Scilab Textbook Companion for Modern Control Engineering by K. Ogata[1]

Created by
Brian Coutinho
Control Engineering
Electrical Engineering
IIT Jodhpur
College Teacher
Dr. Swagat Kumar
Cross-Checked by
Aditya Sengupta, IIT Bombay

July 17, 2017

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Modern Control Engineering

Author: K. Ogata

Publisher: Princton Hall Of India Private Limited, New Delhi

Edition: 5

Year: 2010

ISBN: 978-81-203-4010-7

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes 4

2 Mathematical Modelling of Control Systems 6

5 Transient and Steady State Response Analysis 18

6 Control Systems Analysis and Design by Root Locus Method 57

7 Control Systems Analysis and Design by Frequency Response Method 127

8 PID Controllers and Modified PID Controllers 198

9 Control Systems Analysis in State Space 237

10 Control Systems Design in State Space 248

List of Scilab Codes

Exa 2.i.1 Series Parallel Feedback connection of Systems 6
Exa 2.i.2 Transfer Function to State Space Model . . 7
Exa 2.b.4 Step and Ramp response of different Controllers 8
Exa 2.a.7 Transfer Function to Controllable State Space form 11
Exa 2.a.11 State space to Transfer Function model SISO system 11
Exa 2.a.12 State space to Transfer Function model MIMO system 12
Exa 2.b.14 Verifying linearization of a non linear system 13
Exa 2.4 Convert State space to Transfer Function model 14
Exa 5.a.3 Verifying design to match given response curve 18
Exa 5.a.4 Determining K and k for required step response 20
Exa 5.a.5 Verifying design to match given response . . 20
Exa 5.a.8 Unit step response and partial fraction expansion 21
Exa 5.a.9 Effect of zeros on step response of a system 24
Exa 5.a.10 Step response characteristics 25
Exa 5.a.11 Step Response for different zeta and wn . . 26
Exa 5.a.12 Response to unit ramp and exponential input 28
Exa 5.a.13 Response to input r equals 2 plus t 30
Exa 5.a.14 Response to unit acceleration input 32
Exa 5.a.15 Step Responses for different zeta 34
Exa 5.a.16 Response to initial conditions 35
Exa 5.2 Determining K and Kh for required step response 36

Exa 5.3	Step response of MIMO system	38
Exa 5.4	Second order systems with different damping ratio	39
Exa 5.5	Impulse Response of a Second order System	42
Exa 5.6	Unit Ramp response of a second order system	45
Exa 5.7	Response to step and exponential input . . .	47
Exa 5.8	Response to initial condition	48
Exa 5.9	Response to initial conditions using state space	51
Exa 5.10	Response to initial condition using syslin x0	53
Exa 5.12	Constructing Routh array	54
Exa 5.13	Constructing Routh array	56
Exa 6.i.1	Finding the Gain K at any point on the root locus	57
Exa 6.i.2	Orthogonality Constant gain curves and Root Locus	59
Exa 6.i.3	Effect of adding poles or zeros on the root locus	61
Exa 6.a.6	Root locus	64
Exa 6.a.13.1	Lead Compensator Design Attempt 1	65
Exa 6.a.13.2	Lead Compensator Design Attempt 2	69
Exa 6.a.17	Design of lag lead compensator	73
Exa 6.a.18	Design of a compensator for a highly oscillactory system	77
Exa 6.1	Root Locus	81
Exa 6.2	Root Locus	81
Exa 6.3	Root Locus	83
Exa 6.4	Root Locus	86
Exa 6.5	Root locus of system in state space	86
Exa 6.6.1	Design of a lead compensator using root locus	88
Exa 6.6.2	Step and ramp response of lead compensated systems	91
Exa 6.7.1	Design of a lag compensator using root locus	94
Exa 6.7.2	Step and ramp response of lag compensated system	99
Exa 6.8.1	Design of a lag lead compensator using root locus	100
Exa 6.8.2	Evaluating Lag Lead compensated system .	103

Exa 6.9.1	Design of lag lead compensator using root locus 2	107
Exa 6.9.2	Evaluating Lag Lead compensated system	109
Exa 6.10	Design of parallel compensation by root locus	115
Exa 6.15	Design of lag compensator	119
Exa 6.16	Design of lag lead compensator	123
Exa 7.a.1	Bode plot	127
Exa 7.i.1	Bode plot for 2nd order systems with varying zeta	130
Exa 7.a.3	Bode plot for system in state space	131
Exa 7.a.4	Bode plot for different gain K	131
Exa 7.a.8	Stability check	133
Exa 7.a.10	Nyquist Plot with transport lag	135
Exa 7.a.11	Nyquist Plot	135
Exa 7.a.12	Nyquist plot for positive omega	137
Exa 7.a.13	Nyquist plot with points at selected frequencies	138
Exa 7.a.14	Nyquist plot for positive and negative feedback	140
Exa 7.a.18	Verifying experimentally derived Transfer function	142
Exa 7.a.23	Nichols plot	143
Exa 7.1	Steady state sinusoidal output	144
Exa 7.2	Steady state sinusoidal output lag and lead	146
Exa 7.3	Bode Plot in Hz	147
Exa 7.4	Bode Plot with transport lag	150
Exa 7.5	Bode Plot in rad per s	150
Exa 7.6	Bode plot in rad per s	152
Exa 7.7	Bode Plot for a system in State Space	155
Exa 7.8	Polar Plot of a linear system	156
Exa 7.9	Polar Plot with transport lag	158
Exa 7.10	Nyquist Plot	159
Exa 7.11	Nyquist Plot	161
Exa 7.12	Nyquist Plots of system in state space	162
Exa 7.13	Nyquist Plot of MIMO system	164
Exa 7.14	Nyquist Stability Check	165
Exa 7.19	Nyquist plot stability check	167
Exa 7.20	Gain and phase margins for different K	168
Exa 7.21	Stability Margins	172

Exa 7.22	Correlating bandwidth and speed of response	172
Exa 7.23	Frequency charecteristics	177
Exa 7.24	Polar and Nichols plot with M circles	177
Exa 7.25	Verifying experimentally derived Transfer function	179
Exa 7.26.1	Design of Lead compensator with Bode plots	182
Exa 7.26.2	Evaluating Lead compensated system	186
Exa 7.27.1	Design of Lag compensator with Bode plots	187
Exa 7.27.2	Evaluating Lag compensated system	191
Exa 7.28.1	Design of Lag lead compensation with Bode plots	192
Exa 7.28.2	Evaluating Lag Lead compensated system	194
Exa 8.i.1	PID Design with Frequency Response	198
Exa 8.a.5	PID design	202
Exa 8.a.6	PID design	203
Exa 8.a.7.1	PID Design with Frequency Response	209
Exa 8.a.12	Computing optimal solution	210
Exa 8.a.13	Design of system with two degrees of freedom	214
Exa 8.1	Tuning a PID controller using Nichols Second Rule	219
Exa 8.2	Computation of Optimal solution 1	223
Exa 8.3	Computation of Optimal solution 2	224
Exa 8.4	Design of system with two degrees of freedom	227
Exa 8.5	Design of system with two degrees of freedom 2	231
Exa 9.b.3	Obtaining canonical form	237
Exa 9.a.5	Conversion from transfer function model to state space model	238
Exa 9.a.16	Controllability and pole zero cancellation	238
Exa 9.a.17	Controllability observability and pole zero cancellation	239
Exa 9.1	Transfer function to controllable observable and jordon canonical forms	240
Exa 9.2	Transformations in state space	241
Exa 9.3	Conversion from state space to transfer function model	242
Exa 9.4	Conversion from state space to transfer function model	243

Exa 9.5	State transition matrix	244
Exa 9.7	Finding e to the power At using laplace transforms	245
Exa 9.9	Linear dependence of vectors	246
Exa 9.14	State and ouput controllability and observability	246
Exa 9.15	Observability	247
Exa 10.i.1	Designing a regulator using a minimum order observer	248
Exa 10.i.2	Designing a control system with a minimum order observer	252
Exa 10.a.5	Feedback gain for moving eigen values	256
Exa 10.a.6	Gain matrix determination	257
Exa 10.a.9	Transforming to canonical form	257
Exa 10.a.13	Designing a regulator using a minimum order observer	258
Exa 10.a.14	Designing a regulator using a minimum and full order observer	260
Exa 10.a.17	Design of quadratic optimal regulator system and finding the response	265
Exa 10.1	Gain matrix using characteristic eq and Ackermanns formula	267
Exa 10.2	Gain matrix using ppol and Ackermanns formula	268
Exa 10.3	Response to initial condition	268
Exa 10.4	Design of servo system with integrator in the plant	269
Exa 10.5	Design of servo system without integrator in the plant	271
Exa 10.6	Observer Gain matrix using ch eq and Ackermanns formula	275
Exa 10.7	Designing a controller using a full order observer	276
Exa 10.8	Designing a controller using a minimum order observer	278
Exa 10.9	Design of quadratic optimal regulator system	279
Exa 10.10	Design of quadratic optimal regulator system	280
Exa 10.11	Design of quadratic optimal regulator system	281

Exa 10.12 Design of quadratic optimal regulator system and finding the response 281
Exa 10.13 Design of quadratic optimal regulator system and finding the response 282
AP 1 Determine Gains and transfer function for minimal order observer 286
AP 2 Plot System Response 287
AP 3 Compute the feedback gain matrix using ackermanns formula 287
AP 4 Transfer function of A,B,C,D. 288
AP 5 Inverse Laplace transform of a rational polynomial in s 288
AP 6 Partial Fraction Residue 289
AP 7 Plot the root locus in a box 290
AP 8 Step response characteristics 290
AP 9 Polar plot of a linear system 291
AP 10 Display gain and phase margins 292
AP 11 Frequency response characteristics 292
AP 12 Gain at a point on a root locus 293

List of Figures

2.1 Step and Ramp response of different Controllers 10
2.2 Verifying linearization of a non linear system 15
2.3 Verifying linearization of a non linear system 16

5.1 Verifying design to match given response curve 19
5.2 Verifying design to match given response 22
5.3 Unit step response and partial fraction expansion 23
5.4 Effect of zeros on step response of a system 25
5.5 Step response characteristics 27
5.6 Step Response for different zeta and wn 29
5.7 Response to unit ramp and exponential input 30
5.8 Response to unit ramp and exponential input 31
5.9 Response to input r equals 2 plus t 33
5.10 Response to unit acceleration input 34
5.11 Step Responses for different zeta 35
5.12 Response to initial conditions 37
5.13 Step response of MIMO system 40
5.14 Step response of MIMO system 41
5.15 Second order systems with different damping ratio 43
5.16 Second order systems with different damping ratio 44
5.17 Impulse Response of a Second order System 46
5.18 Unit Ramp response of a second order system 47
5.19 Response to step and exponential input 49

5.20 Response to step and exponential input 50
5.21 Response to initial condition 52
5.22 Response to initial conditions using state space 53
5.23 Response to initial condition using syslin x0 55

6.1 Finding the Gain K at any point on the root locus 58
6.2 Orthogonality Constant gain curves and Root Locus 60
6.3 Effect of adding poles or zeros on the root locus 62
6.4 Root locus 65
6.5 Lead Compensator Design Attempt 1 67
6.6 Lead Compensator Design Attempt 1 68
6.7 Lead Compensator Design Attempt 2 71
6.8 Lead Compensator Design Attempt 2 72
6.9 Design of lag lead compensator 75
6.10 Design of lag lead compensator 76
6.11 Design of a compensator for a highly oscillactory system . . 79
6.12 Design of a compensator for a highly oscillactory system . . 80
6.13 Root Locus 82
6.14 Root Locus 84
6.15 Root Locus 85
6.16 Root Locus 87
6.17 Root locus of system in state space 89
6.18 Design of a lead compensator using root locus 92
6.19 Design of a lead compensator using root locus 93
6.20 Step and ramp response of lead compensated systems 95
6.21 Step and ramp response of lead compensated systems 96
6.22 Design of a lag compensator using root locus 98
6.23 Step and ramp response of lag compensated system 100
6.24 Design of a lag lead compensator using root locus 103
6.25 Design of a lag lead compensator using root locus 104
6.26 Evaluating Lag Lead compensated system 106
6.27 Evaluating Lag Lead compensated system 107
6.28 Design of lag lead compensator using root locus 2 110
6.29 Design of lag lead compensator using root locus 2 111
6.30 Evaluating Lag Lead compensated system 113
6.31 Evaluating Lag Lead compensated system 114
6.32 Design of parallel compensation by root locus 117
6.33 Design of parallel compensation by root locus 118

6.34 Design of lag compensator 121
6.35 Design of lag compensator 122
6.36 Design of lag lead compensator 125
6.37 Design of lag lead compensator 126

7.1 Bode plot 128
7.2 Bode plot for 2nd order systems with varying zeta 129
7.3 Bode plot for system in state space 132
7.4 Bode plot for different gain K 134
7.5 Nyquist Plot with transport lag 136
7.6 Nyquist Plot 137
7.7 Nyquist plot for positive omega 139
7.8 Nyquist plot with points at selected frequencies 141
7.9 Nyquist plot for positive and negative feedback 142
7.10 Verifying experimentally derived Transfer function 143
7.11 Nichols plot 145
7.12 Steady state sinusoidal output 146
7.13 Steady state sinusoidal output lag and lead 148
7.14 Bode Plot in Hz 149
7.15 Bode Plot with transport lag 151
7.16 Bode Plot in rad per s 153
7.17 Bode Plot in rad per s 154
7.18 Bode plot in rad per s 155
7.19 Bode Plot for a system in State Space 157
7.20 Polar Plot of a linear system 158
7.21 Polar Plot with transport lag 160
7.22 Nyquist Plot 161
7.23 Nyquist Plot 163
7.24 Nyquist Plots of system in state space 164
7.25 Nyquist Plot of MIMO system 166
7.26 Nyquist Stability Check 167
7.27 Nyquist plot stability check 169
7.28 Gain and phase margins for different K 170
7.29 Gain and phase margins for different K 171
7.30 Stability Margins 173
7.31 Correlating bandwidth and speed of response 175
7.32 Correlating bandwidth and speed of response 176
7.33 Frequency charecteristics 178

7.34 Polar and Nichols plot with M circles 180
7.35 Polar and Nichols plot with M circles 181
7.36 Verifying experimentally derived Transfer function 182
7.37 Design of Lead compensator with Bode plots 184
7.38 Design of Lead compensator with Bode plots 185
7.39 Evaluating Lead compensated system 187
7.40 Design of Lag compensator with Bode plots 189
7.41 Design of Lag compensator with Bode plots 190
7.42 Evaluating Lag compensated system 192
7.43 Design of Lag lead compensation with Bode plots 194
7.44 Design of Lag lead compensation with Bode plots 195
7.45 Evaluating Lag Lead compensated system 197

8.1 PID Design with Frequency Response 199
8.2 PID Design with Frequency Response 200
8.3 PID design 204
8.4 PID design 205
8.5 PID design 207
8.6 PID design 208
8.7 PID Design with Frequency Response 211
8.8 PID Design with Frequency Response 212
8.9 Computing optimal solution 215
8.10 Design of system with two degrees of freedom 217
8.11 Design of system with two degrees of freedom 218
8.12 Tuning a PID controller using Nichols Second Rule 221
8.13 Tuning a PID controller using Nichols Second Rule 222
8.14 Computation of Optimal solution 1 225
8.15 Computation of Optimal solution 2 228
8.16 Design of system with two degrees of freedom 232
8.17 Design of system with two degrees of freedom 233
8.18 Design of system with two degrees of freedom 2 235
8.19 Design of system with two degrees of freedom 2 236

10.1 Designing a regulator using a minimum order observer . . . 249
10.2 Designing a regulator using a minimum order observer . . . 250
10.3 Designing a control system with a minimum order observer . 253
10.4 Designing a control system with a minimum order observer . 254
10.5 Designing a regulator using a minimum order observer . . . 260

10.6 Designing a regulator using a minimum and full order observer 263
10.7 Designing a regulator using a minimum and full order observer 264
10.8 Design of quadratic optimal regulator system and finding the response 266
10.9 Response to initial condition 270
10.10 Design of servo system with integrator in the plant 272
10.11 Design of servo system without integrator in the plant 274
10.12 Designing a controller using a full order observer 278
10.13 Design of quadratic optimal regulator system and finding the response 283
10.14 Design of quadratic optimal regulator system and finding the response 285

Chapter 2

Mathematical Modelling of Control Systems

Scilab code Exa 2.i.1 Series Parallel Feedback connection of Systems

```
// Illustration 2.1
// Section 2-3 in the book
// Demonstrating Series,Parallel and feedback connection of Linear Systems

clear; clc; close;

// Define Polynomials in variable 's'
// Please NOTE : The list of coeficients has to be given in
//                  INCREASING  powers of 's',

n1 = poly( [10] ,'s','c');
d1 = poly( [10 2 1] ,'s','c'); // 10 + 2*s + s^2

// Alternate method to define transfer functions in scilab
// using '%s'
s = %s;
```

```
n2 = 5;
d2 = 5 + s;


G1 = syslin('c',n1,d1);     //define continuous LTI systems systems
G2 = syslin('c',n2,d2);

disp(G1,'G1 =');disp(G2,'G2 ='); //display variables on the screen

series   = G1 * G2;
parallel = G1 + G2;
feedback = G1 /. G2 ;  // feedback is via G2.

disp(series,'series =');
disp(parallel,'parallel =');
disp(feedback,'feedback =');
```

Scilab code Exa 2.i.2 Transfer Function to State Space Model

```
// Illustration 2.2
// Conversion from transfer function model to state space model
// Section 2-6 of the Book

//     This example demonstrates that there is no unique
//  state space reperesentation of a transfer function.

clear; clc; close; mode(0);
s = %s;
num = s;
den = 160 + 56*s + 14*s^2 + s^3;
```

```
Htf = syslin('c',num,den)

// There are infinite state space models for the
   same transfer
// function. The tf2ss() function will return one of
    them,

Hss = tf2ss(Htf);
ssprint(Hss);        //Print the state space model

//Alternatively: you can directly get the A,B,C,D
[A,B,C,D] = abcd(Htf)

//To cross check, let us find the transfer function
Htf2 = clean(ss2tf(Hss))    //which matches with Htf



Hssc = cont_frm(Htf.num,Htf.den)
Htfc = clean(ss2tf(Hssc))

// The same transfer function again
```

Scilab code Exa 2.b.4 Step and Ramp response of different Controllers

```
// Exercise B-2-4
// Plotting the response of different types of
   controllers
// to unit step and unit ramp input.

clear; clc; xdel(winsid());

Kp = 4;     //proportional gain
Ki1 = 2;    //integral gain
Td = 0.8;   //differential time
```

```
Ti = 2;     //integral time
Ki2 = Kp / Ti;

s = %s;
Gi = syslin('c',Ki1/s);

t = 0:0.05:3;
ramp = t;
subplot(3,2,1);
p1 = Kp * ones(1,length(t));
p2 = Kp * t;
plot2d(t ,p1 , style=2);
plot2d(t ,p2 , style=3);
xtitle('Proportional control','t','y');
legend('step input','ramp input');
xgrid(color('gray'));

subplot(3,2,2);
i1 =  csim("step",t,Gi);
i2 =  csim(ramp,t,Gi);
plot2d(t ,i1, style=2);
plot2d(t ,i2, style=3) ;
xtitle('Integral control','t','y');
xgrid(color('gray'));
i1 = i1 * Ki2 / Ki1; //change of gain
i2 = i2 * Ki2 / Ki1;


subplot(3,2,3);
plot2d(t ,p1 + i1, style=2);
plot2d(t ,p2 + i2, style=3);
xtitle('Proportional integral control','t','y');
xgrid(color('gray'));

subplot(3,2,4);
pd1 = p1;
pd2 = p2 + Kp*Td*ones(1,length(t)); //derivative
   term
```

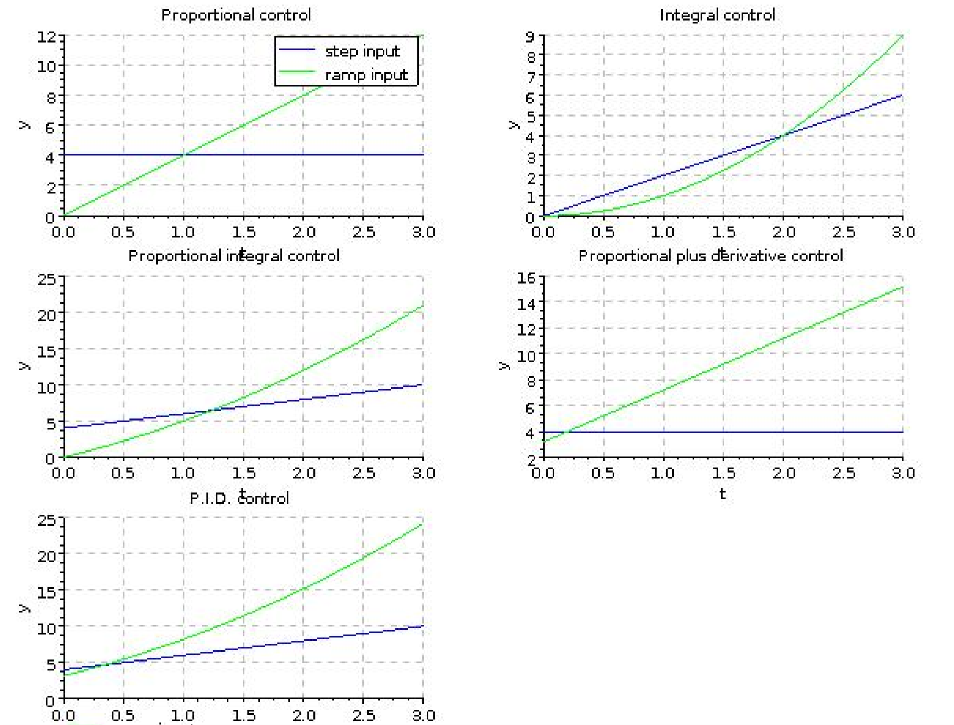

Figure 2.1: Step and Ramp response of different Controllers

```
plot2d(t ,pd1 , style=2);
plot2d(t ,pd2 , style=3);
xtitle('Proportional plus derivative control','t','y
   ');
xgrid(color('gray'));

subplot(3,2,5);
plot2d(t ,pd1 + i1 , style=2);
plot2d(t ,pd2 + i2 , style=3,leg='ramp input') ;
xtitle('P.I.D. control','t','y');
xgrid(color('gray'));
```

Scilab code Exa 2.a.7 Transfer Function to Controllable State Space form

```
// Example A-2-7
// Transfer function to controllable form (state
   space)

clear; clc;close;mode(0);

s = %s;
Num = 2*s^3 + s^2 + s + 2;    n = coeff(Num);
Den = s^3 + 4*s^2 + 5*s + 2; d = coeff(Den);
for i = 1:4 ;  b(i) = n(5 - i); a(i) = d(5 - i); end

// Method 1
_beta(1) = b(1);
_beta(2) = b(2) - a(2)*_beta(1);
_beta(3) = b(3) - a(2)*_beta(2) - a(3)*_beta(1);
_beta(4) = b(4) - a(2)*_beta(3) - a(3)*_beta(2) - a
   (4)*_beta(1);

A = [0 1 0; 0 0 1; -d(1:3)]
B = _beta(2:4)
C = [1 0 0 ]
D = b(1)

// method 2
H2 = cont_frm(Num,Den)
```

Scilab code Exa 2.a.11 State space to Transfer Function model SISO system

```
// Example A-2-11
// Conversion from state space model to transfer
   function model
// for a Single Input Single Output System

```

```
clear; clc; close;

// Please edit the path below
// cd "/your code directory/";
// exec("transferf.sci");

A = [-1 1 0; 0 -1 1; 0 0 -2];
B = [0; 0; 1];
C = [1 0 0];
D = [0];

Htf = transferf(A,B,C,D);          // Htf is the
   tranfer function
disp(Htf,'Htf =');                 // polynomial. ie.
   Htf = num / den
```

check Appendix AP 4 for dependency:

`transferf.sci`

Scilab code Exa 2.a.12 State space to Transfer Function model MIMO system

```
// Example A-2-12
// Conversion from state space model to transfer
   function model
//           for a multiple input multiple output
   system

clear; clc; close;

// Please edit the path below
// cd "/your code directory/";
// exec("transferf.sci");

A = [0 1; -25 -4];
B = [1 1; 0 1];
```

```
C = [1 0; 0 1];
D = [0 0; 0 0];

Htf = transferf(A,B,C,D)       // Htf is the tranfer
    function matrix,
disp(Htf,'Htf =');             // with four transfer
    functions -
                               // Htf(1,1),Htf(1,2),
                                  Htf(2,1),Htf(2,2);
```

check Appendix AP 4 for dependency:

`transferf.sci`

Scilab code Exa 2.b.14 Verifying linearization of a non linear system

```
// Exercise B-2-14

// An illustration on Linearization
// Linearize the function y = f(x) = 0.2*x^3 at x=2
// SOLUTION : y = 2.4*x - 3.2

// Let us observe graphically the linear
    approximation
// and the error, and percentage error

clear; clc; xdel(winsid());

x = 0.05:0.05:5;
y = 0.2 * x .^ 3;

yl = 2.4 * x - 3.2 ;        // this is not a linear
    system!
err = abs(y - yl);          //Error in approximation
errpc = err ./ y  * 100;    //Percentage error

```

```
subplot(2,1,1);
plot2d(x,y,style=2);
plot2d(x,yl,style=3,leg="linearized system");
xtitle('Original and linearized system','x','y');

subplot(2,1,2);
plot2d(x,err,style=5);
xtitle('Error','x','error');

scf();
plot2d(x,errpc,style=5,rect=[1 0 3 100]);
plot2d(x, 10 * ones(1,length(x)) ,style=2,leg="10%
    error margin" );
xtitle('Percentage Error','x','% error');
```

check Appendix AP 4 for dependency:

`transferf.sci`

Scilab code Exa 2.4 `Convert State space to Transfer Function model`

```
// Example2-4
// Conversion from state space to transfer function
    model

clear;clc;close;

// Please edit the path below
// cd "/your code directory/";
// exec("transferf.sci");

A = [0 1 0; 0 0 1;-5 -25 -5];
```

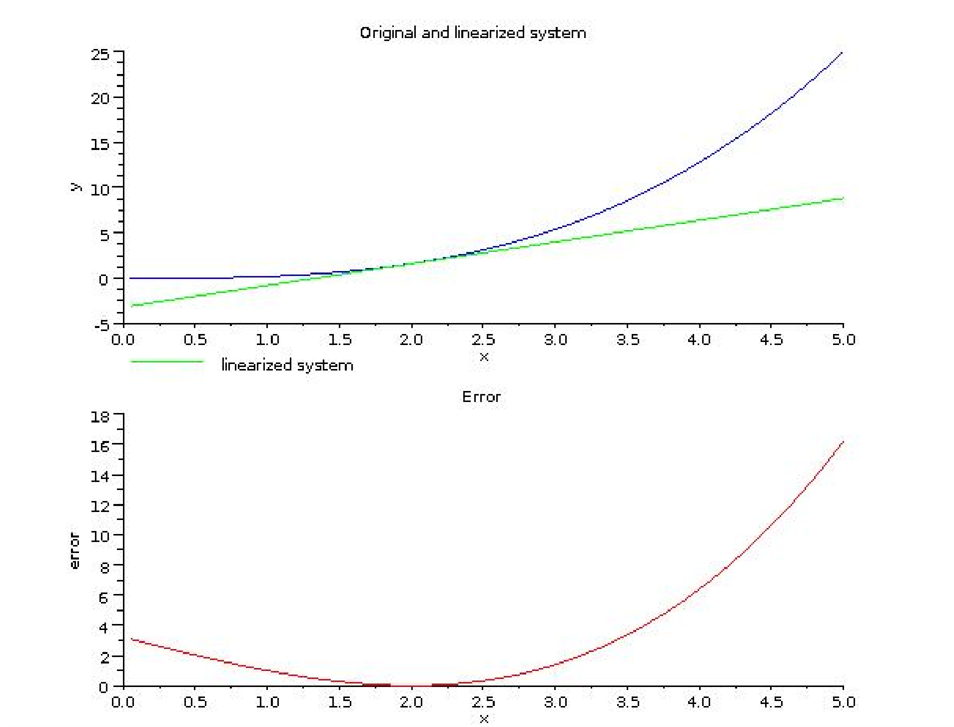

Figure 2.2: Verifying linearization of a non linear system

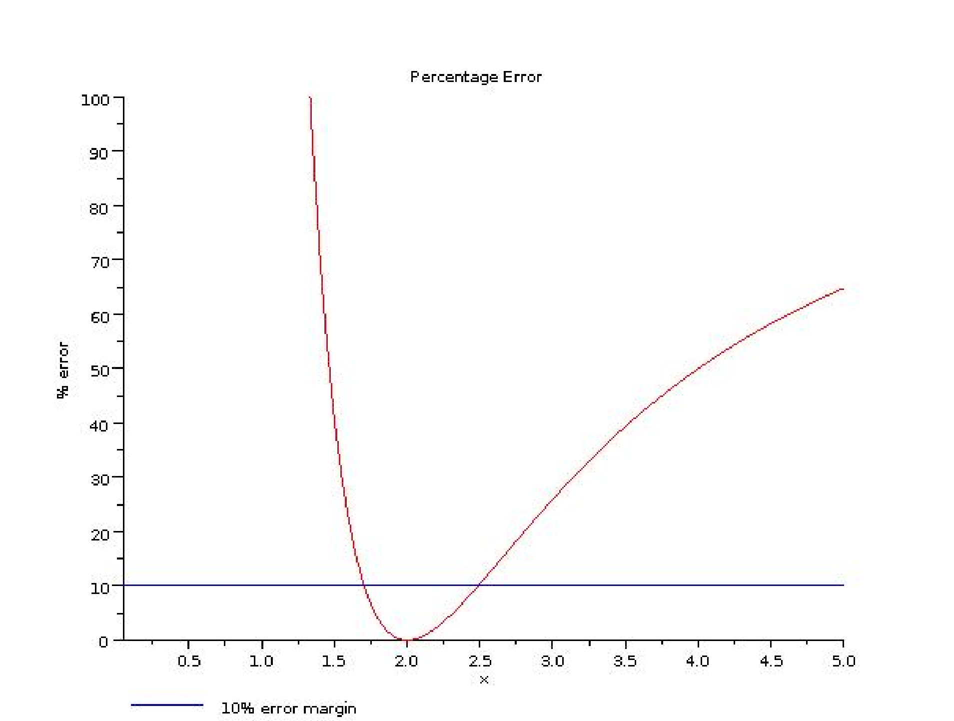

Figure 2.3: Verifying linearization of a non linear system

```
B = [0; 25; -120];
C = [1 0 0];
D = [0];
G = transferf(A,B,C,D);
disp(G,'transfer function = ');
```

