(g,Ts)//discrete model
19
20 // Root locus for the discrete system
21 evans(gz,100)
22 title('Root locus of the discrete-time system')
23 zgrid(0:0.1:0.5)
24 zoom_rect([-1.2 -1.5 1.2 1.5])
25 figure
26
27 // Bode plot for the discrete system
28 bode(gz)
29 f=get("current_figure") //Current figure handle
30 f.background=8
31 title('Bode plot of discrete-time system')
```

---

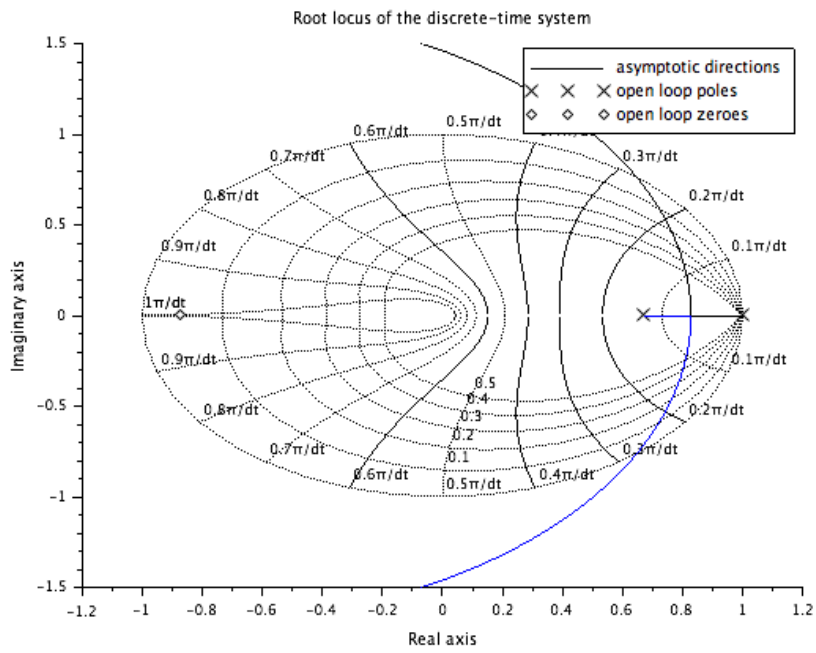


Figure 13.1: Lab13

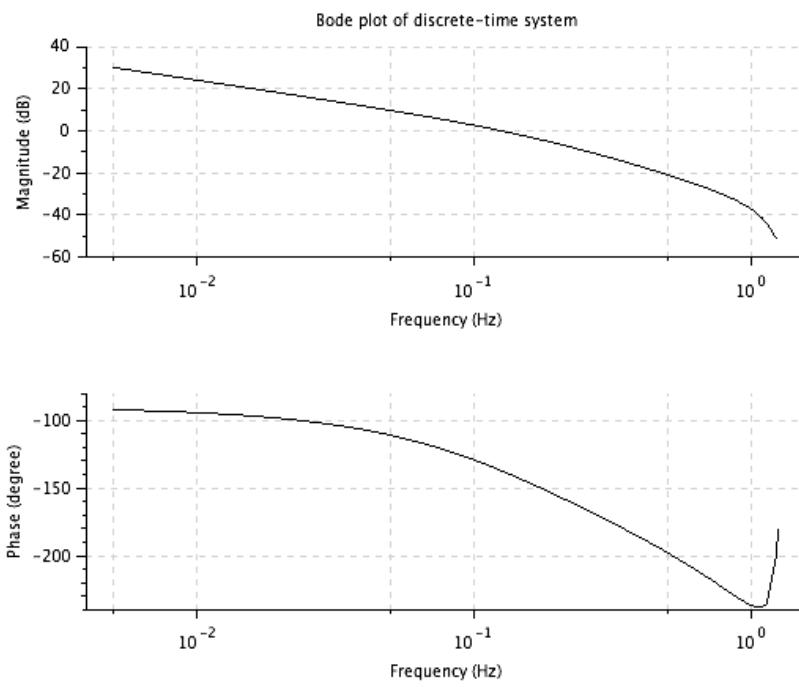


Figure 13.2: Lab13