Chapter 5

Transient and Steady State Response Analysis

Scilab code Exa 5.a.3 Verifying design to match given response curve

```
// Example A-5-3
// Verifying design to match given response curve

clear; clc;
xdel(winsid()); //close all windows

// Please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

s = %s;
K = 1.42;
T = 1.09;
K = 1.42;
G1 = (K/(s*(T*s + 1)) ) /. 1;
G = syslin('c',G1);

t = 0:0.1:10;
u = ones(1,length(t));
```

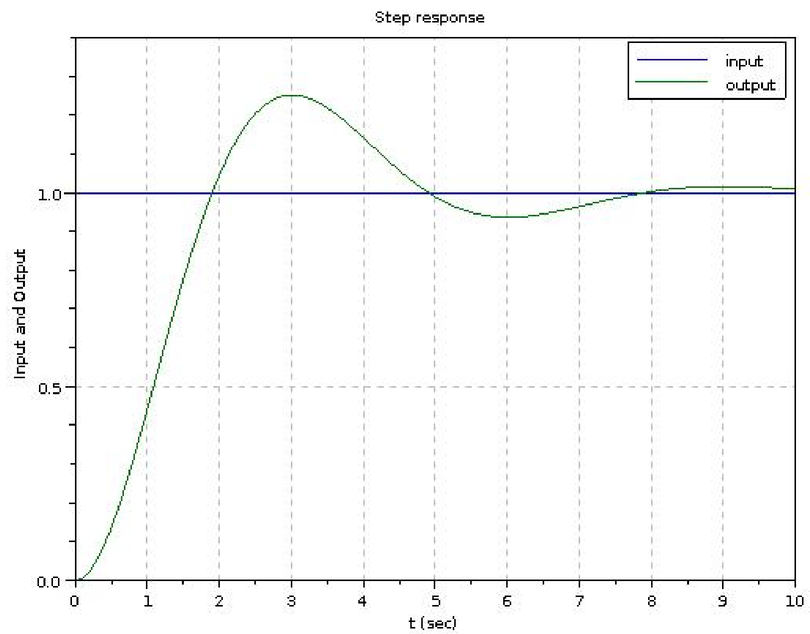

Figure 5.1: Verifying design to match given response curve

```
y = plotresp(u,t,G,'Step response');

[m t] = max(y);
Mp = m - 1;
tp = (t - 1) * 0.1;
disp(Mp,'Mp = ');
disp(tp,'tp = ');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 5.a.4 Determining K and k for required step response

```
// Example A-5-4
// Determining K and k for required step response
   charecteristics

clear; clc;
xdel(winsid());
mode(0);

Mp = 0.25;
tp = 2;
J  = 1; // kg.m^2

z = poly(0,'z');
Eq = (z*%pi)^2 - log(1/Mp)^2 * (1 - z^2);
x = roots(Eq);
zeta = abs(x(1))

wd = %pi / tp
wn = wd / sqrt(1 - zeta^2)
K  = J * wn^2
k = 2*zeta*wn / K
```

Scilab code Exa 5.a.5 Verifying design to match given response

```
// Example A-5-5
// Verifying design to match given response curve

clear; clc;
xdel(winsid());  //close all windows

s = %s;
m = 5.2; // lb / ft^2
b = 12.2;// lb/ft/sec
```

```
k = 20;   // lb /ft
G = syslin('c',1,m*s^2 + b*s + k);

STEP = 0.05; t = 0:STEP:7;
u = 2 * ones(1,length(t));
y = csim(u,t,G);
plot(t,y);
xgrid(color('gray'));
xtitle('Step response','t sec','Response');

[m t] = max(y);
Mp = (m - 0.1) /0.1 * 100;
tp = (t - 1) * STEP;
disp(Mp,'Mp (percent) = ');
disp(tp,'tp = ');
```

Scilab code Exa 5.a.8 Unit step response and partial fraction expansion

```
// Example A-5-8
// Unit step response and partial fraction expansion

clear; clc;
xdel(winsid());  //close all windows

// Please edit path
// cd "<your codes path>/";
// exec("pf_residu.sci");
// exec("plotresp.sci");

s = %s ;
N = poly( [80 72 25 3],'s','c');
D = poly( [80 96 40 8 1],'s','c');
G = syslin('c',N,D)
```

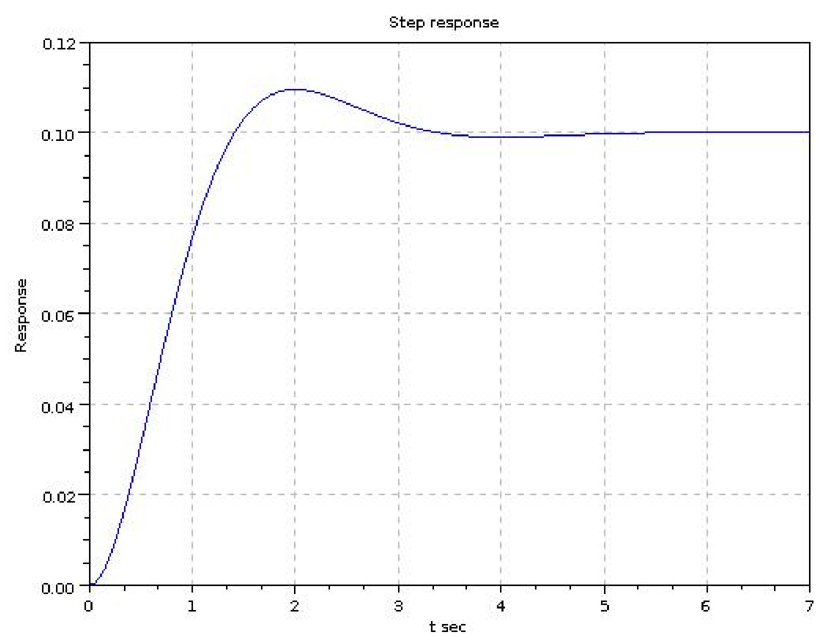

Figure 5.2: Verifying design to match given response

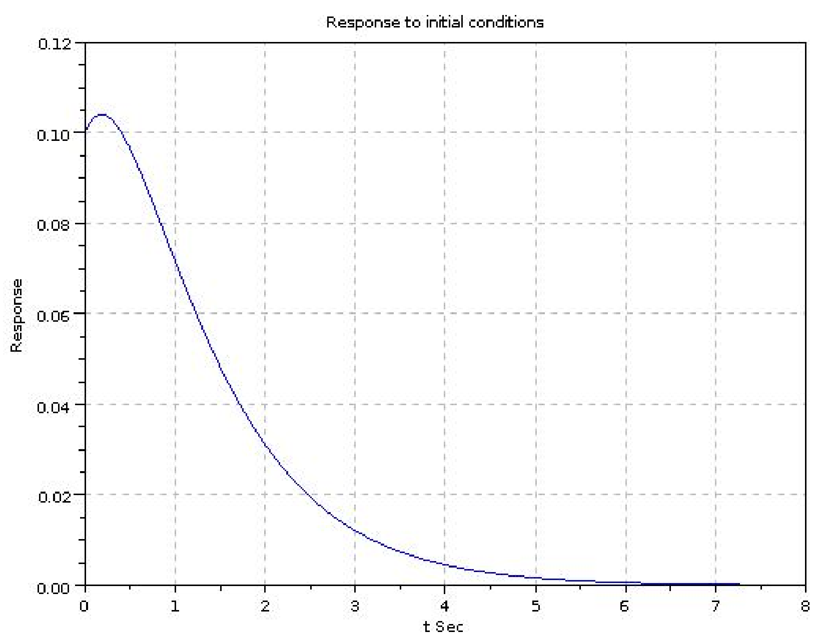

Figure 5.3: Unit step response and partial fraction expansion

```

t = 0:0.05:5;
u = ones(1,length(t));
plotresp(u,t,G,'Unit Step Response of C(s) / D(s)');

// To find the residues of step response
D = D * s;
[r,z,p] = pf_residu(N,D);

disp(z,'zeros = ');disp([p,r],'poles : residues =');
```

check Appendix AP 6 for dependency:

`pf_residu.sci`

check Appendix AP 2 for dependency:

plotresp.sci

Scilab code Exa 5.a.9 Effect of zeros on step response of a system

```
// Example A-5-9
// Effect of zeros on step response of a system
// Interactive program

clear; clc;
xdel(winsid()); //close all windows

function drawg()
  delete(gca())
  N = 4*(s*1/z + 1);
  G = syslin('c',N,D);
  ys = csim('step',t,G);
  m = max(ys);
  Mp = m -1;
  plot(t,ys);
  xtitle('Unit Step Response for zero at z =' +
     string(z) + ' Mp = ' + string(Mp),'t (sec)','
     Output');
  xgrid(color('gray'));
  a = gca();
  a.data_bounds = [0 0;10 4]
endfunction

s = %s;
z = 0.2;
D = s^2 + 4*s + 4;
t = 0:0.02:10;
drawg();
h = uicontrol('style','pushbutton','position','
    250|10|60|20','callback','z = z - 0.1;drawg()','
```

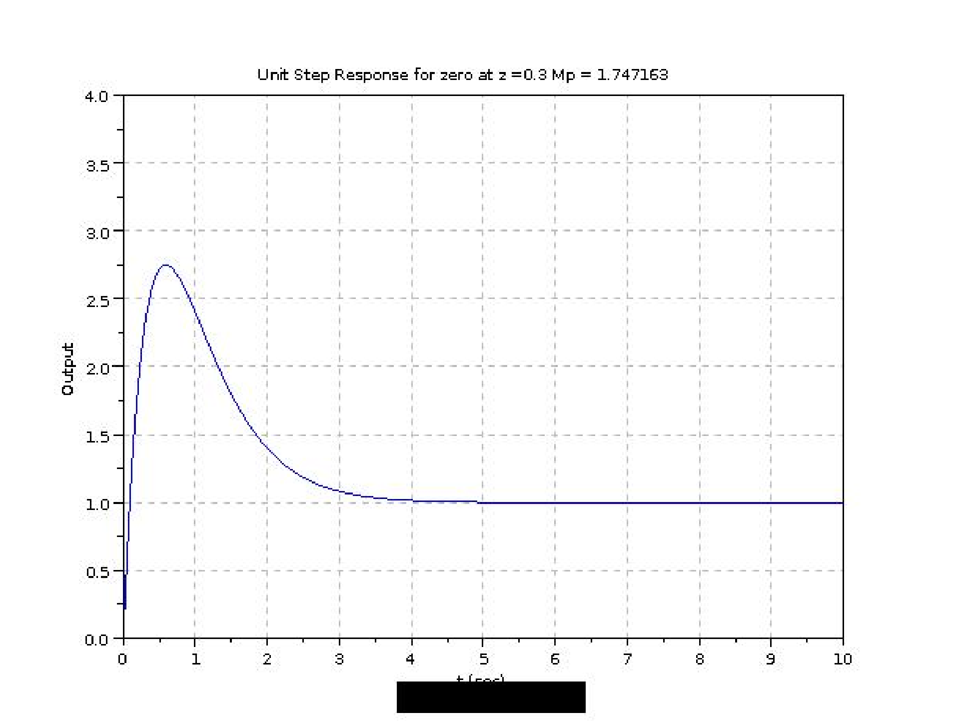

Figure 5.4: Effect of zeros on step response of a system

```
String','<-');
j = uicontrol('style','pushbutton','position','
310|10|60|20','callback','z = z + 0.1;drawg()','
String','->');
```

Scilab code Exa 5.a.10 `Step response characteristics`

```
// Example A-5-10
// Plot the unit step response and find the
transient parameters
// viz. - rise time, peak time , settling time and
```

```
    maximum overshoot

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// Please edit path if needed
// cd "/<your code path>/";
// exec("stepch.sci");

N = poly( [12.811 18 6.3223],'s','c') ;
D = poly( [12.811 18 11.3223 6 1], 's','c');
G = syslin('c',N,D);
[Mp tp tr ts] = stepch(G,0,20,0.01,0.02)
```

check Appendix AP 8 for dependency:

`stepch.sci`

Scilab code Exa 5.a.11 `Step Response for different zeta and wn`

```
// Example A-5-11
// Unit Step Response for different systems for
    different zeta,wn

clear; clc;
xdel(winsid()); //close all windows

zeta = [0.3 0.5 0.7 0.8];
wn   = [1 2 4 6];
n    = wn .^ 2;
sigma= 2 .* zeta .* wn;

s = %s;
```

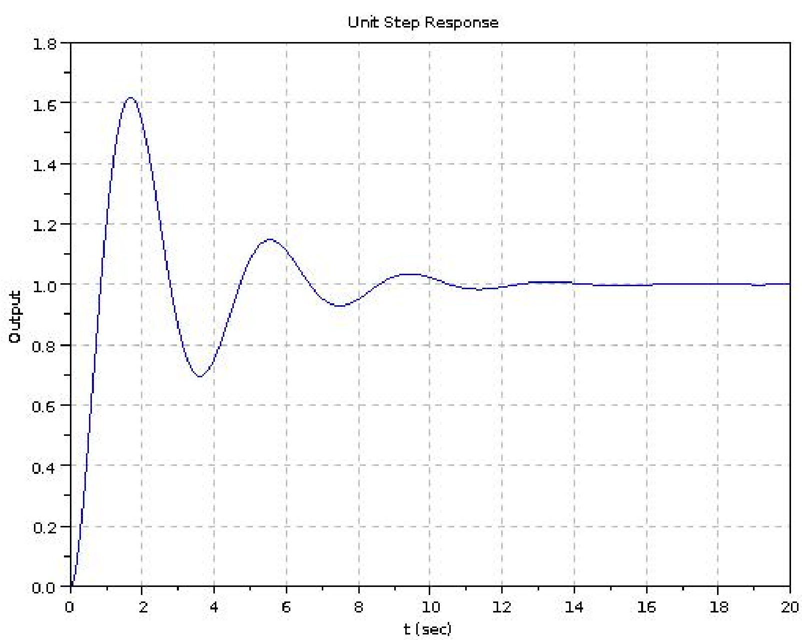

Figure 5.5: Step response characteristics

```
t = 0:0.1:10;
for i= 1:4
z(i,:) = csim('step',t,syslin('c', n(i), s^2 + sigma
    (i)*s + n(i) ));
end

plot(t,z);  // 2d plot of step responses

xtitle('Plot of step response curves with different
    wn and zeta','t sec','Response');
xgrid(color('gray'));
legend('(zeta,wn) = (0.3 , 1)','(0.5 , 2)','(0.7 ,
    4)','(0.8 , 6)');
```

Scilab code Exa 5.a.12 `Response to unit ramp and exponential input`

```
// Example A-5-12
// Response to unit ramp and exponential input

clear; clc;
xdel(winsid());  //close all windows

// Please edit path if needed
// cd "/<your code folder>/"
// exec("plotresp.sci");

s = %s;
G = syslin('c', s + 10, s^3 + 6*s^2 + 9*s + 10);

t = 0:0.05:10;
e = exp(-0.5 * t);
plotresp(t,t,G,'Response to unit ramp input');
scf();
```

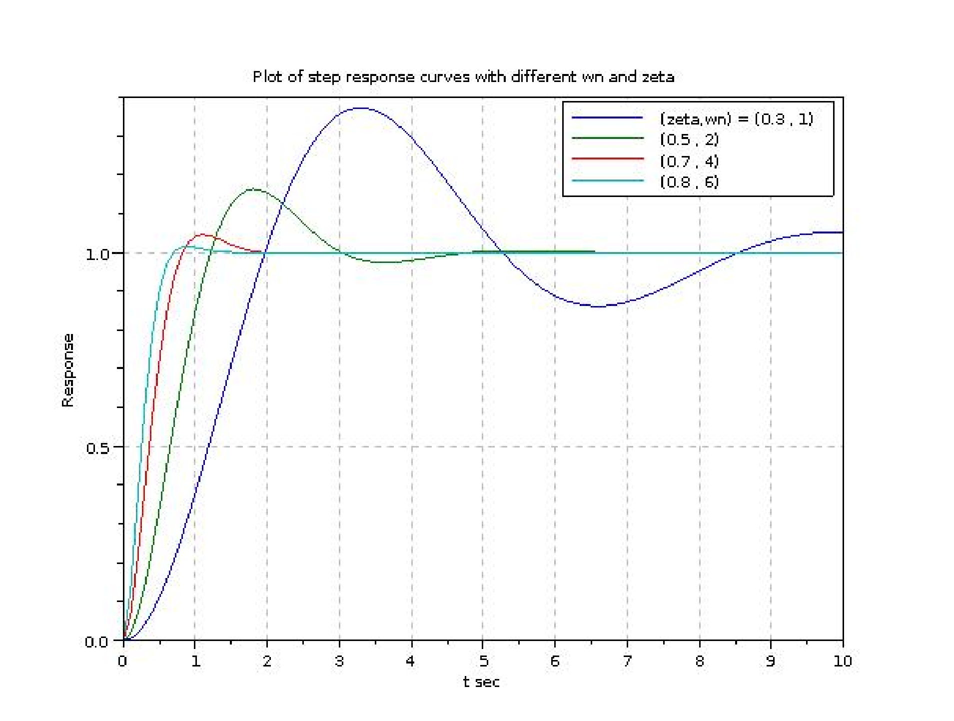

Figure 5.6: Step Response for different zeta and wn

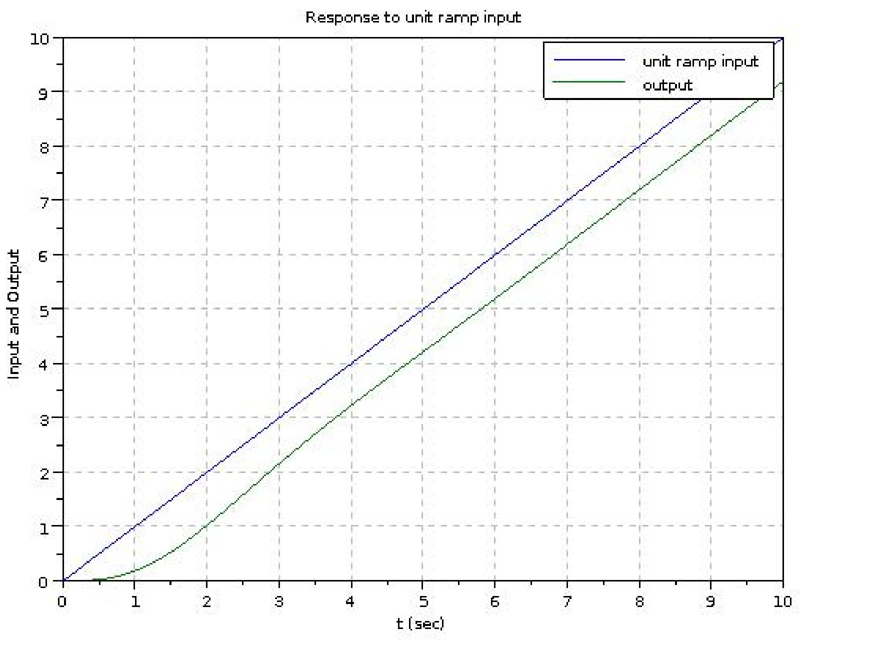

Figure 5.7: Response to unit ramp and exponential input

```
plotresp(e,t,G,'Response to exponential input');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 5.a.13 Response to input r equals 2 plus t

```
// Example A-5-13
// Response to input r = 2 + t
```

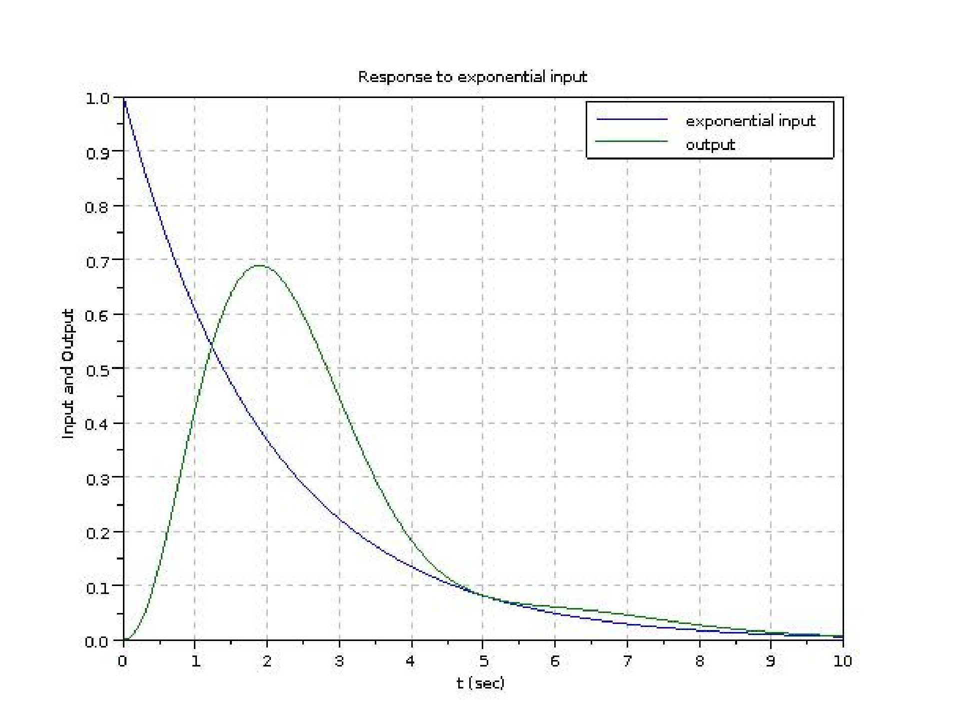

Figure 5.8: Response to unit ramp and exponential input

```
clear; clc;
xdel(winsid());  //close all windows

// Please edit the path
// cd "/<your code folder>/Codes/chapter_5";
// exec("plotresp.sci")

s = %s;
G = syslin('c', 5, s^2 + s + 5);
t = 0:0.05:10;
r = 2 + t;
plotresp(r,t,G,'Response to input r = 2 + t');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 5.a.14 `Response to unit acceleration input`

```
// Example A-5-14
// Response to unit acceleration r = (1/2) * t^2

clear; clc;
xdel(winsid());  //close all windows

// please edit the path
// cd "/<your code folder>/Codes/chapter_5"
// exec("plotresp.sci")

s = %s;
G = syslin('c', 2, s^2 + s + 2);
t = 0:0.05:10;
r = (1/2) * t.^2;
```

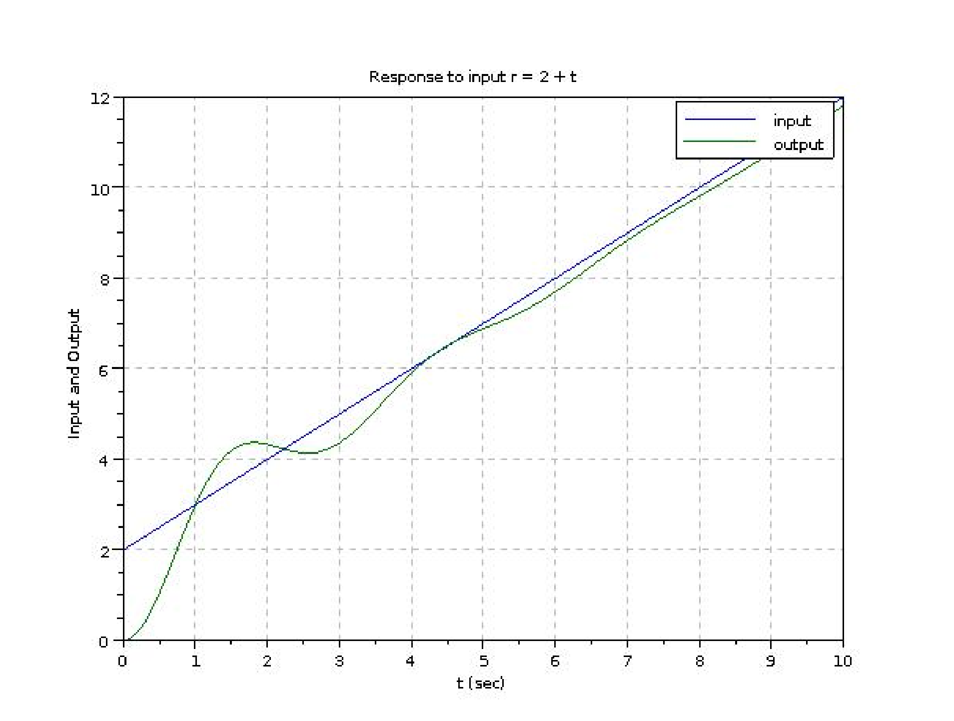

Figure 5.9: Response to input r equals 2 plus t

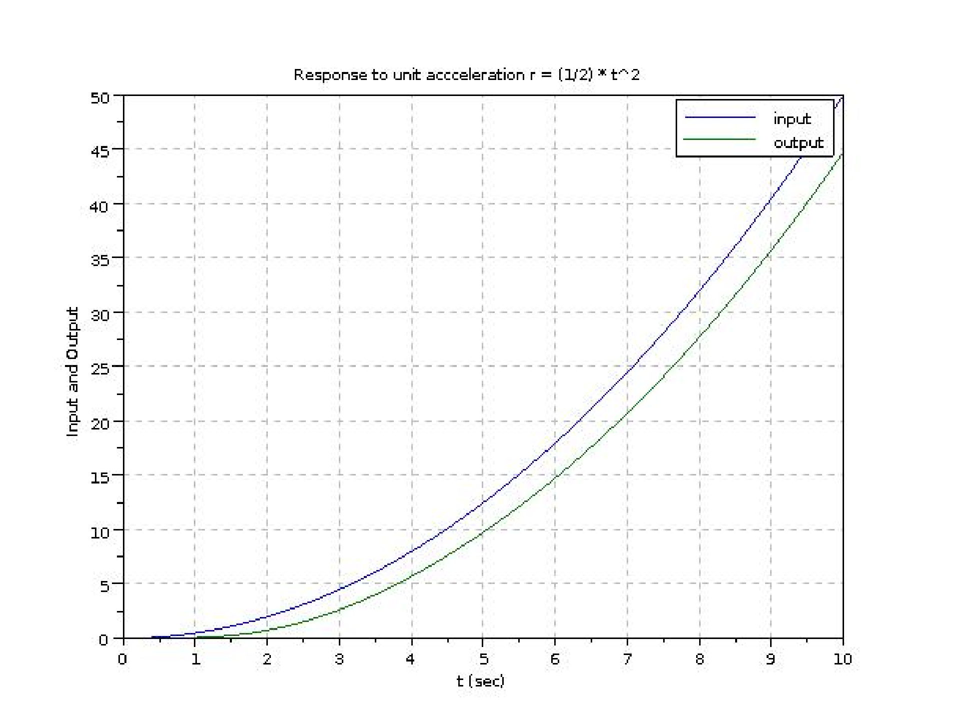

Figure 5.10: Response to unit acceleration input

```
plotresp(r,t,G,'Response to unit accceleration r =
    (1/2) * t^2');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 5.a.15 `Step Responses for different zeta`

```
// Example A-5-15
// 2d and 3d plot for various values of zeta

```

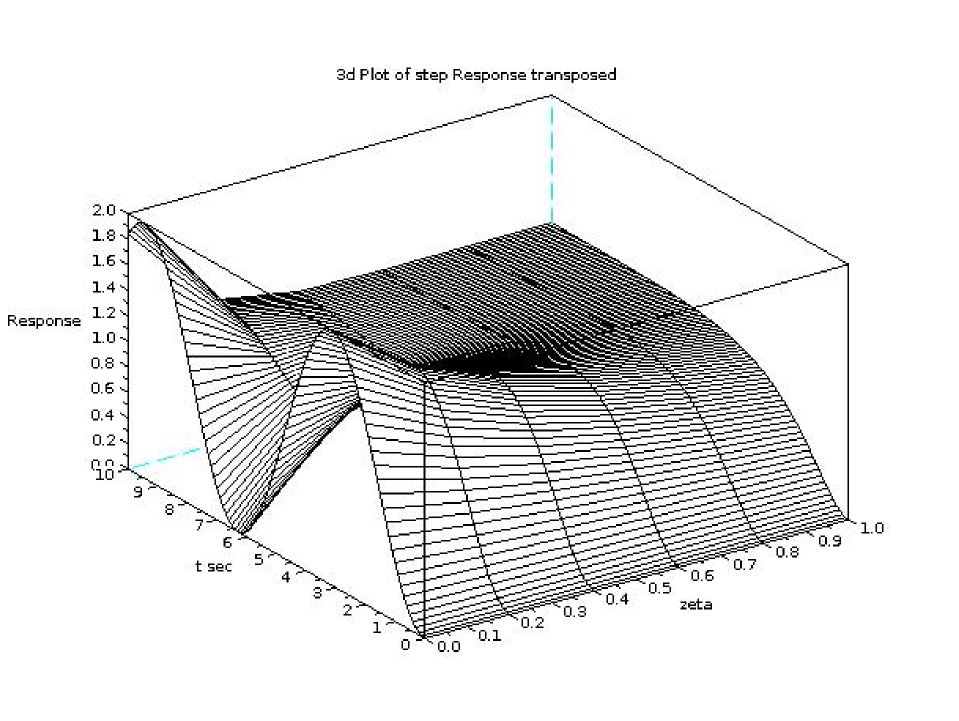

Figure 5.11: Step Responses for different zeta

```
// Please refer to example 5-4

// To get the trasnposed plot please add the lines

scf();
mesh(y,x,z);
xtitle(' 3d Plot of step Response transposed','zeta'
   ,'t sec','Response');
```

Scilab code Exa 5.a.16 `Response to initial conditions`

```
// Example A-5-16
// Response to initial conditions

clear; clc;
xdel(winsid()); //close all windows

A = [0 1 0; 0 0 1; -10 -17 -8];
C = [1 0 0];
x0 = [2; 1; 0.5];
G = syslin('c',A,[0; 0; 0],C,0,x0);

t = 0:0.05:10;
u = zeros(1,length(t));
y = csim(u,t,G);

plot(t,y);
xgrid(color('gray'));
xtitle('Response to initial condition','t (sec)','
   output');
```

Scilab code Exa 5.2 Determining K and Kh for required step response

```
// Example 5-2
// Determining K and Kh for required step response
   charecteristics

clear; clc;
xdel(winsid());
mode(0);

Mp = 0.2;
tp = 1;
J  = 1; // kg.m^2
```

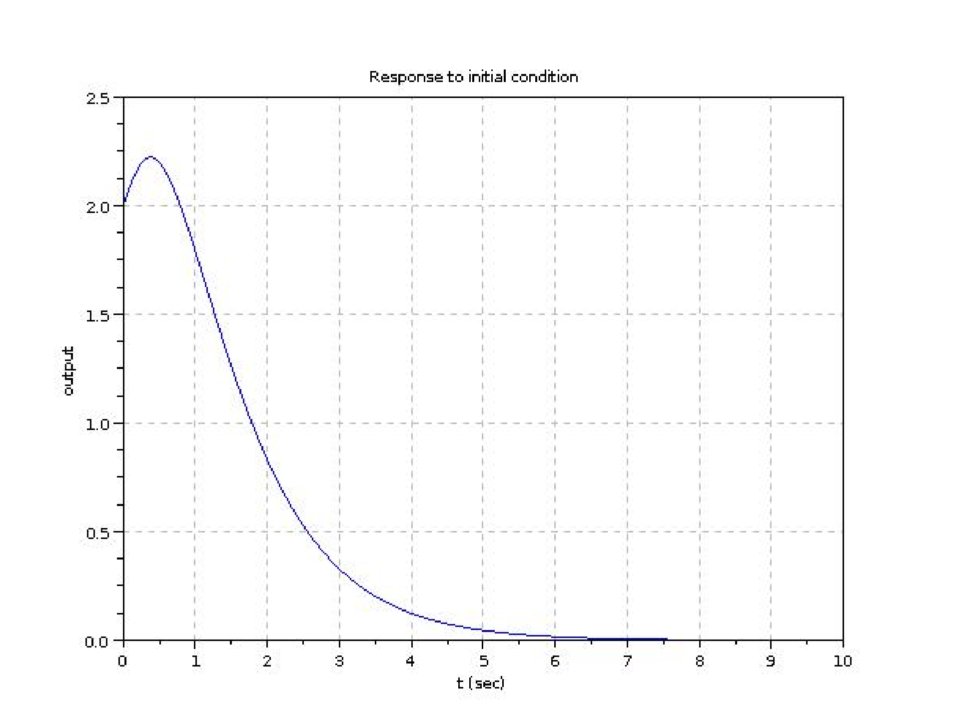

Figure 5.12: Response to initial conditions

```
B  = 1; // N-/rad/sec

z = poly(0,'z');
Eq = (z*%pi)^2 - log(1/Mp)^2 * (1 - z^2);
x = roots(Eq);
zeta = abs(x(1))

wd = %pi / tp
wn = wd / sqrt(1 - zeta^2)
K  = J * wn^2
Kh = (2*sqrt(K*J)*zeta - B) / K

sigma = wn*zeta;
_beta = atan(wd/sigma)
tr = (%pi - _beta) / wd
ts_2percent = 4 / sigma
ts_5percent = 3 / sigma
```

Scilab code Exa 5.3 Step response of MIMO system

```
// Example 5-3
// Step response of a linear System given in State
   Space
// Model (Multiple Input Multiple Output System)

clear; clc;
xdel(winsid()); //close all windows

A = [ -1 -1; 6.5 0];
B = [ 1 1; 1 0];
C = [ 1 0; 0 1];
D = [ 0 0; 0 0];
G = syslin('c',A,B,C,D);
Gtf = clean(ss2tf(G));
disp(Gtf,'Gtf = '); //transfer function matrix
```

```

N = 200;                          //No of points
t = linspace(0,10,N);
u1 = [ones(1,N) ; zeros(1,N)];
u2 = [zeros(1,N); ones(1,N) ];

y1 = csim(u1,t,G);                // find system response
y2 = csim(u2,t,G);

plot(t,y1);
xtitle('Unit Step Response: input = u1 (u2 = 0)','t Sec','Response');
xgrid(color('gray'));         // grid
legend('output: y1','output: y2');

scf(1);                           // new window
plot(t,y2);
xtitle('Unit Step Response: input = u2 (u1 = 0)','t Sec','Response');
xgrid(color('gray'));
legend('output: y1','output: y2');

 // We cannot use csim('step' , , ) because this option is only available
 // for SISO sytems
```

Scilab code Exa 5.4 Second order systems with different damping ratio

```
// Example 5-4
// 2d and 3d plots of standard second order systems
// with wn = 1 and different damping ratios
```

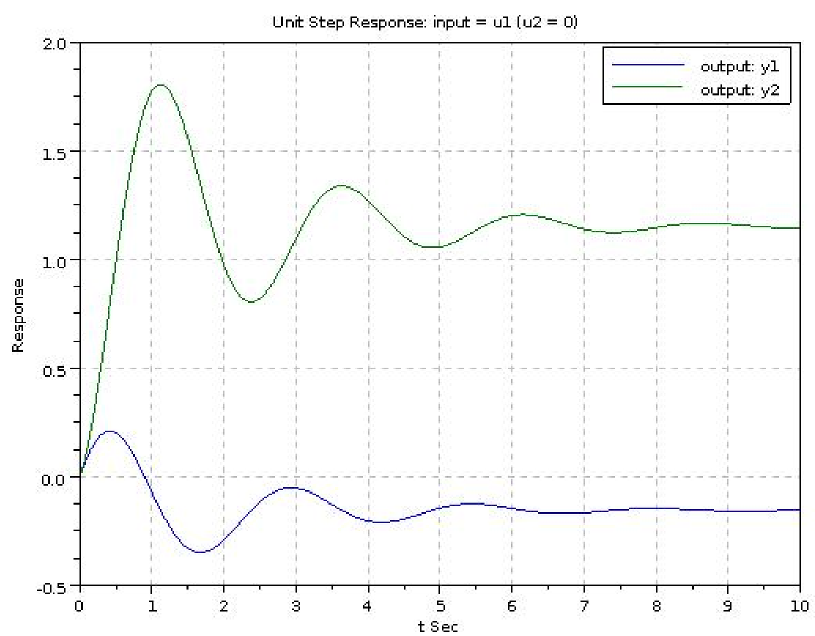

Figure 5.13: Step response of MIMO system

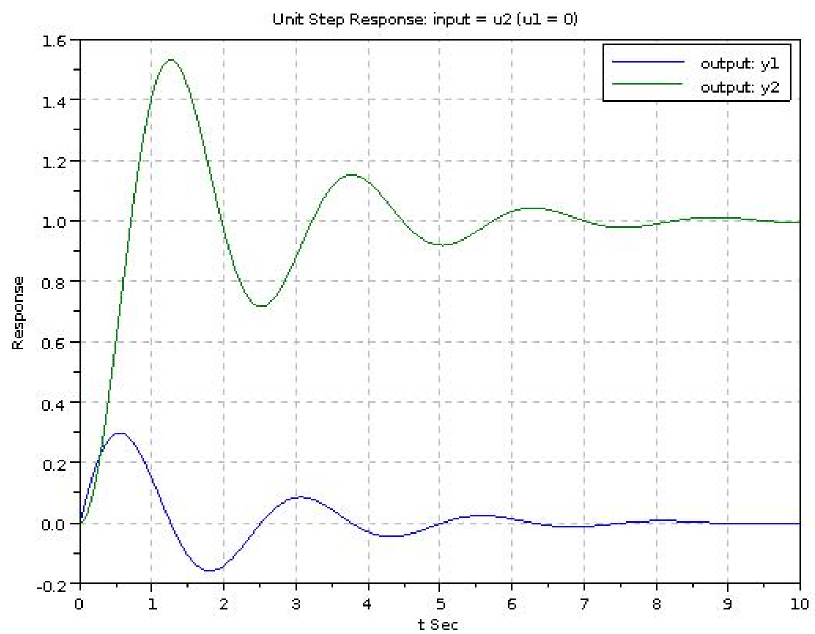

Figure 5.14: Step response of MIMO system

```

clear; clc;
xdel(winsid()); //close all windows

s = %s;
t = 0:0.1:10;
zeta = 0:0.2:1;

for n = 1:6
    z(n,:) = csim('step',t,syslin('c', 1,s^2 + 2*
       zeta(n)*s + 1));
end

plot(t,z);  // 2d plot of step responses
xtitle('Plot of step response curves with wn = 1 and
     different zeta','t sec','Response');
xgrid(color('gray'));
legend('zeta = 0','0.2','0.4','0.6','0.8','1.0');

scf();    // new window

[x,y] = meshgrid(0:0.1:10 , 0:0.2:1);  //needed by
   the mesh command
mesh(x,y,z);
xtitle(' 3d Plot of step Response','t sec','zeta','
   Response');
```

Scilab code Exa 5.5 `Impulse Response of a Second order System`

```
// Example 5-5
// Impulse Response of a Second Order System
```

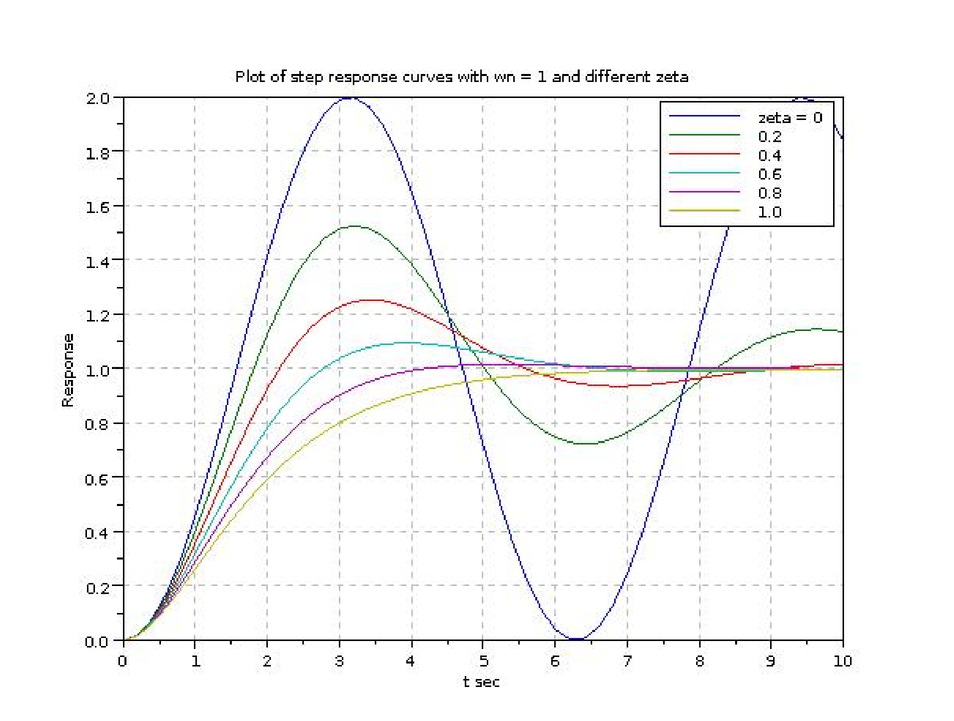

Figure 5.15: Second order systems with different damping ratio

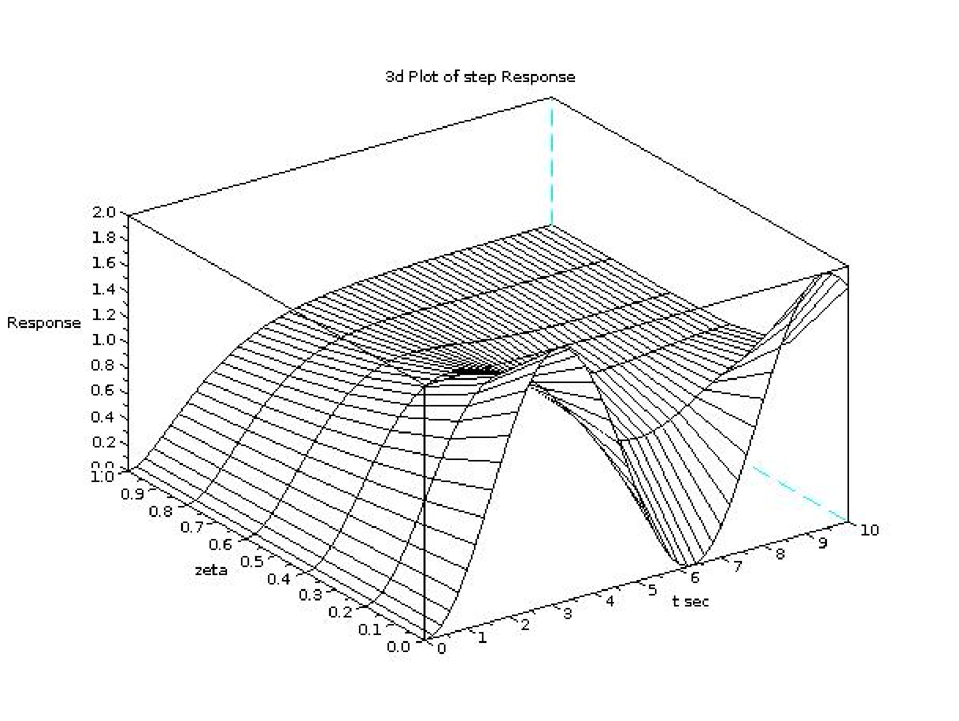

Figure 5.16: Second order systems with different damping ratio

```

clear; clc;
xdel(winsid()); //close all windows

s = %s;
G = syslin('c', 1, s^2 + 0.2*s + 1);

t = 0:0.5:50;
y = csim('impuls',t,G);
plot(t,y);
xtitle('Impulse Response of 1/ (s^2 + 0.2*s + 1)','t sec','Response');
xgrid(color('gray'));
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 5.6 `Unit Ramp response of a second order system`

```
// Example 5-6
// Unit Ramp response of a second order system

clear; clc;
xdel(winsid()); //close all windows

// Please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

s = %s
G = syslin('c', 2*s + 1, s^2 + s + 1);

t = 0:0.05:10;
```

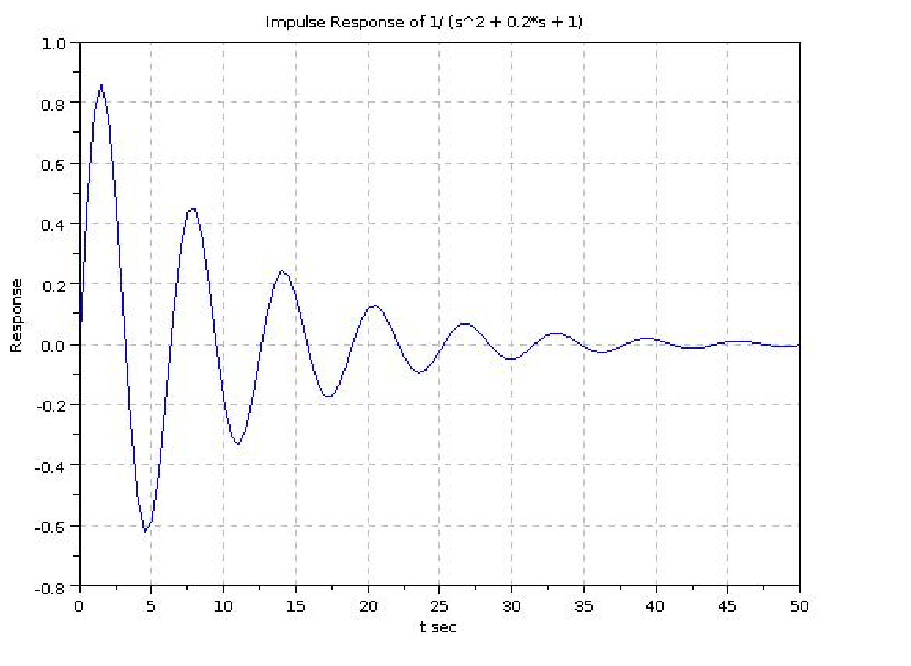

Figure 5.17: Impulse Response of a Second order System

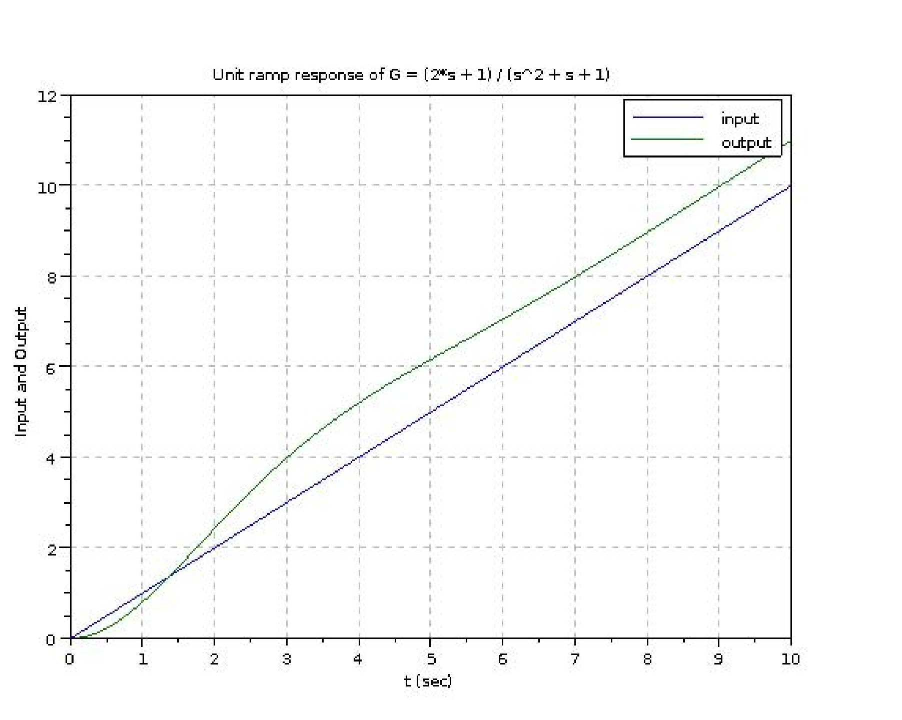

Figure 5.18: Unit Ramp response of a second order system

```
plotresp(t,t,G,'Unit ramp response of G = (2*s + 1)
    / (s^2 + s + 1)');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 5.7 `Response to step and exponential input`

```
// Example 5-7
// Response to step and exponential input

```

```
clear; clc;
xdel(winsid()); //close all windows

// Please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

t = 0:0.1:16;
A = [-1  0.5; -1  0];
B = [0; 1];
C = [1  0];
D = [0];
G = syslin('c',A,B,C,D);

// unit step response
u = ones(1,length(t));
plotresp(u,t,G,'Unit-Step Response');
scf();
// resposne to exponential input = e^(-t)
u = exp(-t);
plotresp(u,t,G,'Response to exponential input');
```

Scilab code Exa 5.8 Response to initial condition

```
// Example 5-8
// Response to initial condition (Transfer Function)

clear; clc;
xdel(winsid()); //close all windows

s = %s;
```

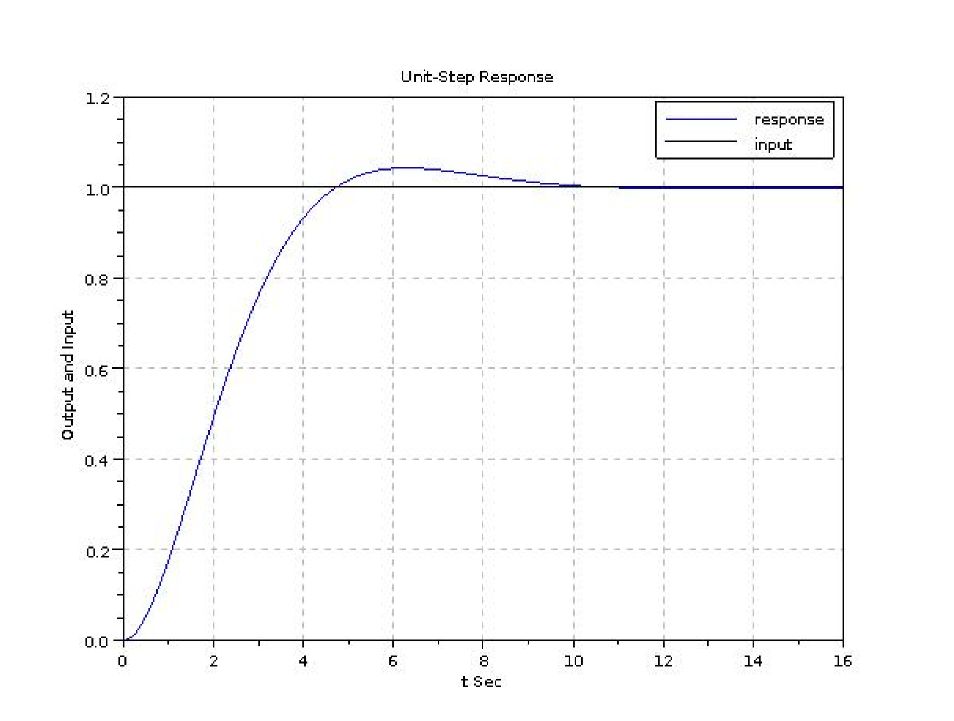

Figure 5.19: Response to step and exponential input

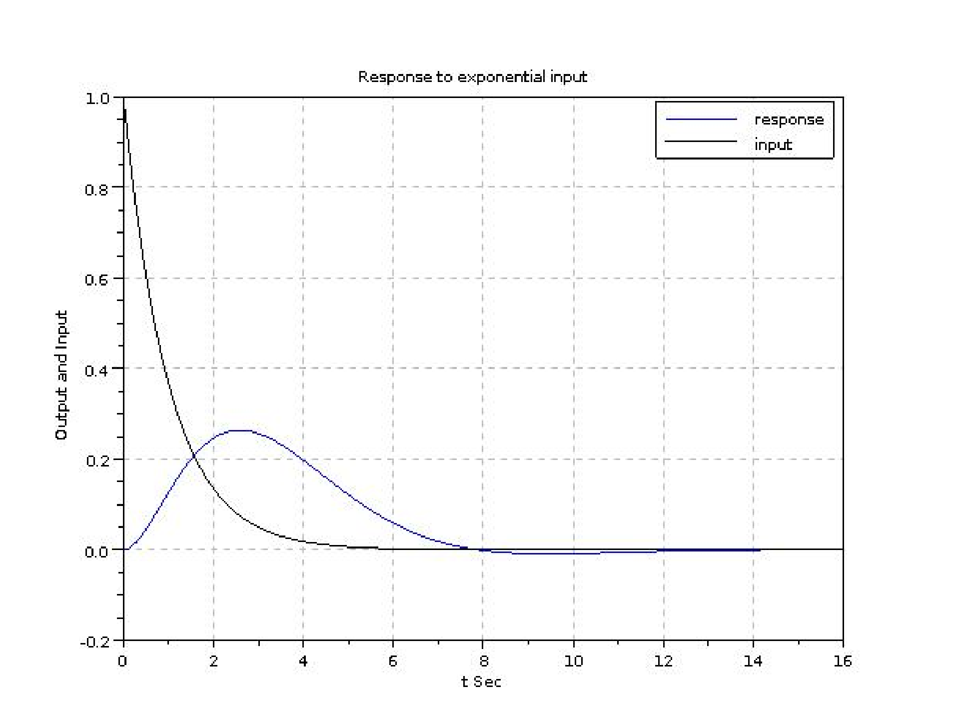

Figure 5.20: Response to step and exponential input

```
N = 0.1*s^2 + 0.35*s ;
D = s^2 + 3*s + 2;
G = syslin('c',N,D);

t = linspace(0,8,200);
u = ones(1,200);
y = csim(u,t,G);

plot(t,y);
xtitle('Response to initial conditions','t Sec','Response');
xgrid(color('gray'));
// We cannot use the 'step' version of csim directly
// as direct feedback sets to zero for the 'step' option
```

Scilab code Exa 5.9 Response to initial conditions using state space

```
// Example 5-9
// Response to initial conditions using state space approach

clear; clc;
xdel(winsid()); //close all windows

A = [0  1; -10 -5];
x0 = [2; 1];
G = syslin('c',A,x0,[0 0],[0]); //use dummy C and D variables

t = 0:0.01:3;
[y,x] = csim('impuls',t,G);

```

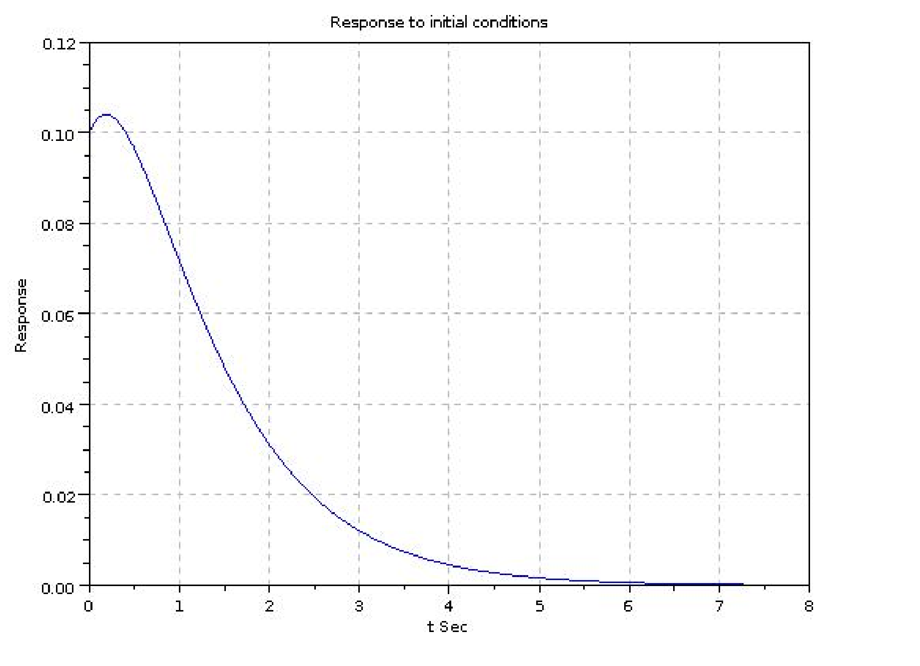

Figure 5.21: Response to initial condition

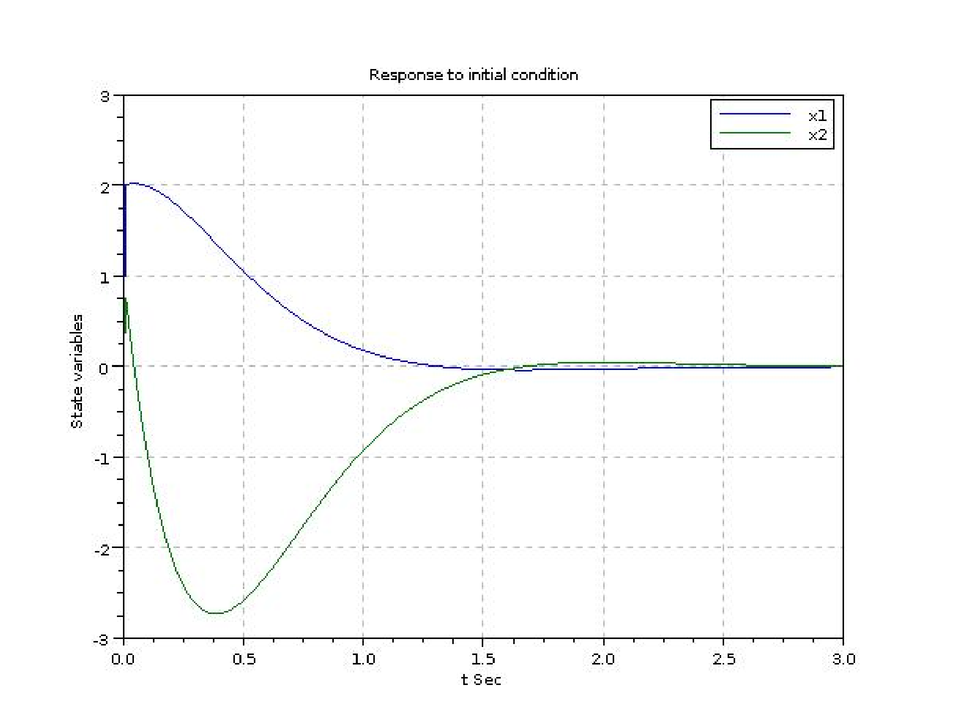

Figure 5.22: Response to initial conditions using state space

```
plot(t, x(1,:), t, x(2,:));
xtitle('Response to initial condition','t Sec','State variables');
xgrid(color('gray'));
legend('x1','x2');
// The State variables x, respond only to A,B matrices
// changning C and D will make no difference.
```

Scilab code Exa 5.10 `Response to initial condition using syslin x0`

```
// Example 5-10
// Response to initial condition (differential
   equation)
// Solution of differential equation with initial
   conditions

clear; clc;
xdel(winsid()); //close all windowss

t = 0:0.05:10;
s = %s;
G1 = cont_frm(1, s^3 + 8*s^2 + 17*s + 10); //get the
    state space model
ssprint(G1);

x0 = [2; 1; 0.5];   // initial states of the system
G = syslin('c', G1.A, G1.B, G1.C, G1.D, x0);

y = csim( zeros(1,length(t)) , t, G);
        // response to zero input will give response
            to initial state
plot(t,y);
xgrid(color('gray'));
xtitle('Response to initial conditions','t Sec','y')
   ;
```

Scilab code Exa 5.12 Constructing Routh array

```
// Example 5-12
// Constructing Routh array in scilab

clear; clc;
xdel(winsid()); //close all windows
```

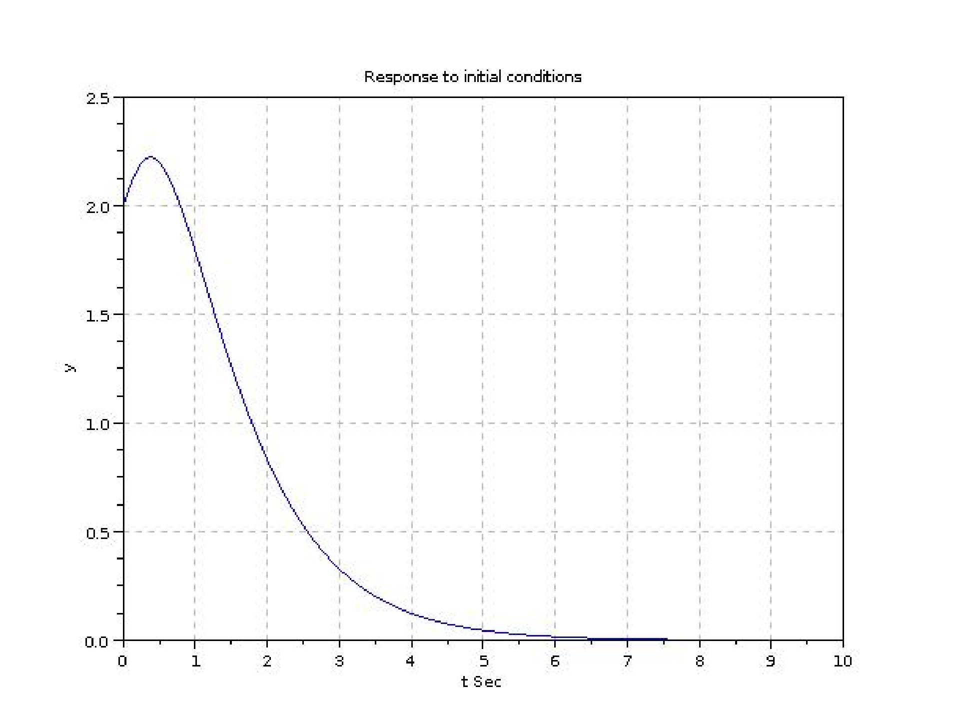

Figure 5.23: Response to initial condition using syslin x0

```
mode(0);

s = %s;
H = s^4 + 2*s^3 + 3*s^2 + 4*s + 5;
routh_t(H)       // display the routh table
```

Scilab code Exa 5.13 Constructing Routh array

```
// Example 5-13
// Constructing Routh array in scilab

clear; clc;
xdel(winsid());  //close all windows
mode(0);

s = %s;
H = s^5 + 2*s^4 + 24*s^3 + 48*s^2 - 25*s - 50;
routh_t(H)

// In this example a zero row forms at s^3
// the function atutomatically computes the
   derivative of the
// auxilliary polynomial 2s^4 + 48s^2 - 50
// viz = 8*s^3 + 96s^2
```

Chapter 6

Control Systems Analysis and Design by Root Locus Method

check Appendix AP 12 for dependency:

gainat.sci

Scilab code Exa 6.i.1 Finding the Gain K at any point on the root locus

```
// Illustration 6.1
// Finding the Gain K at any point on the root locus

clear; clc;
xdel(winsid()); //close all windows

// please set the path
// cd "/<your code directory>/"
// exec("rootl.sci");
// exec("gainat.sci");

function drawr()
```

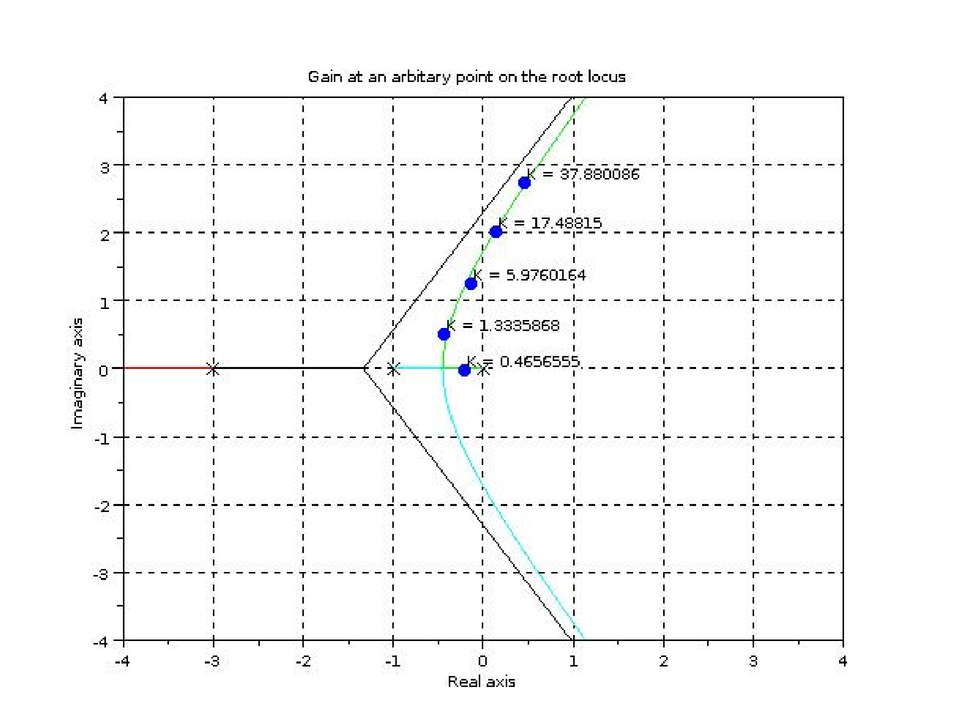

Figure 6.1: Finding the Gain K at any point on the root locus

```
  rootl(G,[-4 -4; 4 4],'Gain at an arbitary point on
      the root locus');
endfunction

s = %s;
G = syslin('c',1, s * (s + 1) * (s + 3) );
drawr();
addmenu(0,'Gain',['Select Point 5 points','clear']);
Gain_0 = ['for i = 1:5; gainat(G); end;','delete(gca
    ());drawr();'];

// click on the Gain menu in the menu bar
// clear will restore your rootlocus
```

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.i.2 `Orthogonality Constant gain curves and Root Locus`

```
// Illustration 6.2
// Orthogonality of constant gain curves and root
    locus
// and the root locus

// Section6.3  Figure 6-29 in the book

clear; clc;
xdel(winsid()); //close all windows

// please set the path
// cd "/<your code directory>/"
// exec("rootl.sci");

```

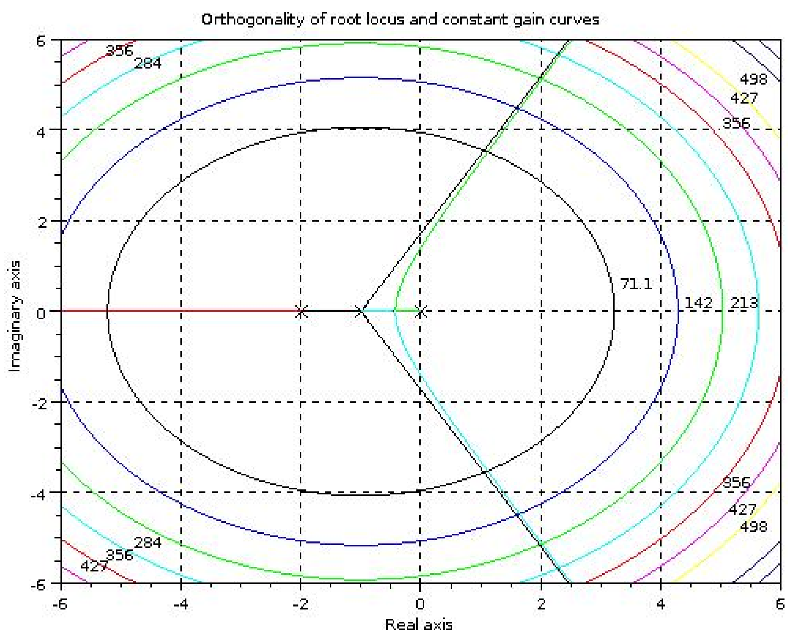

Figure 6.2: Orthogonality Constant gain curves and Root Locus

```
s = %s;
P = 1 /( s * (s + 1) * (s + 2) );
G = syslin('c',P);

rootl(G,[ -6 -6; 6 6 ],'Orthogonality of root locus
    and constant gain curves');

P = 1 / P;
v = -6:0.1:6;
[X,Y] = ndgrid(v,v); // prepares a grid to compute
    the gain
S = X + %i * Y;
K = abs(horner(P,S)) ; // Gain evaluated over the
    grid

contour(v,v,K,10);  // plot lines of constant gain
```

check Appendix AP 7 for dependency:

rootl.sci

Scilab code Exa 6.i.3 Effect of adding poles or zeros on the root locus

```
// Illustration 6.3
// Effect of adding poles or zeros on the root locus
    of the system
// (section6-5). (fig 6-35)
// Interactive Program

// A MENU called "Add" will be added to the window

clear; clc;
xdel(winsid()); //close all windows

```

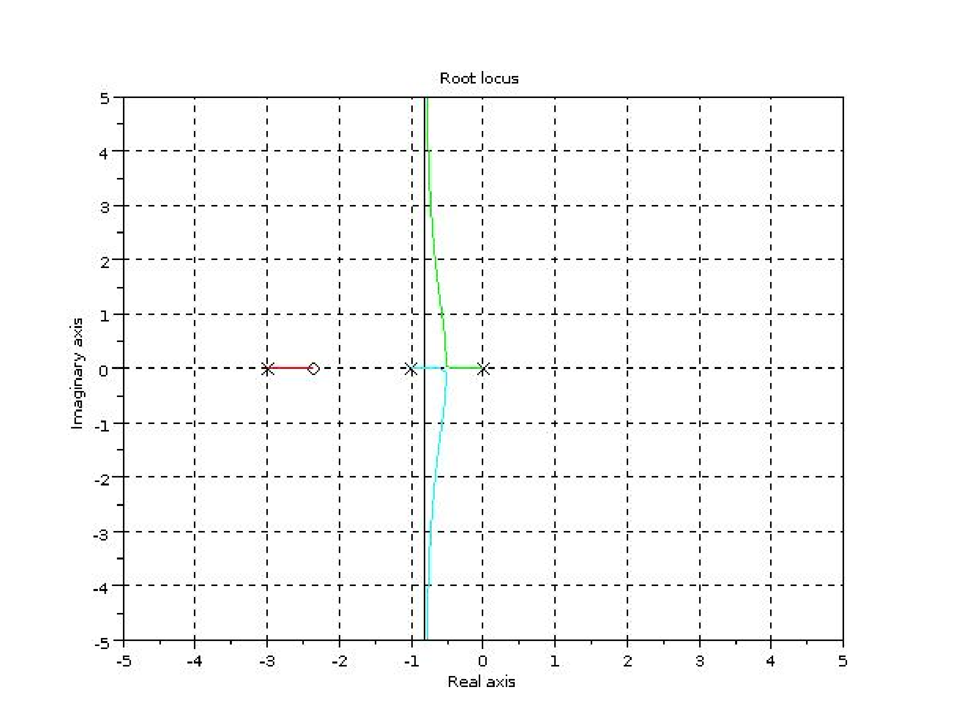

Figure 6.3: Effect of adding poles or zeros on the root locus

```
// please set the path
// cd "/<your code directory>/"
// exec("rootl.sci");

function J = add(n,H)

  z = locate(1,1);
  x = z(1);y = z(2);
  N = H.num;
  D = H.den;
  if abs(y) <= 0.2 then
    if n == 1 then  D = D * (s-x);
      else   N = N * (s-x);
    end
    zp = x;
   else
    if n == 1 then   D = D * (s^2 - 2*x*s + x^2 + y^2);
      else  N = N * (s^2 - 2*x*s + x^2 + y^2);
     end
    zp = x + %i * y;
   end
   J = syslin('c',N,D);
   draws(J);
   if(n == 1) then disp(zp,"p = "); else disp(zp,"z = ");end
   disp(J,"G = ");
endfunction

function draws(P)
  delete(gca());
  rootl(P,[-5 -5; 5 5],'Root locus'); //you can change the range :[-20,-20;20,20];

endfunction

// Main Program
s = %s;
```

```
N = 1;
D = s * (s + 1) * (s + 3);
G = syslin('c',1,D);
H = G;

draws(G);
addmenu(0,'Add',['Pole','zero','Reset']);
Add_0 = ['H = add(1,H)','H = add(2,H);','draws(G);H=
  G;'];

// place a zero close to the pole at -3
// first place it to the right then , to the left
// Then mover farther to the right.[-5 -5; 5 5]
```

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.a.6 `Root locus`

```
// Example A-6-6
// Root locus

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");

s = %s;
G = syslin('c',1,s * (s + 1) * (s^2 + 4*s + 13));
rootl(G,[-6 -5; 6 5],'Root locus plot for 1/ [s * (s
   + 1) * (s^2 + 4*s + 13]');

```

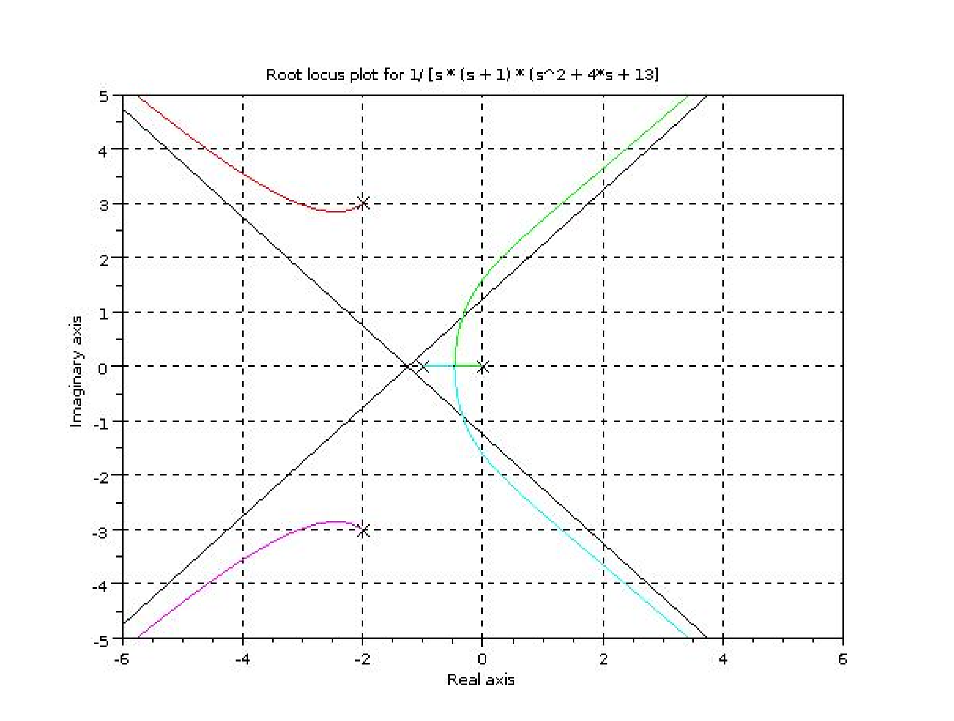

Figure 6.4: Root locus

```
// the same method may be employed to plot root loci
    in examples
// A-6-1,2,3,8,10,
// simply write the transfer function and choose
   suitable range
// [xmin ymin; xmax ymax]
```

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.a.13.1 `Lead Compensator Design Attempt 1`

```
// Example A-6-13-1
// Lead Compensator Design Attempt 1

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");
// exec("plotresp.sci");

s = %s;
G = syslin('c',1,s^2);
H = syslin('c',1,0.1*s + 1);

R = [-1 -1];
I = [1.73205 -1.73205];
dp = R(1) + %i*I(1);

subplot(1,2,1);
rootl(G*H,[-15 -15; 5 15],'Root locus plot for
    uncompensated system');
plot(R,I,'x');
angdef = 180 - phasemag(horner(G*H,dp));
disp(angdef,'angle deficiency =');

z = 1; // zero at -1;
p = 1.73205 / tand(90 - angdef) + 1 ;
Gc = (s + z) / (s + p);
disp(Gc,'lead compensator =');

Kc = abs(1/ horner(G*Gc*H,dp));
disp(Kc,'Kc =');
O = Kc*Gc*G*H;   disp(O,'open loop Transfer function
    =');
C = Kc*Gc*G /. H;      disp(C,'closed loop Transfer
    function =');
disp(roots(C.den),'closed loop poles =');
```

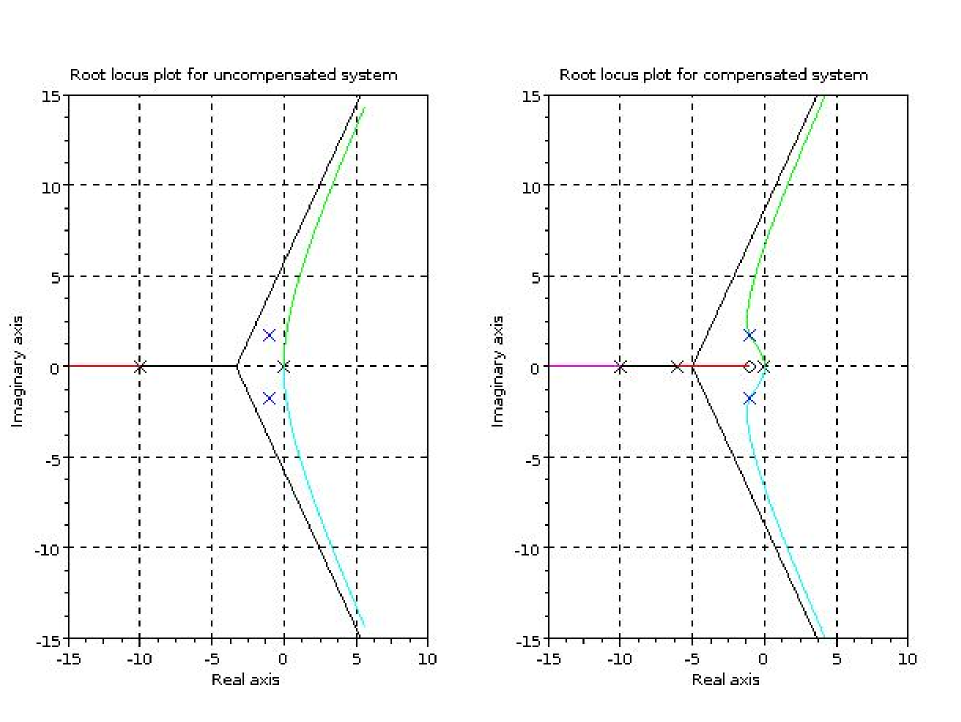

Figure 6.5: Lead Compensator Design Attempt 1

```

subplot(1,2,2);
rootl(0,[-15 -15; 5 15],'Root locus plot for
    compensated system');
plot(R,I,'x');

scf();
t = 0:0.05:10;
u = ones(1,length(t)); //step response
plotresp(u,t,C,'Unit step response');
xstring(1,0.95,'compensated system');
```

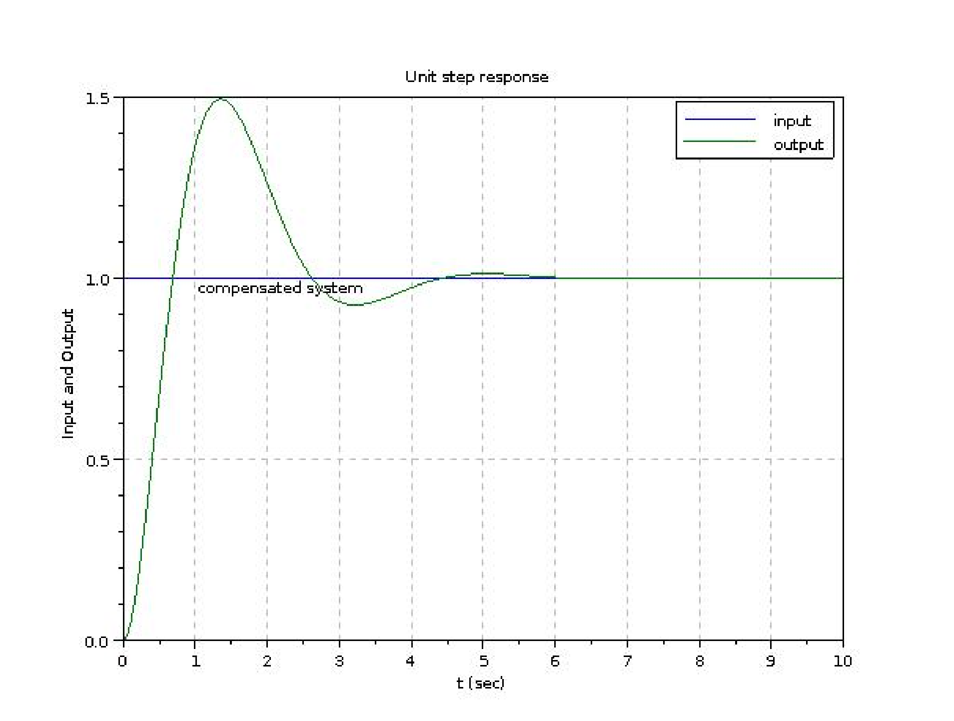

Figure 6.6: Lead Compensator Design Attempt 1

check Appendix AP 2 for dependency:

`plotresp.sci`

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.a.13.2 `Lead Compensator Design Attempt 2`

```
// Example A-6-13-2
// Lead Compensator Design Attempt 2

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");
// exec("plotresp.sci");

s = %s;
G = syslin('c',1,s^2);
H = syslin('c',1,0.1*s + 1);

R = [-1 -1];
I = [1.73205 -1.73205];
dp = R(1) + %i*I(1);

subplot(1,2,1);
rootl(G*H,[-15 -15; 5 15],'Root locus plot for
    uncompensated system');
plot(R,I,'x');
angdef = 180 - phasemag(horner(G*H,dp));
disp(angdef,'angle deficiency =');

```

```
z = 3; // zeros at -3;
p = 1.73205 / tand(40.89334 - angdef/2) + 1 ; disp(p,'p =');
Gc = ((s + z) / (s + p)) ^2;
disp(Gc,'lead compensator =');

Kc = abs(1/ horner(G*Gc*H,dp));
disp(Kc,'Kc =');
O = Kc*Gc*G*H;   disp(O,'open loop Transfer function =');
C = Kc*Gc*G /. H;      disp(C,'closed loop Transfer function =');
disp(roots(C.den),'closed loop poles =');

subplot(1,2,2);
rootl(O,[-15 -15; 5 15],'Root locus plot for compensated system');
plot(R,I,'x');

scf();
t = 0:0.05:10;
u = ones(1,length(t)); //step response
plotresp(u,t,C,'Unit step response');
xstring(1,0.95,'compensated system');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

check Appendix AP 7 for dependency:

`rootl.sci`

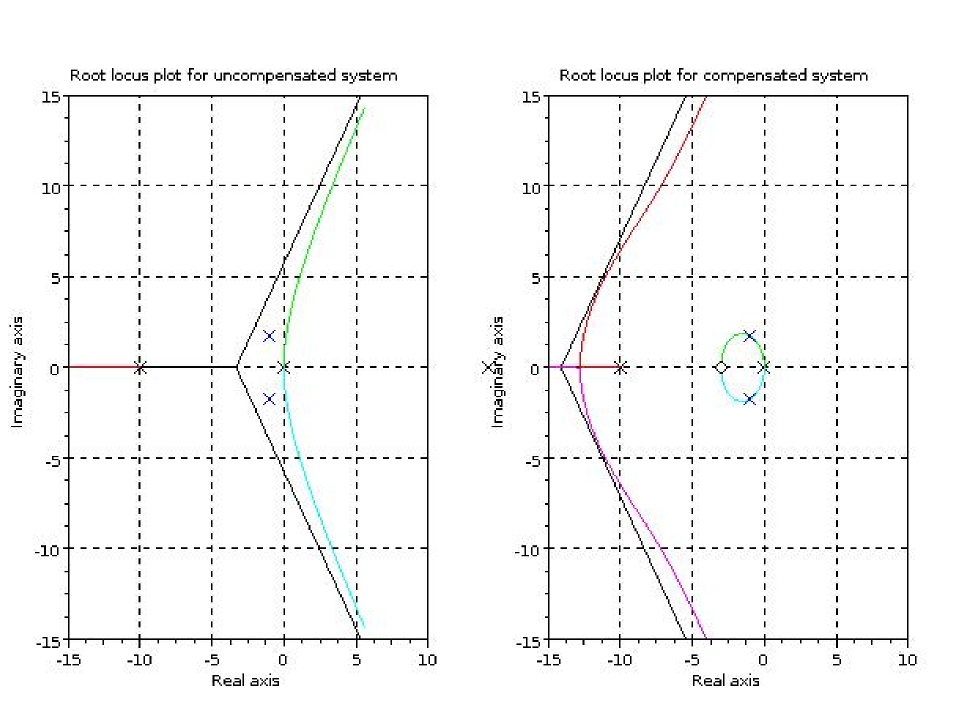

Figure 6.7: Lead Compensator Design Attempt 2

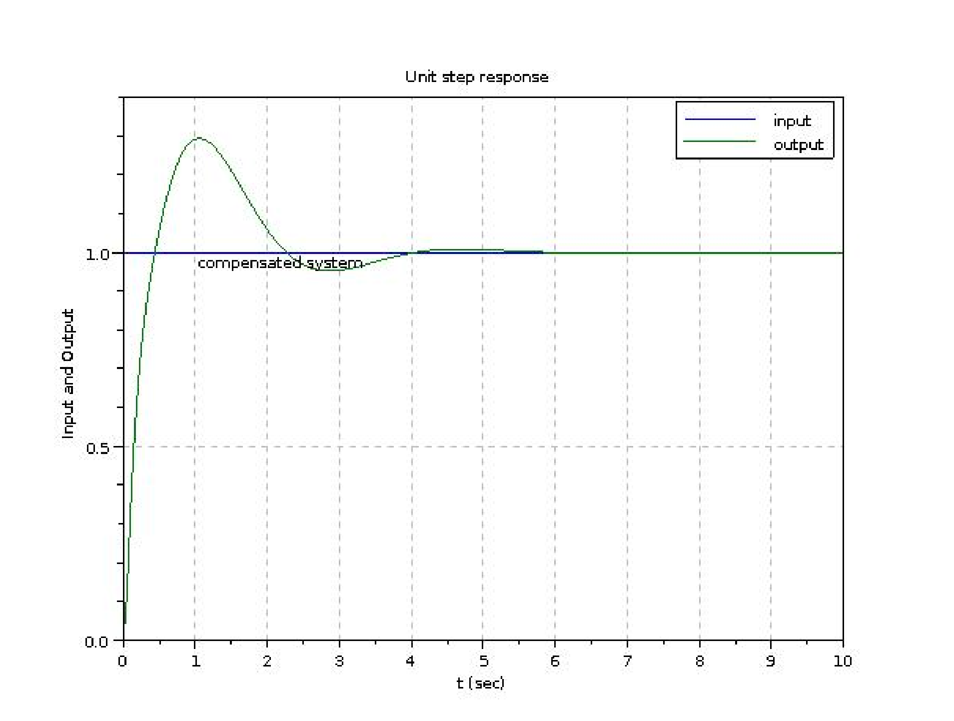

Figure 6.8: Lead Compensator Design Attempt 2

Scilab code Exa 6.a.17 Design of lag lead compensator

```
// Example A-6-17
// Design of lag lead compensator

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");
// exec("plotresp.sci");

s = %s;
G = syslin('c',1 ,s * (s + 1) * (s + 5));

Kv = 50;              // desired velocity constant
disp(horner(s*G,0),'Kv (uncompensated system) = ');

// designing lead part
Kc = Kv /abs(horner(s*G,0))
z1 = 1     //to cancel the pole s = -1 of the plant

_beta = 16.025; disp(_beta,'beta =');
x = 1.9054    // beta and x are found analytically

dp = -x + sqrt(3)*%i*x
R = [-x -x]; I = [imag(dp) -imag(dp)];
p1 = z1 * _beta

Gc1 =Kc * (s + z1)/(s + p1); disp(Gc1,'Lead
    compensator Gc1 =');

// Lag compensator design
p2 = 0.01  //say
z2 = p2 * _beta
Gc2 = (s + z2)/(s + p2);
```

```
disp(Gc2,'Lag compensator Gc2 =');
disp(abs(horner(Gc2,dp)),'magnitude contribution of
    lag part =');
disp(phasemag(horner(Gc2,dp)),'angle contribution of
     lag part =');
// these are acceptable

Gc = Gc1 * Gc2
H = G * Gc ;          // compensated system
H = syslin('c',numer(H),denom(H));

subplot(1,2,1);
rootl(G,[-20 -15; 10 15],'Uncompensated system');
plot(R,I,'x');
xgrid(color('gray'));
subplot(1,2,2);
rootl(H,[-20 -15; 10 15],'Compensated system');
plot(R,I,'x');
xgrid(color('gray'));
xstring(R(1),I(1),'Desired closed loop poles');

G1 = syslin('c',G /. 1);
C = syslin('c',H /. 1);       // final closed loop
    system
disp(C,'closed loop system =');
disp(roots(C.den),'closed loop poles = ');
disp(horner(s*H,0),'velocity error constant Kv =')

scf();
subplot(2,1,1);
t = 0:0.05:10;
u = ones(1,length(t));
plotresp(u,t,G1,'');
plotresp(u,t,C,'Unit step response');
xstring(1,0.1,'uncompensated system');
xstring(0.7,1.12,'compensated system');

subplot(2,1,2);
```

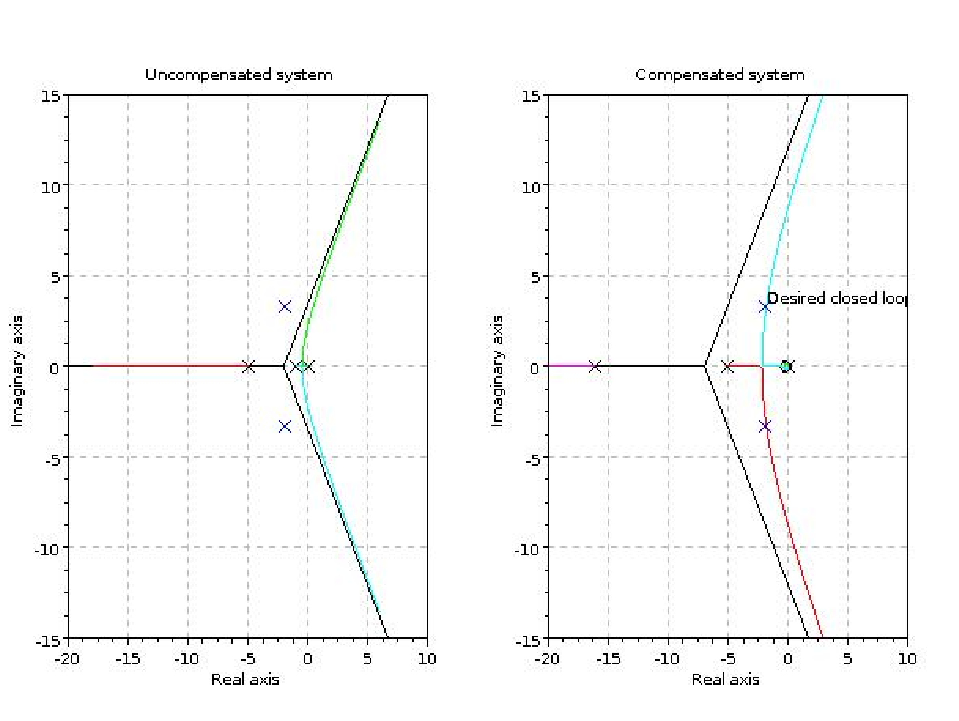

Figure 6.9: Design of lag lead compensator

```
plotresp(t,t,G1,'');
plotresp(t,t,C,'Unit ramp response');
xstring(3,0.9,'uncompensated system');
xstring(0.7,2,'compensated system');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

check Appendix AP 7 for dependency:

`rootl.sci`

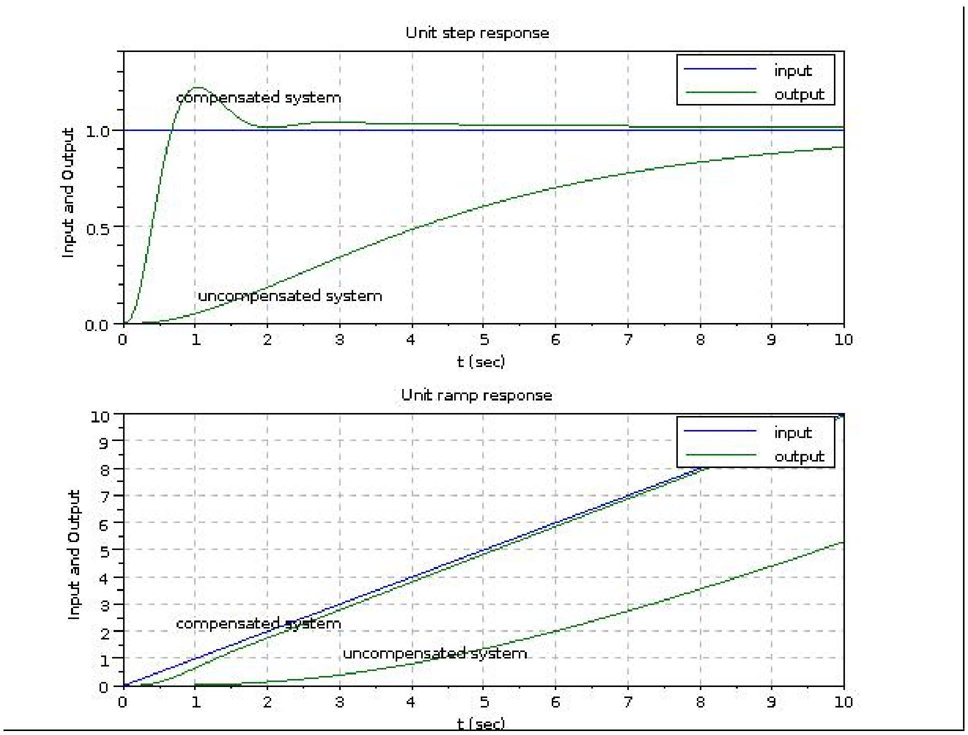

Figure 6.10: Design of lag lead compensator

Scilab code Exa 6.a.18 Design of a compensator for a highly oscillactory system

```
// Example A-6-18
// Design of a compensator for an highly
   oscillactory system

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");
// exec("plotresp.sci");

s = %s;
G = syslin('c',2*s + 0.1,s * (s^2 + 0.1*s + 4));

R = [-2 -2];
I = 2*sqrt(3) * [1 -1];
dp = R(1) + %i*I(1)

// Cancel the zero at -0.1
Gc2 = (s + 4)/(2*s + 0.1)
G1 = G*Gc2

angdef = 180 - phasemag(horner(G1,dp));
disp(angdef,'angle deficiency =')

// Designing two lead comensators in series
angdefby2 = angdef / 2
z = 2 // say
p = 2 + 2 * sqrt(3) * cotd(90 - angdefby2)

```

```
Gc1 = ((s + z)/(s + p))^2
G2 = Gc1 * G1;
Kc = 1 / abs(horner(G2,dp))
Gc = Kc * Gc1 * Gc2

H = Kc * G2;  disp(H,'Gc*G = ');
C = H /. 1;   disp(C,'closed loop Transfer function =');
disp(roots(C.den),'closed loop poles =');

subplot(1,2,1);
rootl(G,[-15 -15; 15 15],'Root locus plot for uncompensated system');
plot(R,I,'x');
xgrid(color('gray'));
subplot(1,2,2);
rootl(H,[-15 -15; 15 15],'Root locus plot for compensated system');
plot(R,I,'x');
xgrid(color('gray'));

scf();
subplot(2,1,1);
t = 0:0.02:5;
u = ones(1,length(t)); //step response
plotresp(u,t,C,'Unit step response of the compensated system');

subplot(2,1,2);
t = 0:0.02:8;
plotresp(t,t,C,'Unit step response of the compensated system');
```

check Appendix AP 2 for dependency:

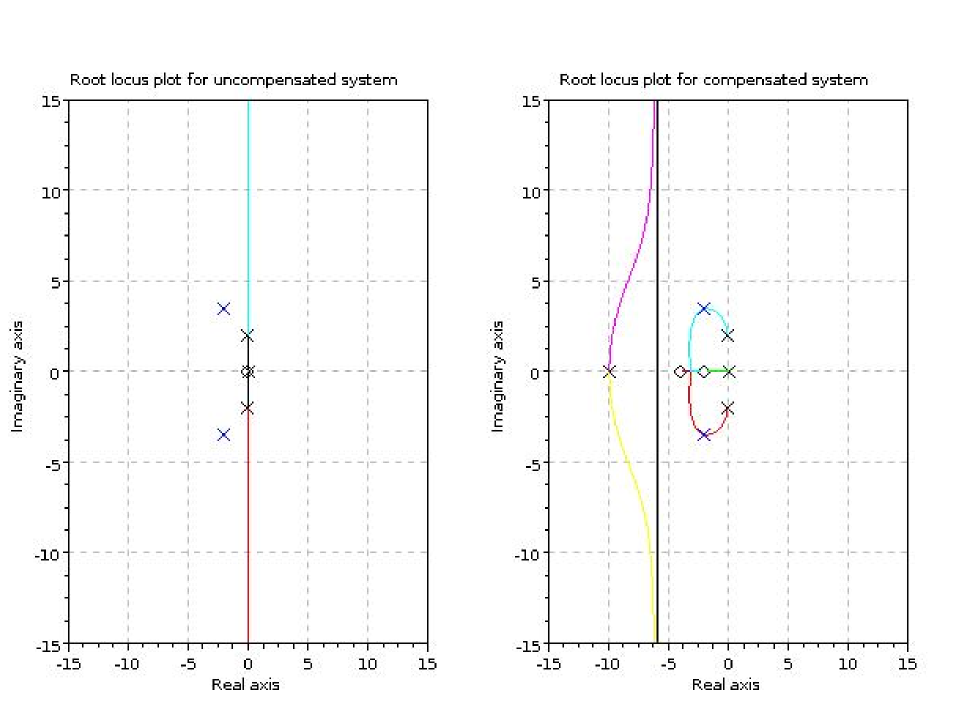

Figure 6.11: Design of a compensator for a highly oscillactory system

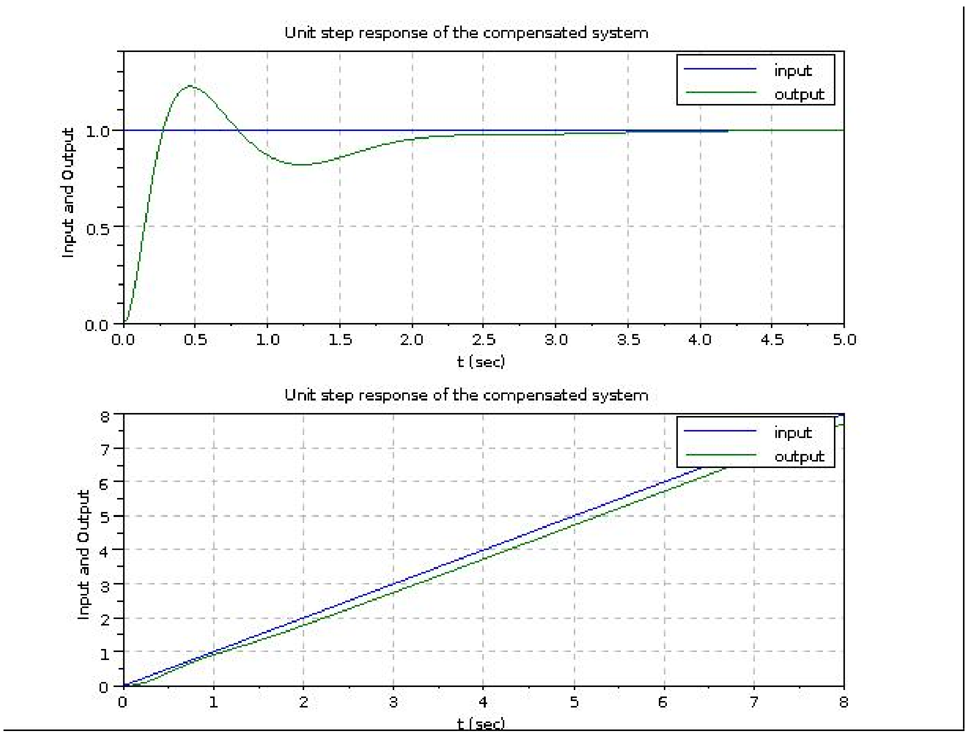

Figure 6.12: Design of a compensator for a highly oscillactory system

`plotresp.sci`

check Appendix AP 7 for dependency:

`rootl.sci`

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.1 `Root Locus`

```
// Example 6-1
// Root Locus

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");

s = %s;
D = s * (s + 1) * (s + 2);
H = syslin('c',1,D);

rootl(H,[-4 -3; 2 3],'Root locus of G(s) = 1/(s*(s +
    1)*(s + 2))');
```

Scilab code Exa 6.2 `Root Locus`

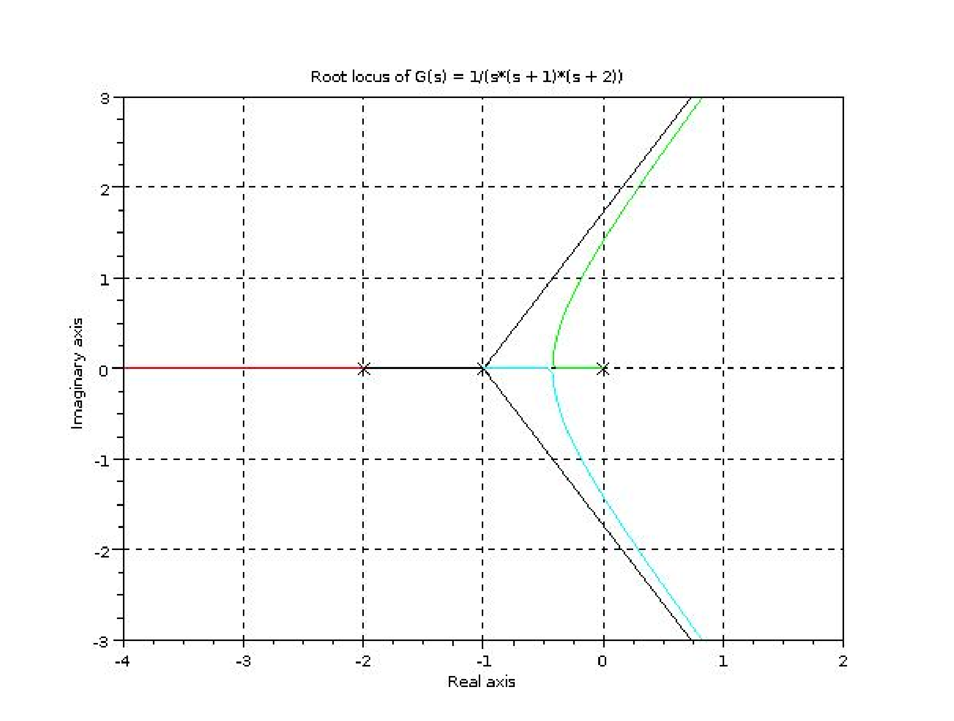

Figure 6.13: Root Locus

```
// Example 6-2
// Root Locus

clear; clc;
xdel(winsid()); //close all windows

s = %s;
H = syslin('c',s + 2, s^2 + 2*s + 3);

evans(H,10);
xgrid();
a = gca();
a.box = "on";
a.data_bounds = [-6 -3; 2 3];
a.children(1).visible = 'off';
xtitle('Root locus of G(s) = (s + 2)/ (s^2 + 2*s + 3)');
```

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.3 Root Locus

```
// Example 6-3
// Root locus

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");

```

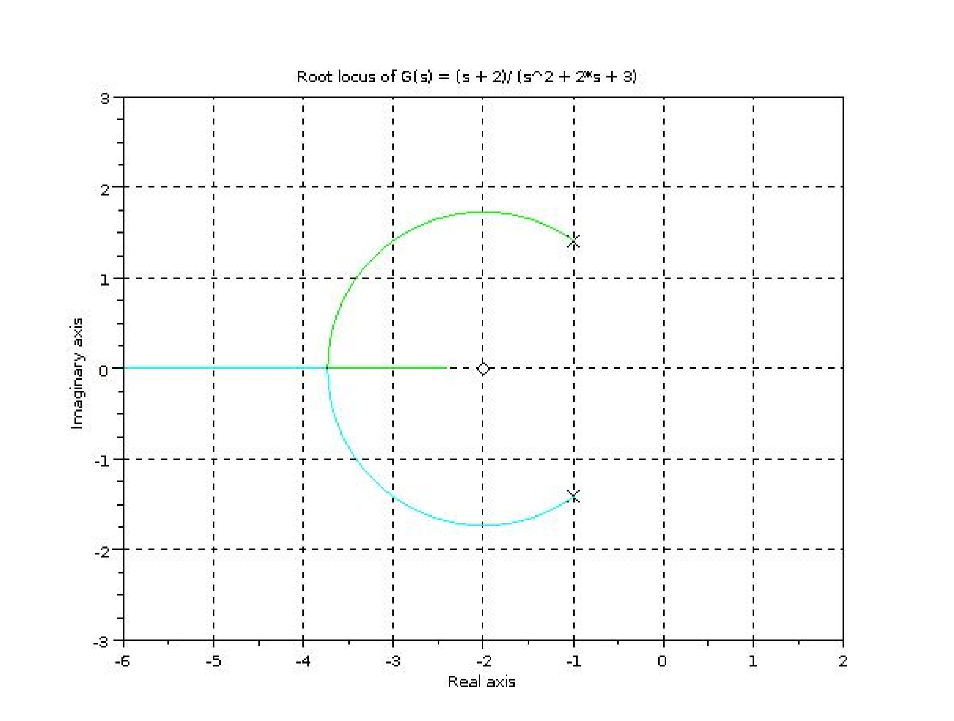

Figure 6.14: Root Locus

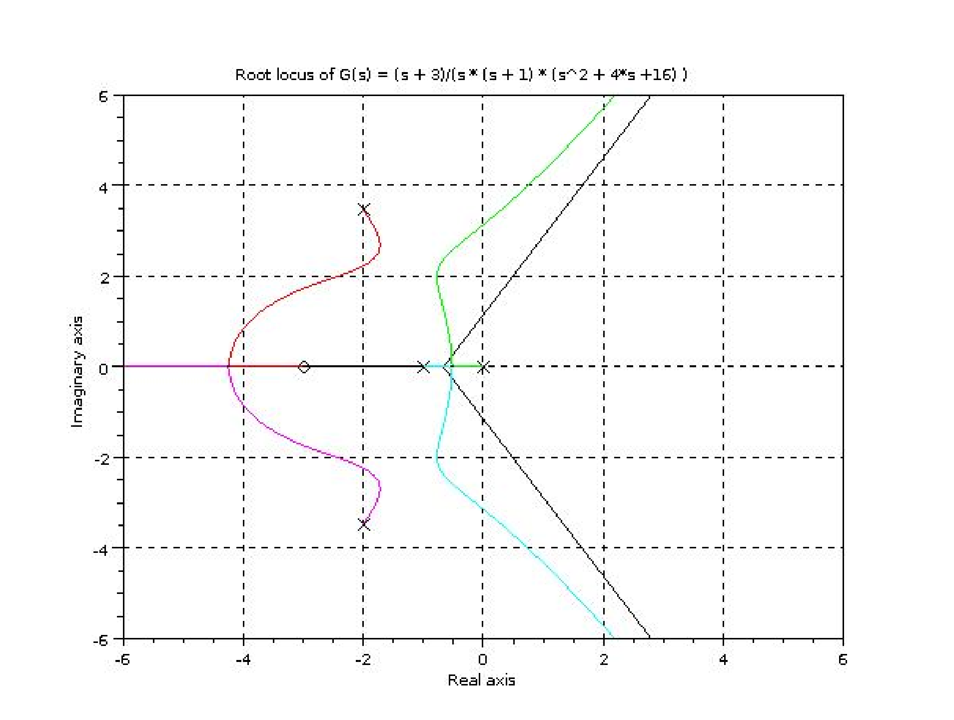

Figure 6.15: Root Locus

```
s = %s;
N = s + 3;
D = s * (s + 1) * (s^2 + 4*s +16);
H = syslin('c',N,D);
disp( roots(D) ,'open loop poles = ');
disp( roots(N) ,'open loop zeros = ');

rootl(H,[-6 -6; 6 6],'Root locus of G(s) = (s + 3)/(
    s * (s + 1) * (s^2 + 4*s +16) )');
```

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.4 Root Locus

```
// Example 6-4
// Root locus

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");

s = %s;
D = s*(s + 0.5)*(s^2 + 0.6*s + 10);
H = syslin('c',1,D);
disp(roots(D),'open loop poles =');

rootl(H,[-6 -6; 6 6],'Root locus of G(s) = 1/(s*(s
    + 0.5)*(s^2 + 0.6*s + 10)');
```

check Appendix AP 7 for dependency:

rootl.sci

Scilab code Exa 6.5 Root locus of system in state space

```
// Example 6_5
// Root locus of system in state space

clear; clc;
```

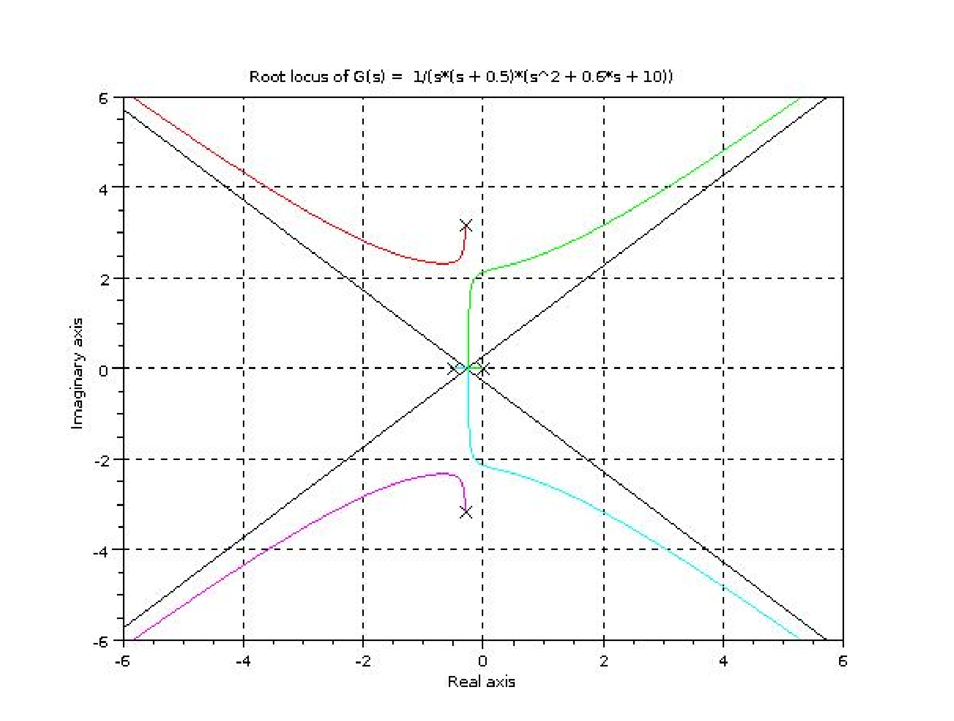

Figure 6.16: Root Locus

```
xdel(winsid());  //close all windows

// please edit the path
// cd "/<your code directory>/";
exec("rootl.sci");

A = [0 1 0; 0 0 1; -160 -56 -14];
B = [0; 1; -14];
C = [1 0 0];
D = [0];
G = syslin('c',A,B,C,D);
H = clean(ss2tf(G));
disp(H,' transfer function = ');

rootl(G,[-20 -20; 20 20],'Root locus plot of State
    Space model');
```

Scilab code Exa 6.6.1 Design of a lead compensator using root locus

```
// Example 6-6-1
// Design of a lead compensator using root locus


clear; clc;
xdel(winsid());  //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");

s = %s;
G = syslin('c',10 , s*(s+1) ); //open loop system

```

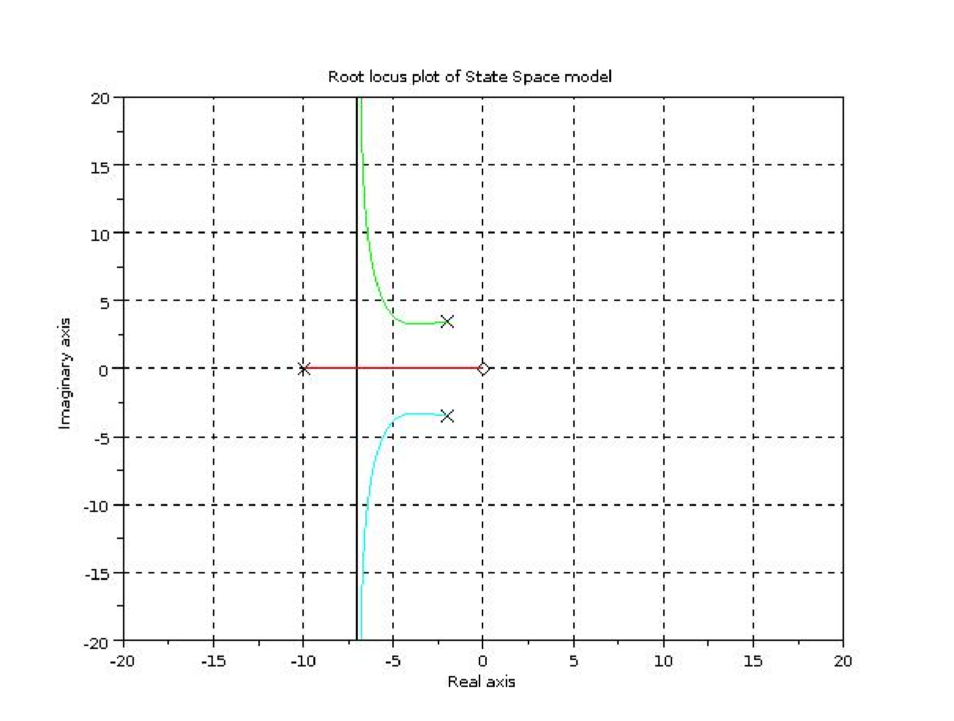

Figure 6.17: Root locus of system in state space

```
R = [ -1.5 -1.5];
I = [2.5981 -2.5981];  // desired closed loop poles
dp = R(1) + %i*I(1);

rootl(G,[-5 -5; 1 5],'Uncompensated system');
xgrid(color('gray'));
plot(R,I,'x');     // A gain adjustment is not enough as the
                   // desired poles do not lie on the roor locus

[phi1 db] = phasemag(horner(G,dp));
angdef = 180 - phi1;
disp(angdef,'Angle deficiency = ');

// Lead compensator for Maximum Kv
// here we will find the pole-zero of the compensator
// using the prescirbed method

[phi2 dbi] = phasemag(dp);
angOPA = phi2;
angPOD = 180 - phi2;
angOPD = (angOPA - angdef) / 2;
angOPC = (angOPA + angdef) / 2;

angPDO = (180 - angPOD - angOPD);
angPCO = (180 - angPOD - angOPC);

//using the sine rule of triangles
DO = sind(angOPD) * abs(dp) / sind(angPDO);
CO = sind(angOPC) * abs(dp) / sind(angPCO);

Gc = (s + DO)/(s + CO);
disp(Gc ,'compensator = ');
H = G.num * Gc / G.den  ;          // compensated system
H = syslin('c',numer(H),denom(H));
```

```
scf();
rootl(H,[-5 -5; 1 5],'Compensated system');
xgrid(color('gray'));
plot(R,I,'x');

// Final system passes through the desired poles
// required gain for the system
Kc = abs(1 / horner(H,dp));
disp(Kc,'required gain Kc = ');
C = H*Kc /. 1;       // final closed loop system
disp(C,'closed loop system =');
disp(roots(C.den),'closed loop poles = ');
disp(horner(s*H*Kc,0),'velocity error constant Kv ='
   )
```

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.6.2 `Step and ramp response of lead compensated systems`

```
// Example 6-6-2
// Step and ramp response of lead compensated
   systems

clear; clc;
xdel(winsid());  //close all windows

function Gc = leadcomp(Kc,z,p);
  Gc = Kc* ((s + z)/(s + p));
endfunction
```

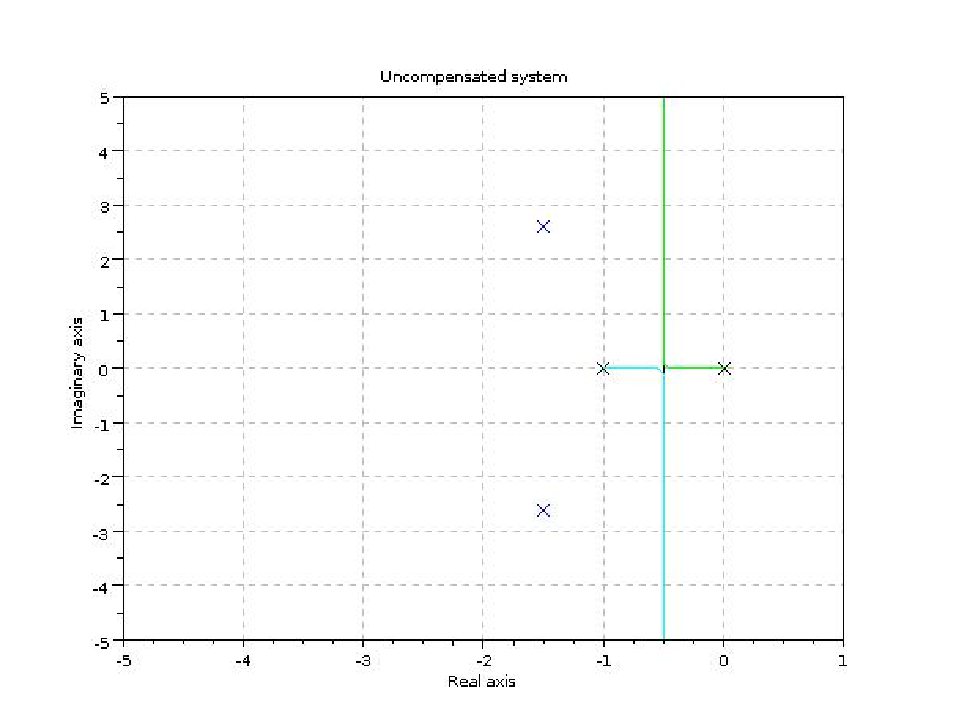

Figure 6.18: Design of a lead compensator using root locus

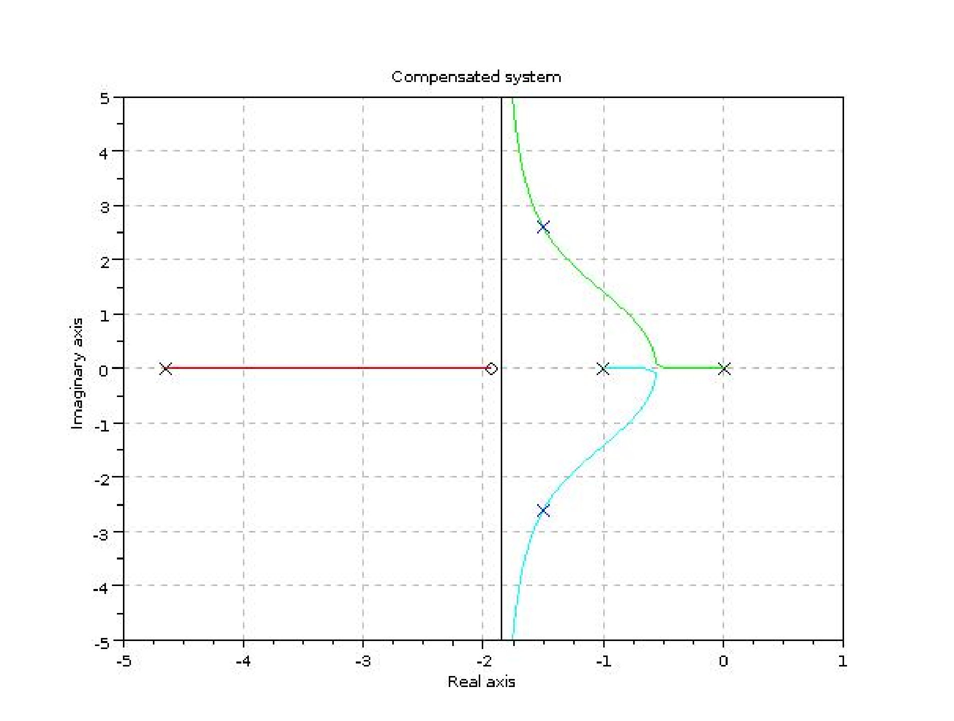

Figure 6.19: Design of a lead compensator using root locus

```
function plotall(u,t,text)
  y   = csim(u,t,H );
  yc1 = csim(u,t,H1);
  yc2 = csim(u,t,H2);

  plot(t,y,t,yc1,t,yc2);
  xgrid(color('gray'));
  xtitle(text + ' Response of compensated and
     uncompensated systems','t sec','Output');
  legend('Uncompensated System','Compensated System
     Method 1','Compensated System Method 2');
endfunction

s = %s;
G = 10 / ( s*(s+1) ); //open loop system
Gc1 = leadcomp(1.2292,1.9373,4.6458);
Gc2 = leadcomp(0.9,1,3);

H =  syslin('c',G /. 1);
H1 = syslin('c', ( G * Gc1) /. 1);
H2 = syslin('c', ( G * Gc2) /. 1);

 t = 0:0.05:5;
 u = ones(1,length(t));
 plotall(u,t,'Step');scf();
 t = 0:0.05:9;
 plotall(t,t,'Ramp');
  plot(t,t,'k');
```

Scilab code Exa 6.7.1 Design of a lag compensator using root locus

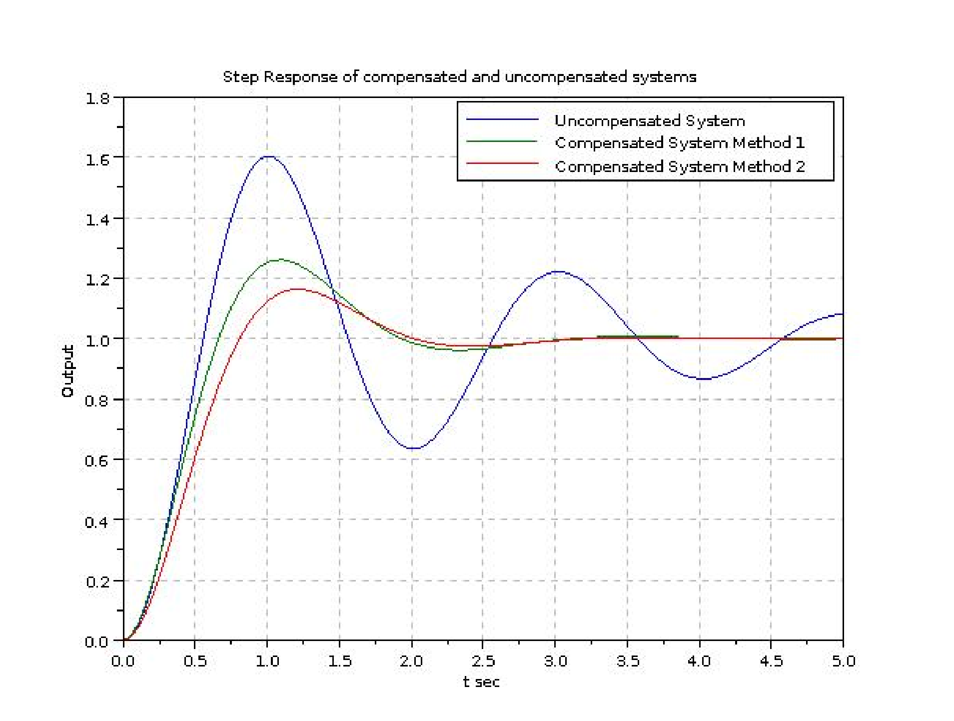

Figure 6.20: Step and ramp response of lead compensated systems

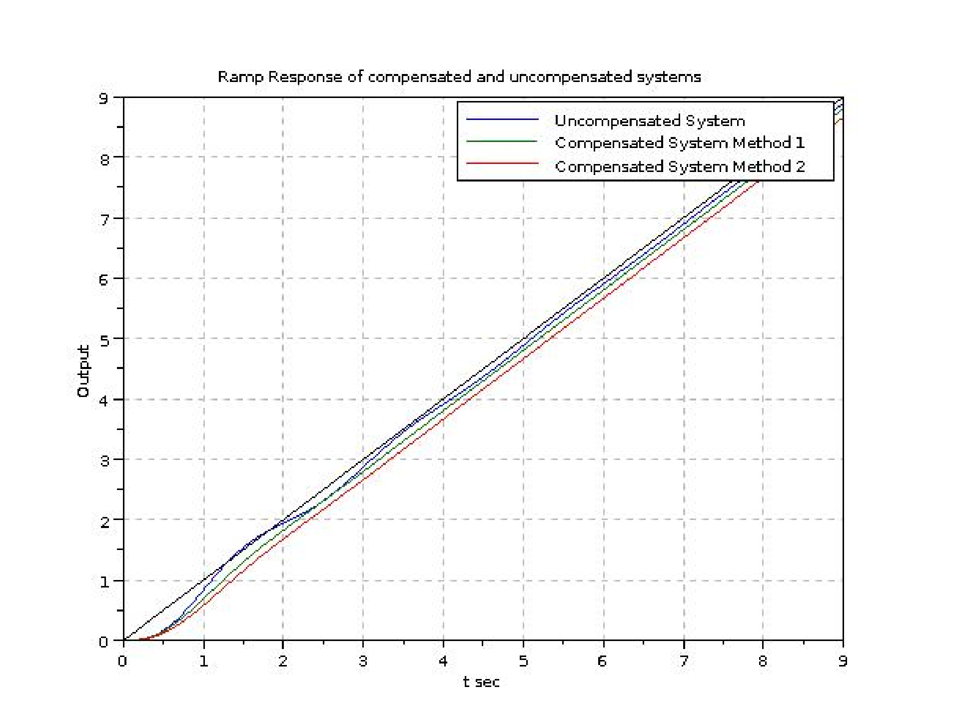

Figure 6.21: Step and ramp response of lead compensated systems

```
// Example 6-7-1
// Design of a lag compensator using root locus


clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");

s = %s;
G = syslin('c',1.06 , s*(s+1)*(s+2)); //open loop
    system
R = [ -0.31 -0.31];
I = [0.55 -0.55]; // desired closed loop poles
dp = R(1) + %i*I(1);
disp(roots(G.den + 1.06),'Closed loop poles (
    uncompensated)=');
disp(horner(s*G,0),'Kv (uncompensated system = ');

rootl(G,[-3 -2; 1 2],'');
plot(R,I,'x');

// Lag compensator for Kv = 5 sec.

_beta = 10; // taking beta as 10
z = 0.05;
p = z / _beta;

Gc = (s + z)/(s + p);
disp(Gc ,'compensator = ');
H = G.num * Gc / G.den ;         // compensated
    system
H = syslin('c',numer(H),denom(H));

rootl(H,[-3 -2; 1 2],'Uncompensated and Compensated
    system');
```

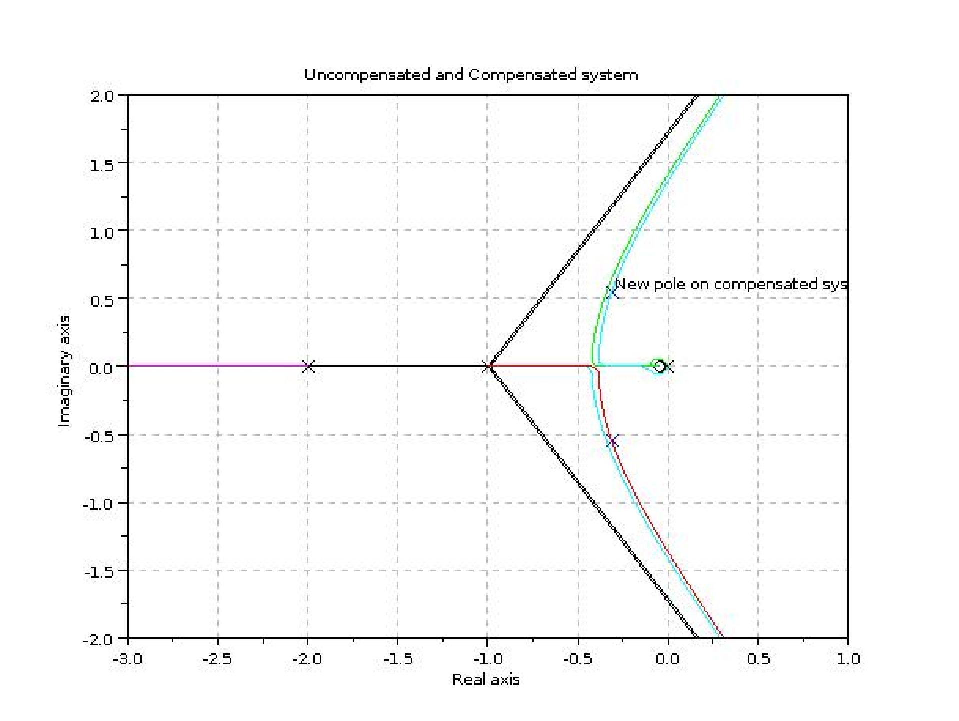

Figure 6.22: Design of a lag compensator using root locus

```
xgrid(color('gray'));
xstring(R(1),I(1),'New pole on compensated sys');

Kc = abs(1 / horner(H,dp));
disp(Kc,'required controller gain Kc = ');
C = H*Kc /. 1;       // final closed loop system
disp(C,'closed loop system =');
disp(roots(C.den),'closed loop poles = ');
disp(horner(s*H*Kc,0),'velocity error constant Kv ='
   )
```

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.7.2 Step and ramp response of lag compensated system

```
// Example 6-7-2
// Step and ramp response of lag compensated system

clear; clc;
xdel(winsid());  //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

s = %s;
G = 1.06 / (s * (s + 1) * (s + 2));

Kc = 0.9956;
z  = 0.05;
p  = 0.005;
Gc = Kc * (s + z)/(s + p);
GGc = G*Gc;

H = syslin('c',G /. 1);
Hc = syslin('c',GGc /. 1);

t = 0:0.5:40;
u1 = ones(1,length(t)); //step response

subplot(2,1,1);plotresp(u1,t,H,'');
plotresp(u1,t,Hc,'Unit step response');
xstring(5,0.9,'uncompensated system');
xstring(0.1,1.2,'compensated system');

t = 0:0.5:50;
u2 = t;                  //ramp response
```

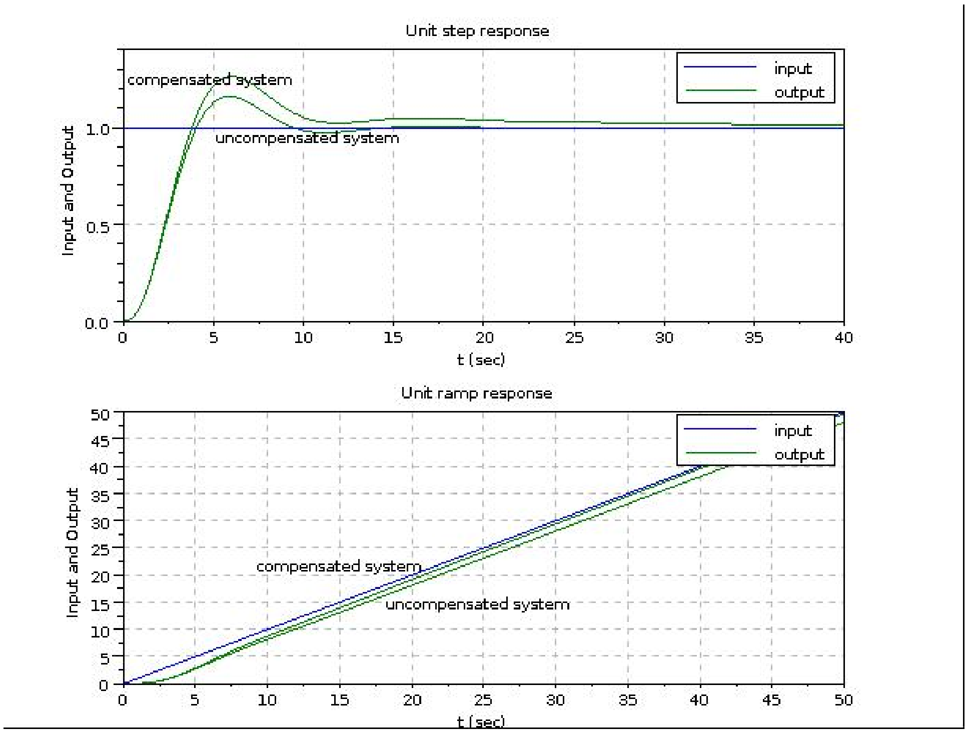

Figure 6.23: Step and ramp response of lag compensated system

```
subplot(2,1,2);plotresp(u2,t,H,'');
plotresp(u2,t,Hc,'Unit ramp response');
xstring(18,13,'uncompensated system');
xstring(9,20,'compensated system');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 6.8.1 `Design of a lag lead compensator using root locus`

```
// Example 6-8-1
```

```
// Design of a lag lead compensator using root locus 1
// zeta ~= gamma (not equal to)

clear; clc;
xdel(winsid()); //close all windows
// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");

s = %s;
G = syslin('c',4 , s * (s + 0.5)); //open loop system

Kv = 80;            // desired velocity constant
wn = 5;             // desired natural frequency and damping
_zeta = 0.5;
sigma = -1*wn * _zeta;
wd = wn * sqrt(1 - _zeta^2);

dp = sigma + %i*wd;     // desired closed loop poles
disp(roots(G.den + 4),'Closed loop poles (uncompensated)=');
disp(horner(s*G,0),'Kv (uncompensated system = ');

rootl(G,[-5 -2; 1 2],'Uncompensated system');
xgrid(color('gray'));
plot([sigma sigma],[wd -wd],'x');
xstring(sigma,wd,'Desired CL poles');

// Designing Lead Part
[phi1 db] = phasemag(horner(G,dp));
angdef = 180 - phi1;
disp(angdef,'Angle deficiency = ');

z1 = 0.5  //Make the lead compensator zero cancel the system zero
```

```
// To determin p1;
// Gc1 = [0.5 +(−2.5 + 4.33j)] / [(p1 −2.5) + 4.33j]
[theta m2] = phasemag(-2.0 + 4.33*%i);
p1 = 2.5 + 4.33*cotd(theta - angdef);  // so that it contributes 'angdef'

Gc1 = (s + z1)/(s + p1);       disp(Gc1,'Lead compensator Gc1 =');
_gamma = p1 / z1;              disp(_gamma,'gamma = ');
Kc = abs(1/horner(G*Gc1,dp)); disp(Kc,'Kc = ');

// Lag compensator design
_beta = Kv * _gamma / Kc / horner(s*G,0); disp(_beta,'beta');

T2 = 5;  //say
z2 = 1 / T2; p2 = z2 / _beta;
Gc2 = (s + z2)/(s + p2);
disp(Gc2,'Lag compensator Gc2 =');
disp(abs(horner(Gc2,dp)),'magnitude contribution of lag part =');
disp(phasemag(horner(Gc2,dp)),'angle contribution of lag part =');
// these are acceptable

Gc = Kc*Gc1*Gc2;
disp(Gc,'final lag lead controller = ');
scf()
rootl(Gc*G,[-5 -2; 1 2],'Compensated system');
xgrid(color('gray'));
plot([sigma sigma],[wd -wd],'x');

C = Gc*G /. 1;
disp(C,'closed loop system =');
disp(roots(C.den),'closed loop poles = ');
disp(horner(s*Gc*G,0),'velocity error constant Kv =')
```

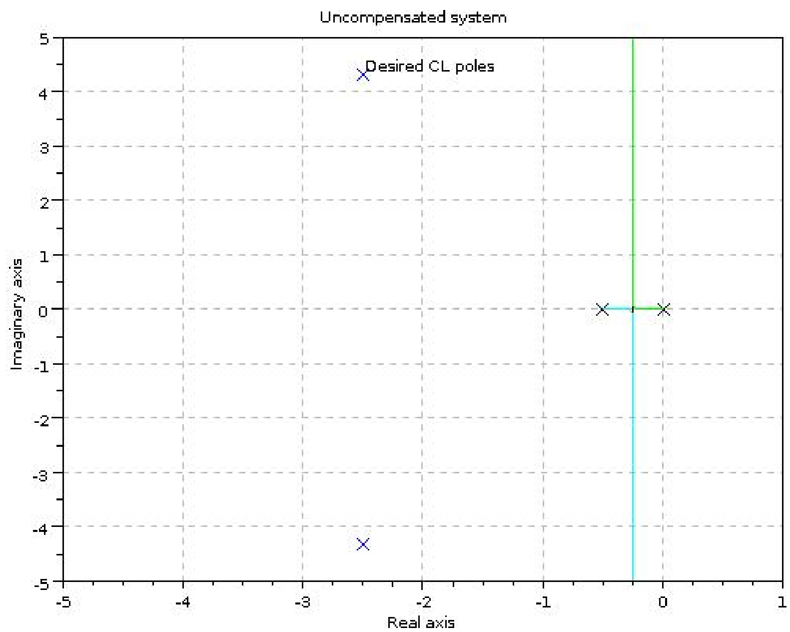

Figure 6.24: Design of a lag lead compensator using root locus

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.8.2 `Evaluating Lag Lead compensated system`

```
// Example 6-8-2
// Evaluating Lag Lead compensated system

```

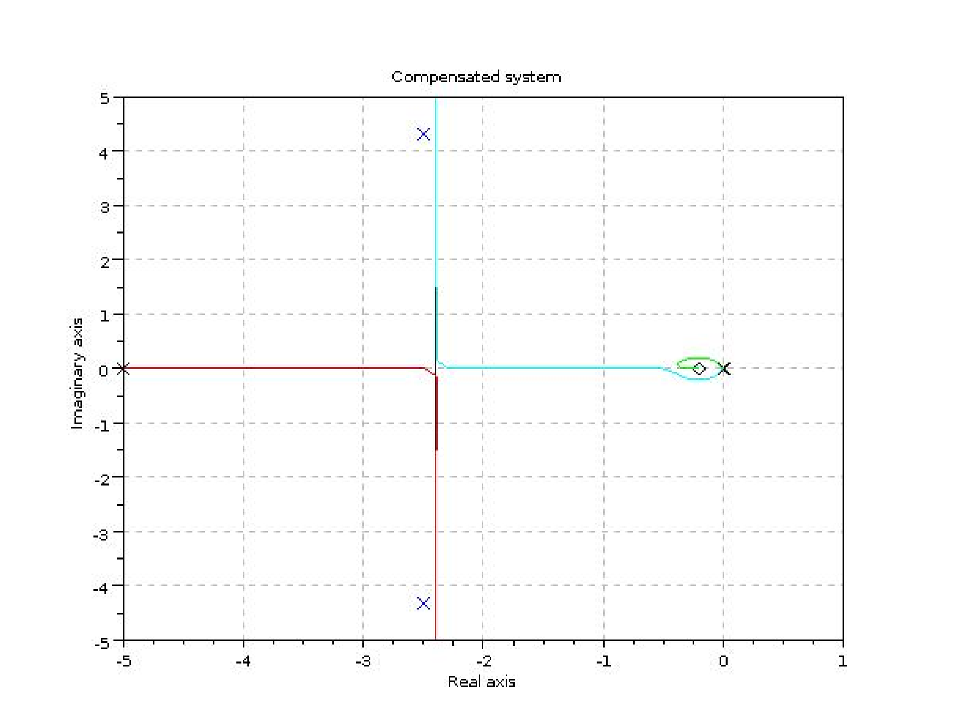

Figure 6.25: Design of a lag lead compensator using root locus

```
clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

s = %s;
G = 4 / (s * (s + 0.5));

Gc = 6.25 * (s + 0.5) * (s + 0.2) / (s + 5) / (s + 0.125);
GGc = G*Gc;

H = syslin('c',G /. 1);
Hc = syslin('c',GGc /. 1);

t = 0:0.05:20;
u1 = ones(1,length(t)); //step response
plotresp(u1,t,H,'');
plotresp(u1,t,Hc,'Unit step response');
xstring(0.5,1.7,'uncompensated system');
xstring(1,0.95,'compensated system');

scf()
t = 0:0.05:10;
plotresp(t,t,H,'');
y2 = plotresp(t,t,Hc,'Unit ramp response');a = gca()
delete(a.children(2)); // deleting the drawn graph and redrawing
// with a different colour
plot(t,y2,'r');
legend('ramp input','uncompensated system','compensated system');
```

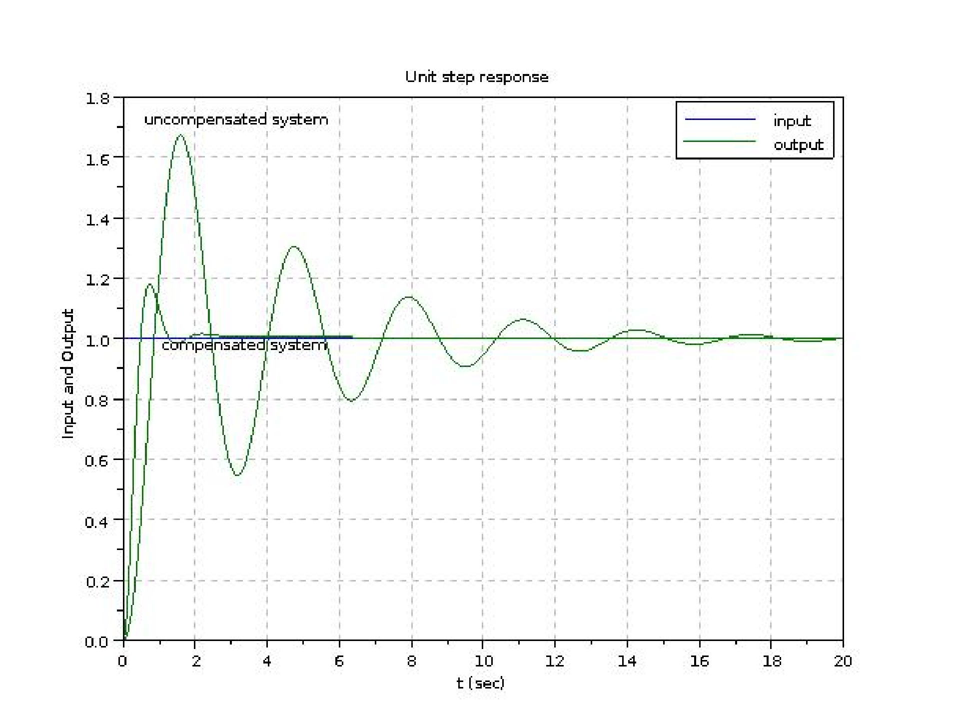

Figure 6.26: Evaluating Lag Lead compensated system

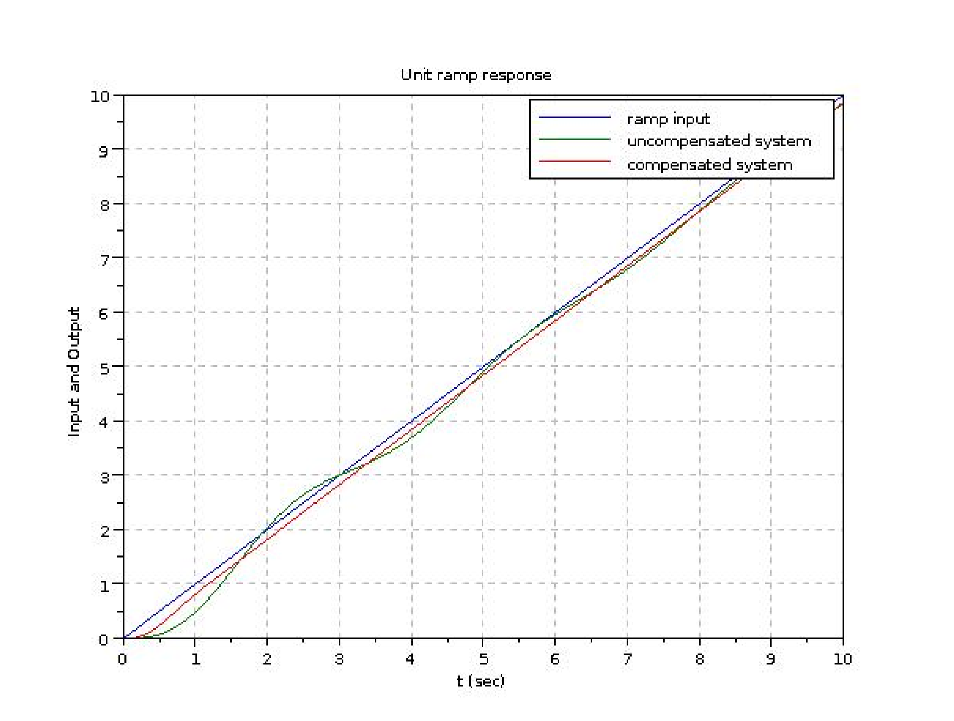

Figure 6.27: Evaluating Lag Lead compensated system

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 6.9.1 `Design of lag lead compensator using root locus 2`

```
// Example 6-9-1
// Design of a lag lead compensator using root locus
    2
// gamma = beta case

```

```
clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");

s = %s;
G = syslin('c',4 , s * (s + 0.5)); //open loop system

Kv = 80;             // desired velocity constant
wn = 5;              // desired natural frequency and damping
_zeta = 0.5;
sigma = -1*wn * _zeta;
wd = wn * sqrt(1 - _zeta^2);
dp = sigma + %i*wd;     // desired closed loop poles
disp(roots(G.den + 4),'Closed loop poles (uncompensated)=');
disp(horner(s*G,0),'Kv (uncompensated system = ');

rootl(G,[-5 -2; 1 2],'Uncompensated system');
xgrid(color('gray'));
plot([sigma sigma],[wd -wd],'x');
xstring(sigma,wd,'Desired CL poles');

// Designing Lead Part
Kc = Kv / horner(s*G,0);       disp(Kc,'Kc = ');
z1 = 2.38;      //z1 and p1 determinded graphically
p1 = 8.34;
T1 = 1 / z1;                   disp(T1,'T1');
_beta = T1 * p1;               disp(_beta,'beta =');

Gc1 =Kc * (s + z1)/(s + p1); disp(Gc1,'Lead compensator Gc1 =');

// Lag compensator design
```

```
T2 = 10;  //say
z2 = 1 / T2; p2 = z2 / _beta;
Gc2 = (s + z2)/(s + p2);
disp(Gc2,'Lag compensator Gc2 =');
disp(abs(horner(Gc2,dp)),'magnitude contribution of
    lag part =');
disp(phasemag(horner(Gc2,dp)),'angle contribution of
     lag part =');
// these are acceptable

Gc = Gc1*Gc2;
disp(Gc,'final lag lead controller = ');
scf()
rootl(Gc*G,[-5 -2; 1 2],'Compensated system');
xgrid(color('gray'));
plot([sigma sigma],[wd -wd],'x');

C = Gc*G /. 1;
disp(C,'closed loop system =');
disp(roots(C.den),'closed loop poles = ');
disp(horner(s*Gc*G,0),'velocity error constant Kv ='
    )
disp(dp,'desired poles =');
```

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.9.2 `Evaluating Lag Lead compensated system`

```
// Example 6-9-2
// Evaluating Lag Lead compensated system
```

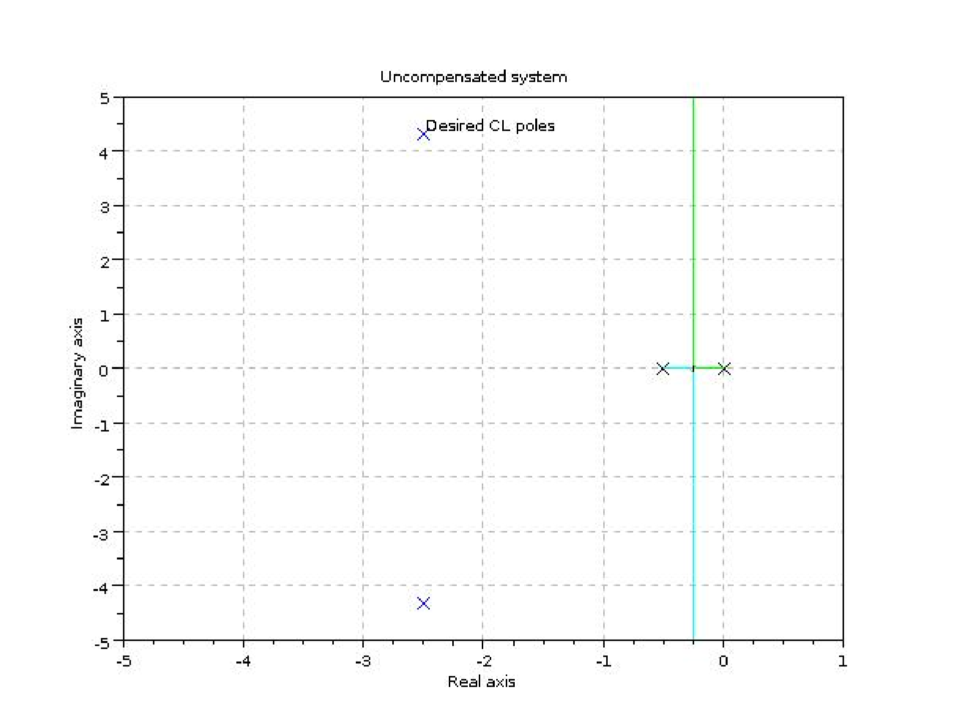

Figure 6.28: Design of lag lead compensator using root locus 2

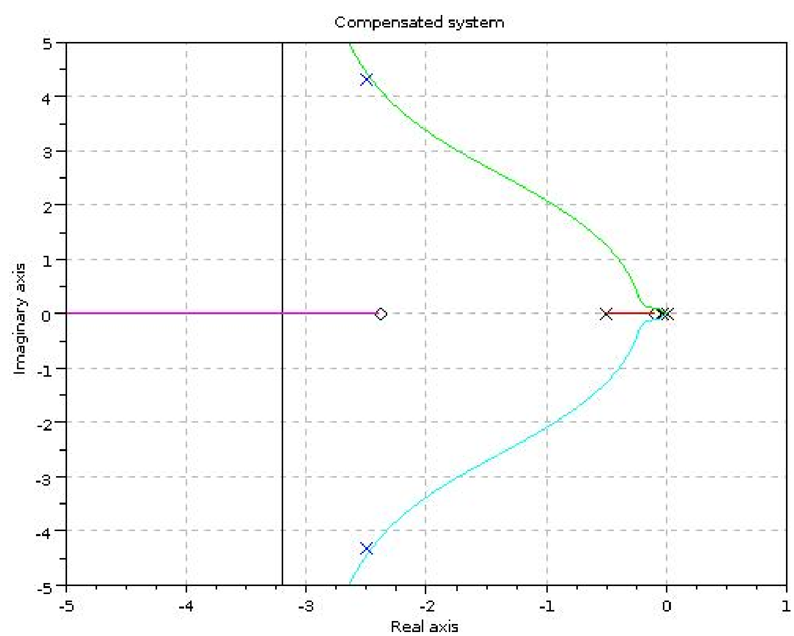

Figure 6.29: Design of lag lead compensator using root locus 2

```
clear; clc;
xdel(winsid());   //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

s = %s;
G = 4 / (s * (s + 0.5));

Gc = 10 * (s + 2.38) * (s + 0.1) / (s + 8.34) / (s +
    0.0285);
GGc = G*Gc;

H = syslin('c',G /. 1);
Hc = syslin('c',GGc /. 1);

t = 0:0.05:20;
u1 = ones(1,length(t)); //step response
plotresp(u1,t,H,'');
plotresp(u1,t,Hc,'Unit step response');
xstring(0.5,1.7,'uncompensated system');
xstring(1,0.95,'compensated system');

scf()
t = 0:0.05:10;
plotresp(t,t,H,'');
plotresp(t,t,Hc,'Unit ramp response');
xstring(1.4,0.9,'uncompensated system');
xstring(0,1.5,'compensated system');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

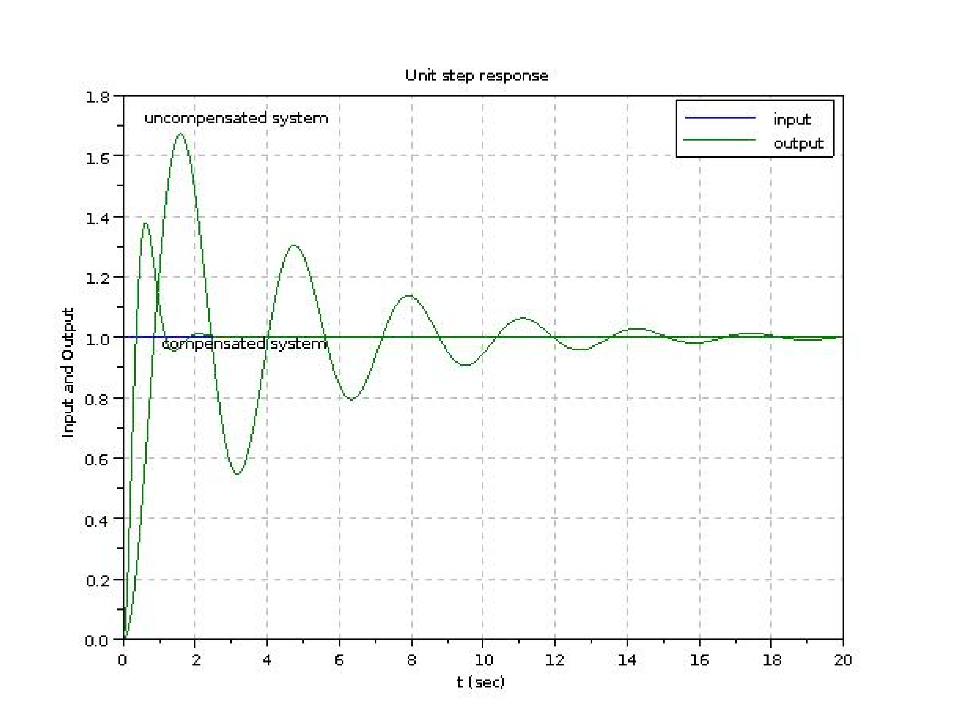

Figure 6.30: Evaluating Lag Lead compensated system

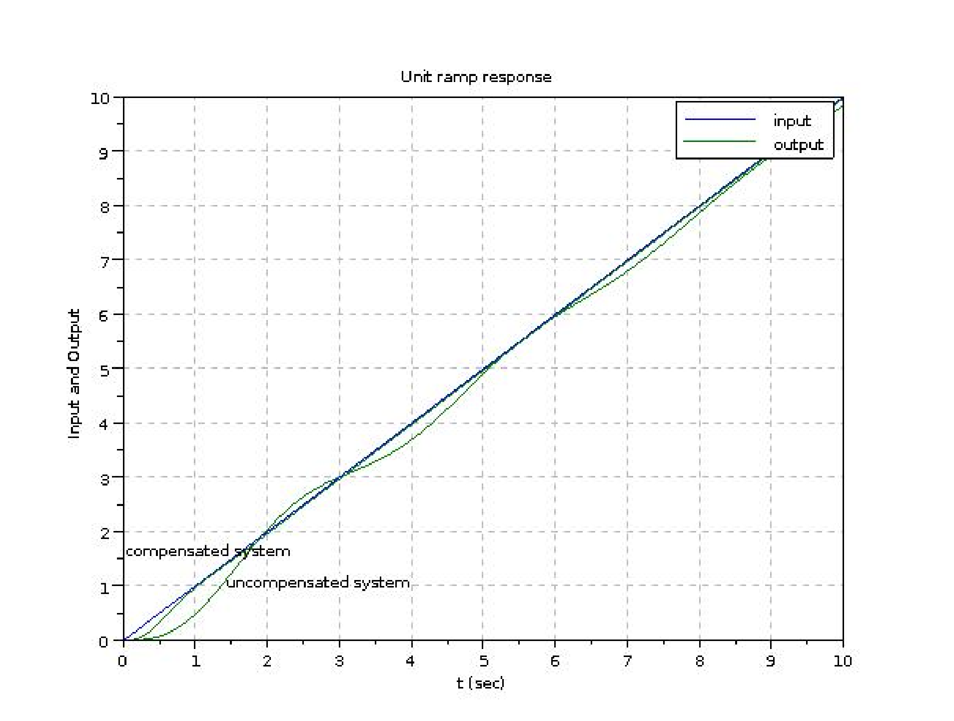

Figure 6.31: Evaluating Lag Lead compensated system

check Appendix AP 2 for dependency:

plotresp.sci

Scilab code Exa 6.10 Design of parallel compensation by root locus

```
// Example 6-10
// Design of parallel compensation by root locus

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

function [G,C] = getsystem(K)
  G = 20 / ( s*(s+1)*(s+4) + K*s ); //open loop system
  C = syslin('c',G /. 1); // closed loop system
endfunction

s = %s;

// Root locus of the denominator polynomial (modified)
H = syslin('c',s , s^3 + 5*s^2 + 4*s + 20);
evans(H);
a= gca();a.children(1).visible = 'off';
sgrid([0.4],[]); // draw zeta = 0.4 line
a.box = "on";
a.data_bounds = [-6 -6;1 6];
xgrid(color('gray'));

r = [ -2.1589 ; -1.049 ]; i =[4.9652; 2.4065];
p  = r + %i * i;
```

```
K = [1; 1] ./ abs(horner(H,p));
plot(r,i,'.');
xstring(r,i,['K =' + string(K(1)),'K =' + string(K
   (2))] );

k = K ./ 20;
disp([K k],'K : k = ');
[G1 C1] = getsystem(K(1));
[G2 C2] = getsystem(K(2));

disp(roots(C1.den),'closed loop poles of system with
    k = ' + string(k(1)));
disp(roots(C2.den),'closed loop poles of system with
    k = ' + string(k(2)));
disp(C1,'C1 =');disp(C2,'C2 =');

scf();
t = 0:0.05:10;
u = ones(1,length(t));
plotresp(u,t,C1,'');
plotresp(u,t,C2,'Step response of parallel
   compensated systems');
xstring(1.3,1.1,'k = ' + string(k(1)));
xstring(2,0.8,'k = ' + string(k(2)));
```

check Appendix AP 2 for dependency:

`plotresp.sci`

check Appendix AP 7 for dependency:

`rootl.sci`

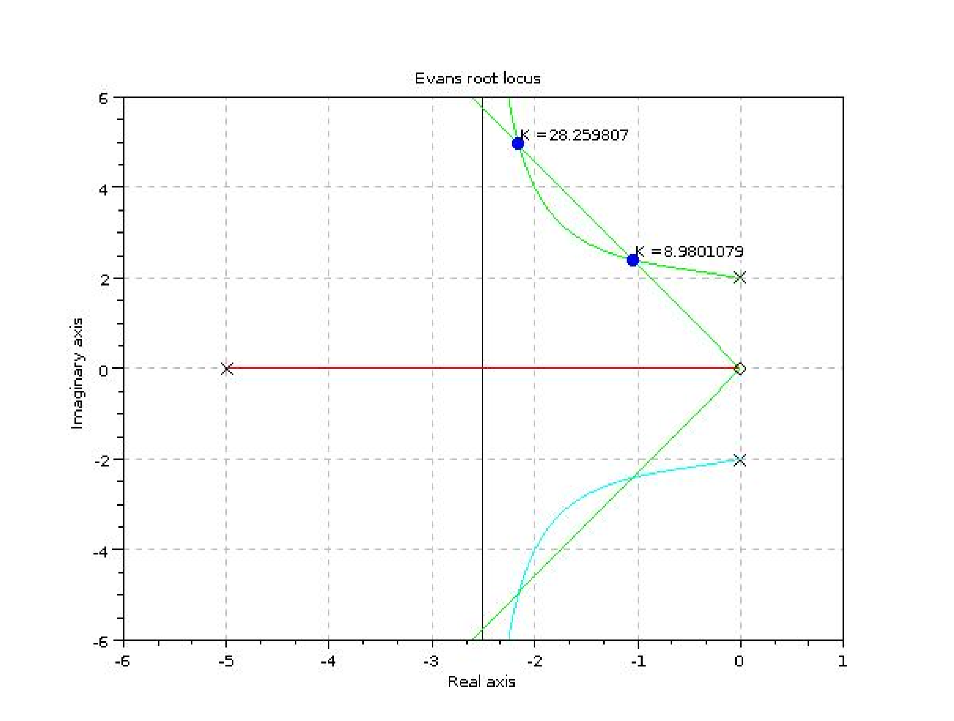

Figure 6.32: Design of parallel compensation by root locus

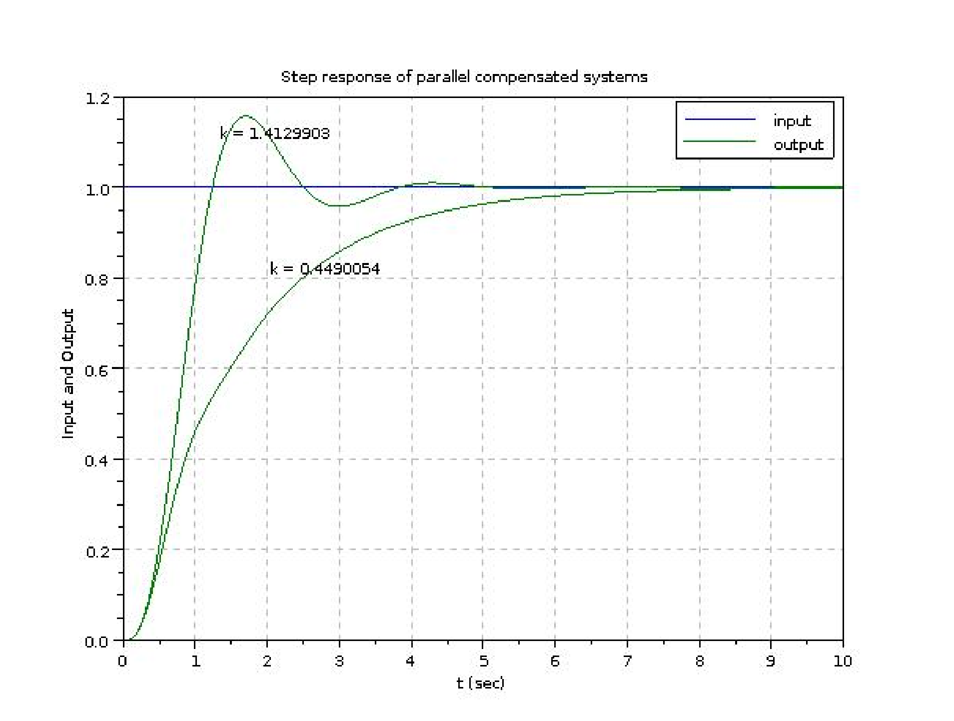

Figure 6.33: Design of parallel compensation by root locus

Scilab code Exa 6.15 Design of lag compensator

```
// Example A-6-15
// Design of lag compensator

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");
// exec("plotresp.sci");

s = %s;
G = syslin('c',10,s * (s + 4));


Kv = 80;            // desired velocity constant
R = [-2 -2];
I = [sqrt(6) -sqrt(6)];
dp = R(1) + %i*I(1)

disp(horner(s*G,0),'Kv (uncompensated system) = ');
_beta = 20;  // taking Kc =1 we get beta as 10
z = 0.1;     // choose z = 0.1
p = z / _beta;
Gc = (s + z)/(s + p);
disp(Gc ,'compensator = ');
H = G * Gc ;          // compensated system
H = syslin('c',numer(H),denom(H));
Gdp = horner(Gc,dp);
disp(abs(Gdp),'Magnitude contribution of controller =');
disp(phasemag(Gdp),'Angle contribution of controller =');

rootl(G,[-3 -4; 1 4],'');
```

```
rootl(H,[-3 -4; 1 4],'Uncompensated and Compensated system');
xgrid(color('gray'));
plot(R,I,'x');
xstring(R(1),I(1),'Original pole on uncompensated sys');

G1 = syslin('c',G /. 1);
C = syslin('c',H /. 1);        // final closed loop system
disp(C,'closed loop system =');
disp(roots(C.den),'closed loop poles = ');
disp(horner(s*H,0),'velocity error constant Kv =')

scf();
subplot(2,1,1);
t = 0:0.05:10;
u = ones(1,length(t));
plotresp(u,t,G1,'');
plotresp(u,t,C,'Unit step response');
xstring(1,0.9,'uncompensated system');
xstring(0.7,1.12,'compensated system');


t = 0:0.5:20;
subplot(2,1,2);
plotresp(t,t,G1,'');
plotresp(t,t,C,'Unit ramp response');
xstring(2,0.9,'uncompensated system');
xstring(0.1,4,'compensated system');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

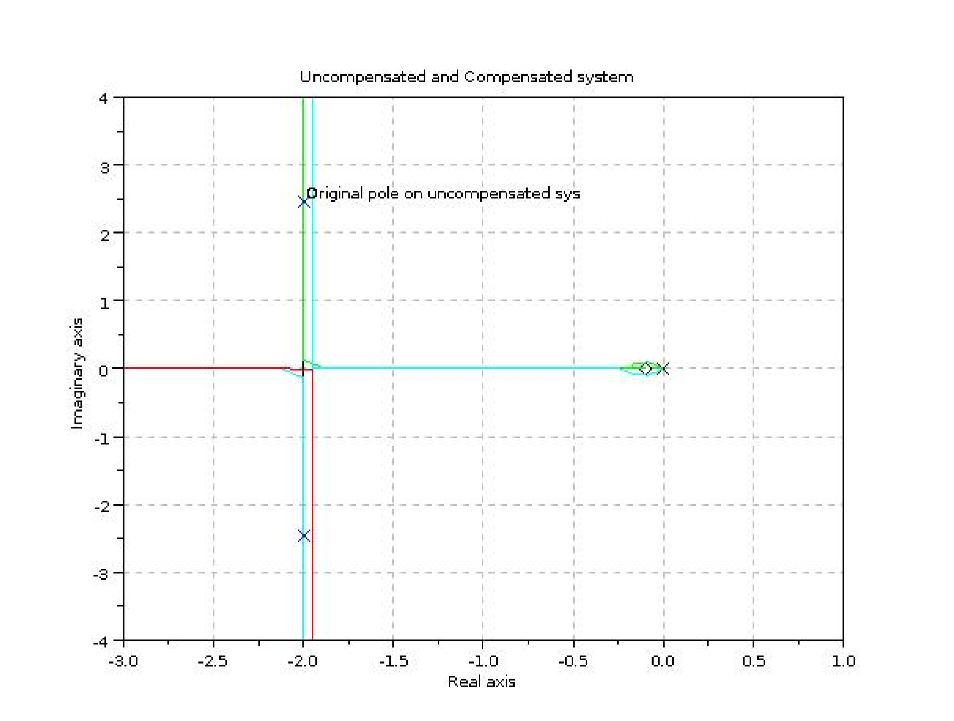

Figure 6.34: Design of lag compensator

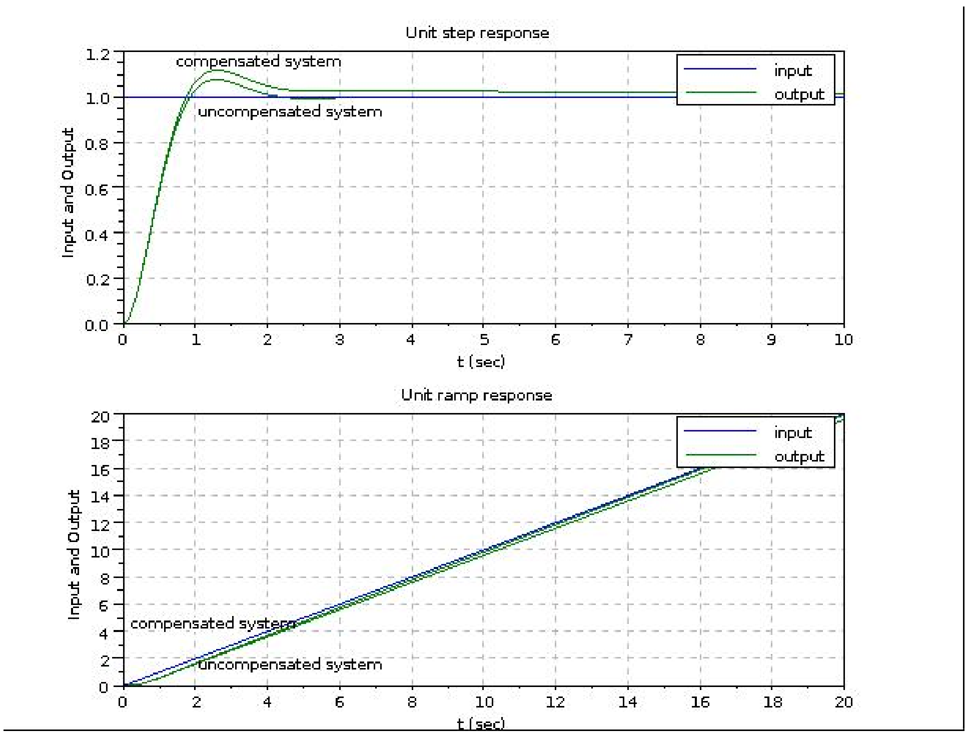

Figure 6.35: Design of lag compensator

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 6.16 Design of lag lead compensator

```
// Example A-6-16
// Design of lag lead compensator

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "/<your code directory>/";
// exec("rootl.sci");
// exec("plotresp.sci");

s = %s;
G = syslin('c',10,s * (s + 2) * (s + 8));

Kv = 80;            // desired velocity constant
R = [-2 -2];
I = [2*sqrt(3) -2*sqrt(3)];
dp = R(1) + %i*I(1)

disp(horner(s*G,0),'Kv (uncompensated system) = ');

// designing lead part
Kc = Kv /abs(horner(s*G,0))
angdef = 180 - phasemag(horner(G,dp))
z1 = 3.7      //z1 and p1 determinded graphically
p1 = 53.35
T1 = 1 / z1
_beta = T1 * p1; disp(_beta,'beta =');

```

```
Gc1 =Kc * (s + z1)/(s + p1); disp(Gc1,'Lead
    compensator Gc1 =');

// Lag compensator design
p2 = 0.01  //say
z2 = p2 * _beta
Gc2 = (s + z2)/(s + p2);
disp(Gc2,'Lag compensator Gc2 =');
disp(abs(horner(Gc2,dp)),'magnitude contribution of
    lag part =');
disp(phasemag(horner(Gc2,dp)),'angle contribution of
     lag part =');
// these are acceptable

Gc = Gc1 * Gc2
H = G * Gc ;          // compensated system
H = syslin('c',numer(H),denom(H));

subplot(1,2,1);
rootl(G,[-10 -10; 10 10],'Uncompensated system');
plot(R,I,'x');
xgrid(color('gray'));
subplot(1,2,2);
rootl(H,[-10 -10; 10 10],'Compensated system');
plot(R,I,'x');
xgrid(color('gray'));
xstring(R(1),I(1),'Desired closed loop poles');

G1 = syslin('c',G /. 1);
C = syslin('c',H /. 1);      // final closed loop
    system
disp(C,'closed loop system =');
disp(roots(C.den),'closed loop poles = ');
disp(horner(s*H,0),'velocity error constant Kv =')

scf();
subplot(2,1,1);
t = 0:0.05:10;
```

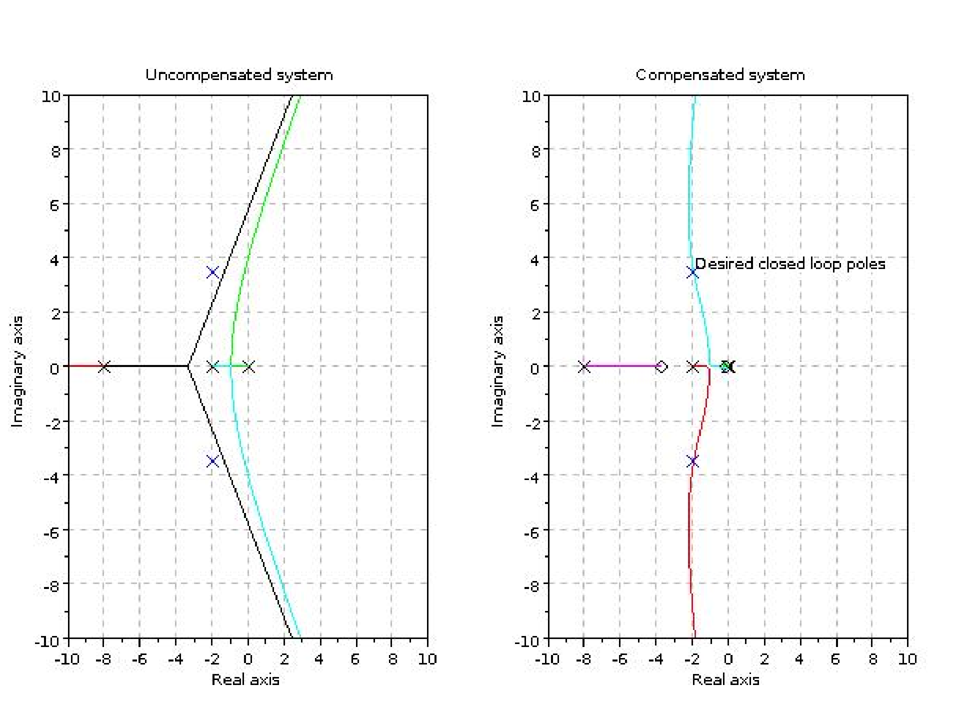

Figure 6.36: Design of lag lead compensator

```
u = ones(1,length(t));
plotresp(u,t,G1,'');
plotresp(u,t,C,'Unit step response');
xstring(1,0.5,'uncompensated system');
xstring(0.7,1.12,'compensated system');

subplot(2,1,2);
plotresp(t,t,G1,'');
plotresp(t,t,C,'Unit ramp response');
xstring(2,0.9,'uncompensated system');
xstring(0.5,2,'compensated system');
```

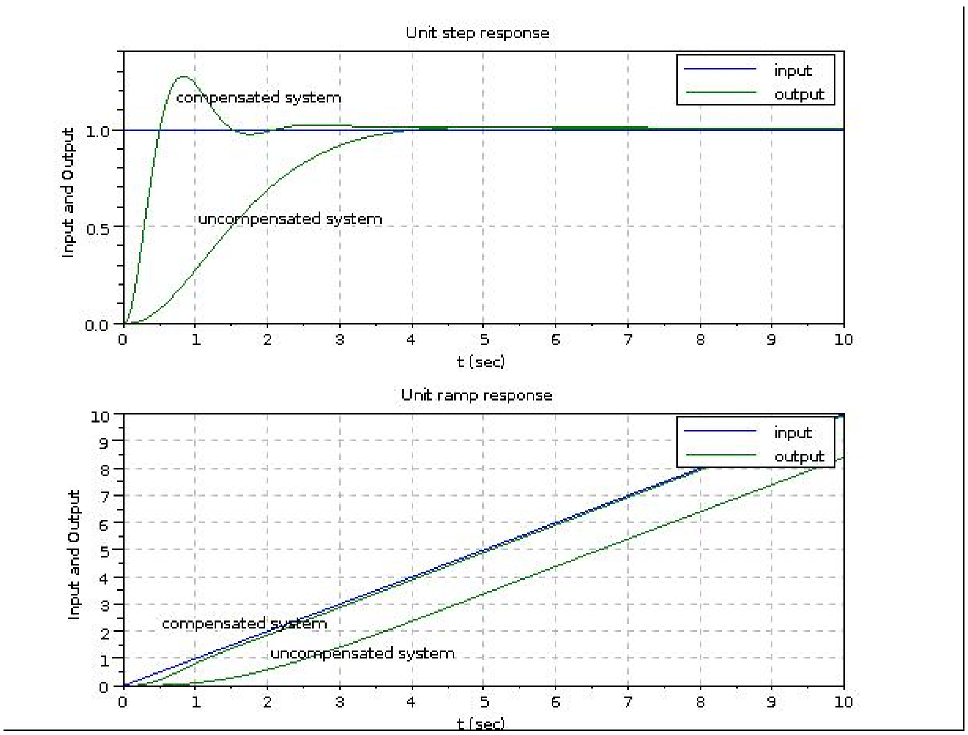

Figure 6.37: Design of lag lead compensator

Chapter 7

Control Systems Analysis and Design by Frequency Response Method

Scilab code Exa 7.a.1 `Bode plot`

```
// Example A-7-1
// Bode plot

clear; clc;
xdel(winsid()); //close all windows

s = %s /2 /%pi; // frequencies in rad/s
G = syslin('c', 10*(s + 1), (s + 2)*(s + 5));
bode(G,0.1,100);
xtitle('Bode plot of G(s) = 10*(s + 1)/[(s + 2)*(s + 5)]','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
```

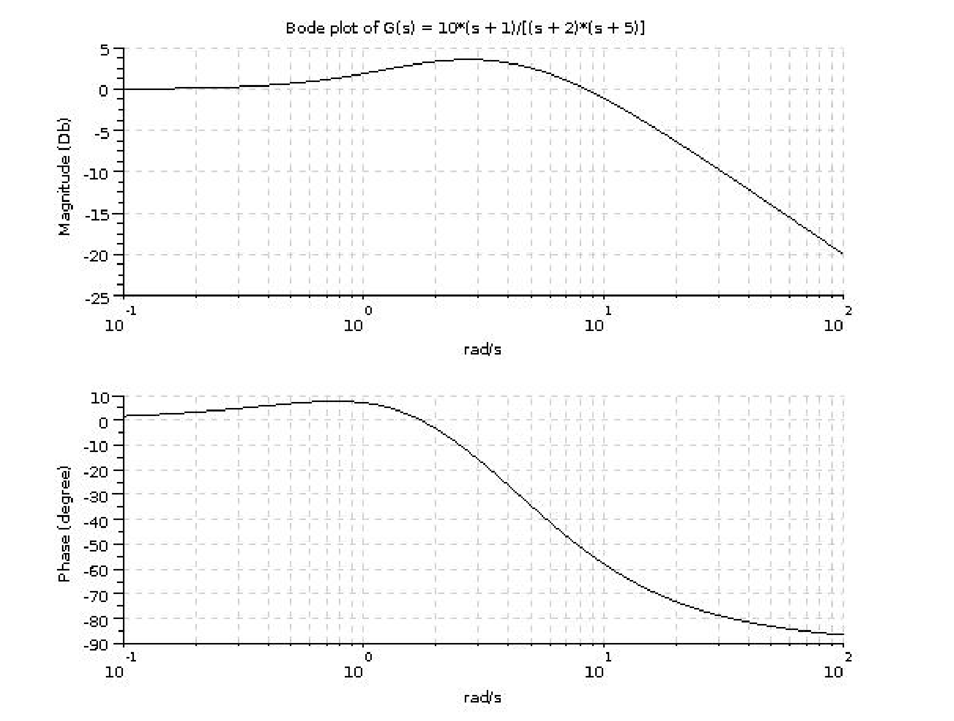

Figure 7.1: Bode plot

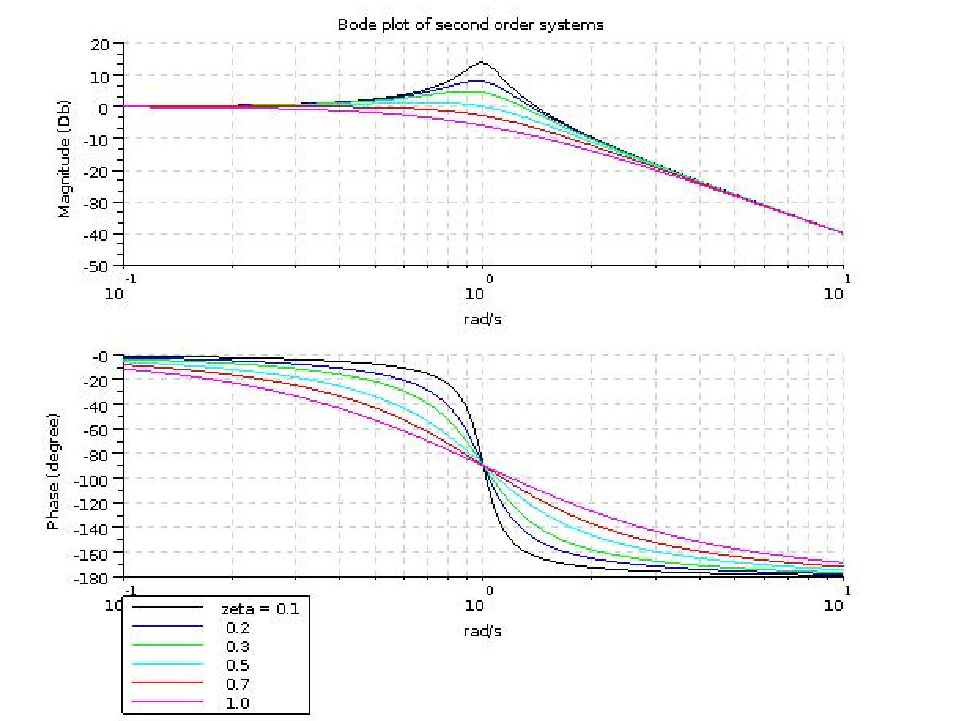

Figure 7.2: Bode plot for 2nd order systems with varying zeta

Scilab code Exa 7.i.1 Bode plot for 2nd order systems with varying zeta

```
// Illustration 7-1
// Bode plot of second order systems with varying
   damping (zeta)

// With refernce to section 7.2  (Figure 7.9)

clear; clc;
xdel(winsid());  //close all windows

s = %s;
// Taking wn = 1 in all cases
zeta = [0.1 0.2 0.3 0.5 0.7 1.0];


N = ones(6,1);
D = zeros(6,1);
for i = 1:6
  D(i) = s^2 + 2*zeta(i)*s + 1;
end
H = syslin('c',N,D);

omega = logspace(-1,1,100);
f = omega / 2 / %pi;
repf = repfreq(H,f);    // Frequency response

bode(omega,repf, ['zeta = 0.1',' 0.2',' 0.3',' 0.5',
   ' 0.7',' 1.0']);
xtitle('Bode plot of second order systems','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
```

Scilab code Exa 7.a.3 `Bode plot for system in state space`

```
// Example A-7-3
// Bode plot for system in state space

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("transferf.sci");

A = [0 1; -25 -4];
B = [1 1; 0 1];
C = [1 0; 0 1];
D = zeros(2,2);
G = transferf(A,B,C,D);disp(G,"transfer function = "
    );

subplot(2,2,1);
bode(G(1,1));
subplot(2,2,2);
bode(G(1,2));
subplot(2,2,3);
bode(G(2,1));
subplot(2,2,4);
bode(G(2,2));
```

check Appendix AP 4 for dependency:

`transferf.sci`

Scilab code Exa 7.a.4 `Bode plot for different gain K`

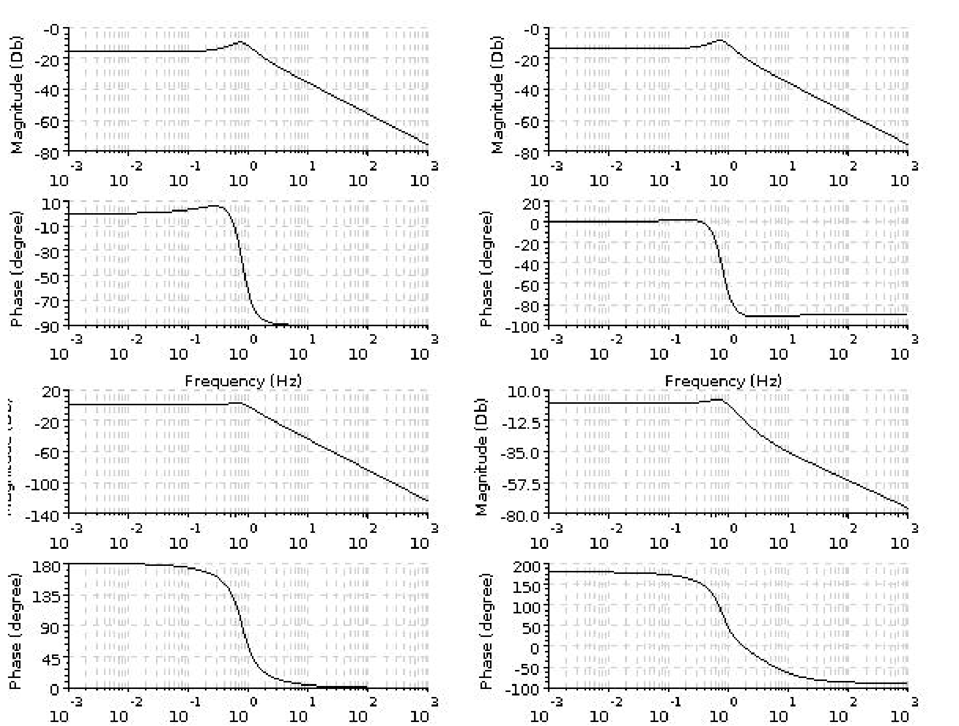

Figure 7.3: Bode plot for system in state space

```
// Example A-7-4
// Bode plot for different gain K

clear; clc;
xdel(winsid()); //close all windows

s = %s /2/%pi ;
P = s*(s+1)*(s+5);
num = [1,10,20];
den = [P+1 , P+10, P+20];
Gtf = num ./ den;
G = syslin('c',Gtf);

bode([G(1,1); G(1,2); G(1,3)],0.1,100,['K = 1';'K =
    10';'K = 20'] );
xtitle('','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
```

Scilab code Exa 7.a.8 Stability check

```
// Example A-7-8
// Stability check

clear; clc;
xdel(winsid()); //close all windows

s = %s;
K = 2;
P = s*(s+1)*(2*s+1) + K;
disp(routh_t(P))
// unstable since two roots are in RHP
```

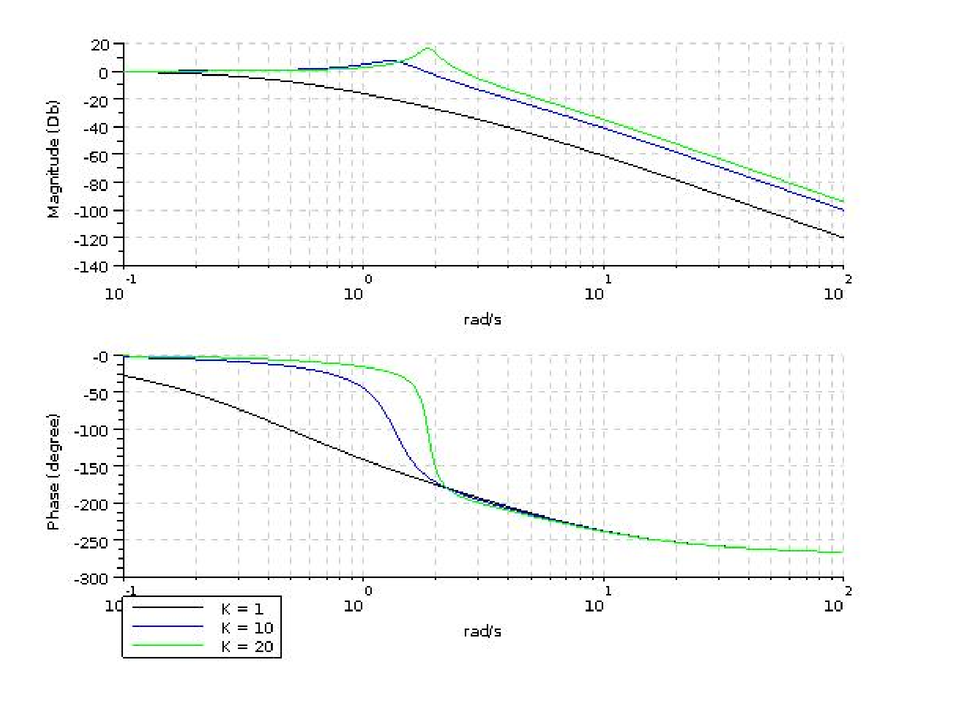

Figure 7.4: Bode plot for different gain K

Scilab code Exa 7.a.10 Nyquist Plot with transport lag

```
// Example A-7-10
// Nyquist Plot with transport lag

clear; clc;
xdel(winsid()); //close all windows


omega = logspace(-2,2,100);
f = omega ./ (2*%pi);
repf = 2.65 * exp(%i*omega*-0.8) ./ (ones(1,length(
    omega)) + %i*omega);

nyquist(f,repf);
plot(-1,0,'.');
xstring(-0.9,0,'passes -1',0,1);
```

Scilab code Exa 7.a.11 Nyquist Plot

```
// Example A-7-11
// Nyquist Plot

clear; clc;
xdel(winsid()); //close all windows

s = %s /2 /%pi;
num = 20 * ( s^2 + s + 0.5);
den = s * (s + 1) * (s + 10);
G = syslin('c',num,den);
```

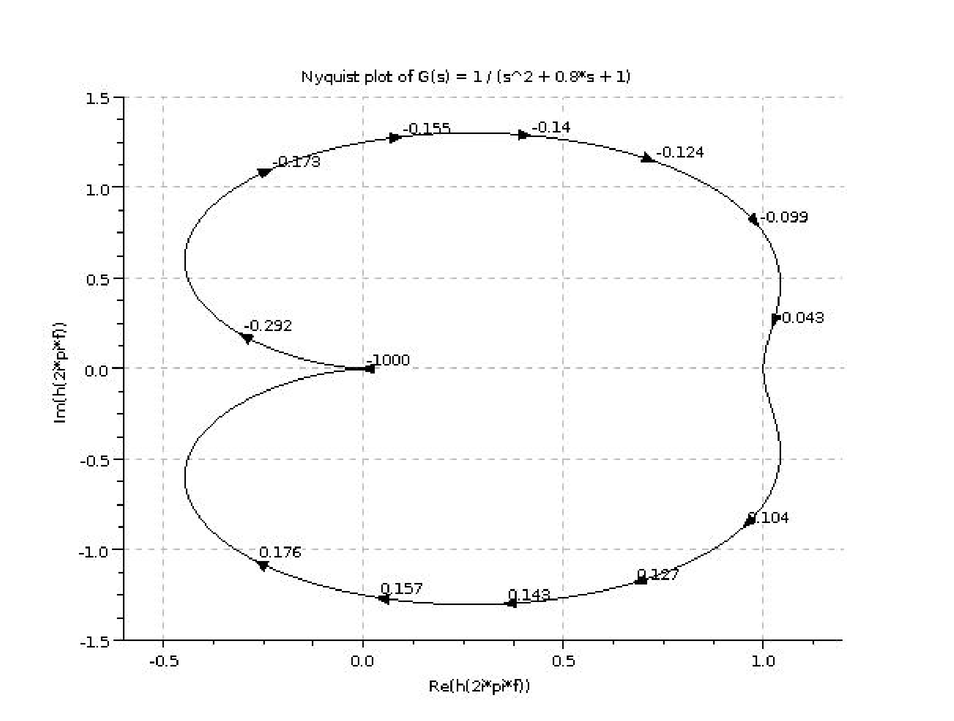

Figure 7.5: Nyquist Plot with transport lag

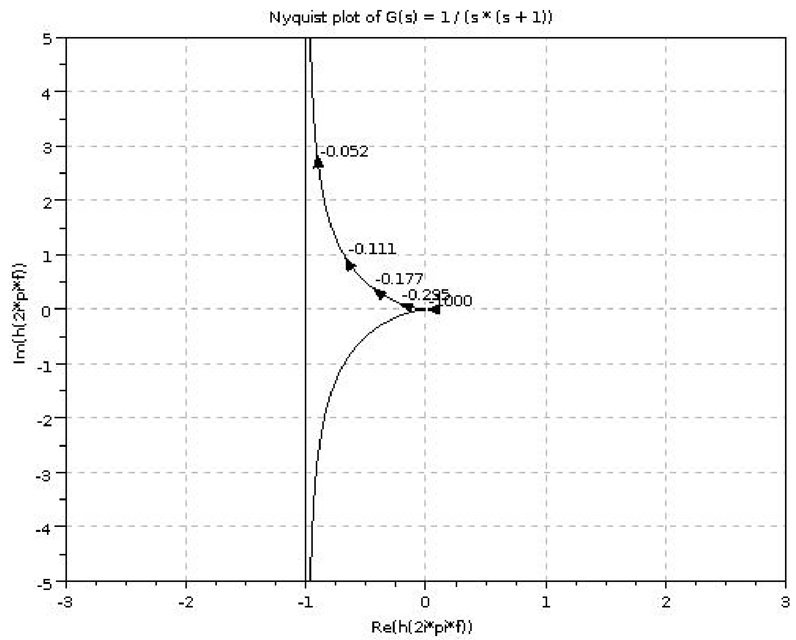

Figure 7.6: Nyquist Plot

```

a = gca();
a.clip_state = 'on';
nyquist(G,-1000,1000);
xgrid(color('gray'));
a.data_bounds = [-2 -3 ; 3 3];
a.box = 'on';
```

Scilab code Exa 7.a.12 Nyquist plot for positive omega

```
// Example A-7-12
```

```
// Nyquist plot for positive omega

clear; clc;
xdel(winsid()); //close all windows

s = %s /2 /%pi;
num = 20 * ( s^2 + s + 0.5);
den = s * (s + 1) * (s + 10);
G = syslin('c',num,den);

a = gca();
a.clip_state = 'on';
nyquist(G,0.01,1000);
xgrid(color('gray'));
a.data_bounds = [-3 -5 ; 3 1];
a.box = 'on';
```

Scilab code Exa 7.a.13 Nyquist plot with points at selected frequencies

```
// Example A-7-13
// Nyquist plot with points plotted at selected
   frequencies

clear; clc;
xdel(winsid()); //close all windows

s = %s /2 /%pi;
num = 20 * ( s^2 + s + 0.5);
den = s * (s + 1) * (s + 10);
G = syslin('c',num,den);

a = gca();
a.clip_state = 'on';
```

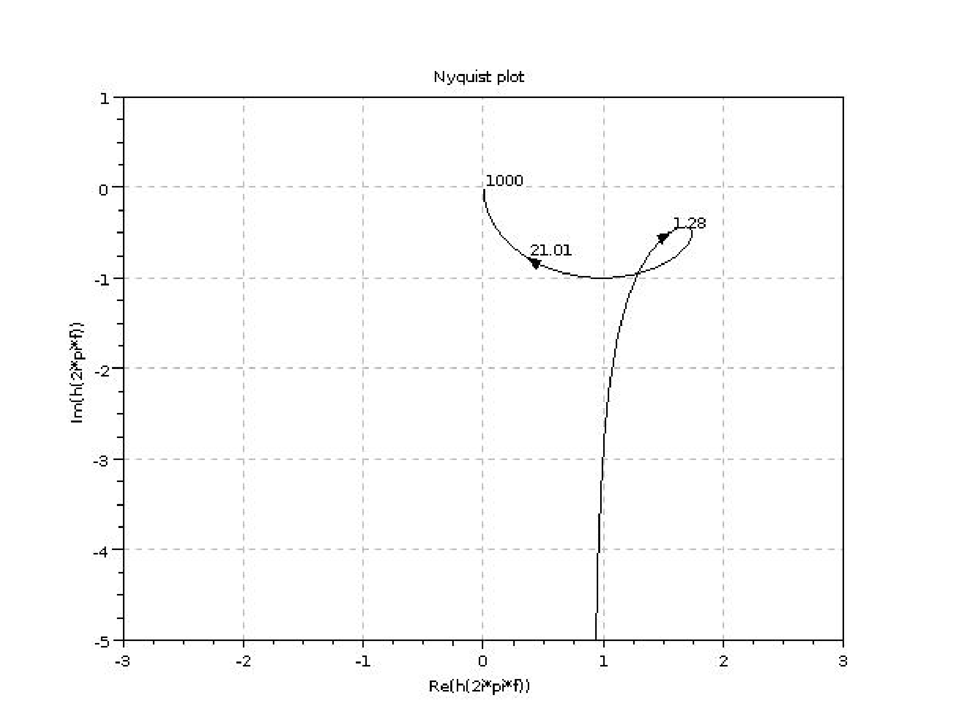

Figure 7.7: Nyquist plot for positive omega

```
nyquist(G,0.01,1000);
xtitle('Nyquist Diagram');
a.data_bounds = [-2 -5 ; 3 0];
a.box = 'on';

omega = [0.2 0.3 0.5 1 2 6 10 20];
z = repfreq(G,omega);
plot(real(z), imag(z),'.k');

x = [1      1.1  1.2   1.3   1.8  1.5  0.8  0.25];
y = [-4.7 -3.3 -1.7  -0.51 -0.4  -1   -1.3  -1];
text = ['w = 0.2' '0.3' '0.5' '1.0' '2.0' '6.0' '10'
    '20'];
xstring(x,y,text,0,1);

[phi db] =phasemag(z);
mag = abs(z);
disp([omega' mag' phi'] , '[w mag phi] = ');
```

Scilab code Exa 7.a.14 Nyquist plot for positive and negative feedback

```
// Example A-7-14
// Nyquist plot for positive and negative feedback

clear; clc;
xdel(winsid()); //close all windows

s = %s;
num = s^2 + 4*s + 6;
den = s^2 + 5*s + 4;
G = syslin('c',num,den);
H = syslin('c',-1 * num,den);

```

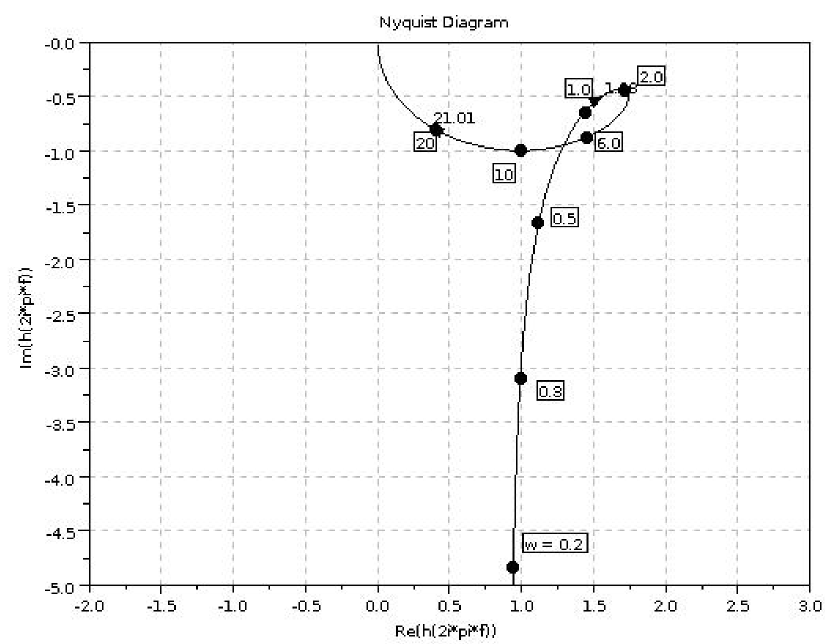

Figure 7.8: Nyquist plot with points at selected frequencies

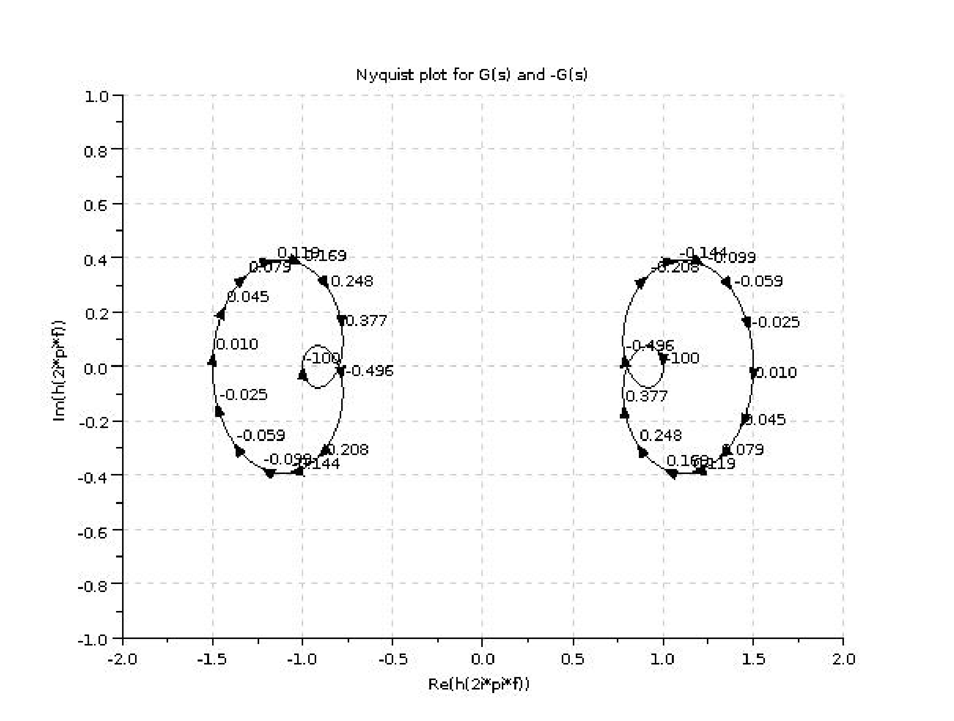

Figure 7.9: Nyquist plot for positive and negative feedback

```
nyquist(G,-100,100);
nyquist(H,-100,100);
xtitle('Nyquist plot for G(s) and -G(s)');
a = gca(); a.data_bounds = [-2 -1; 2 1];
```

Scilab code Exa 7.a.18 Verifying experimentally derived Transfer function

```
// Example A-7-18
// Verifying experimentally derived Transfer
   function

```

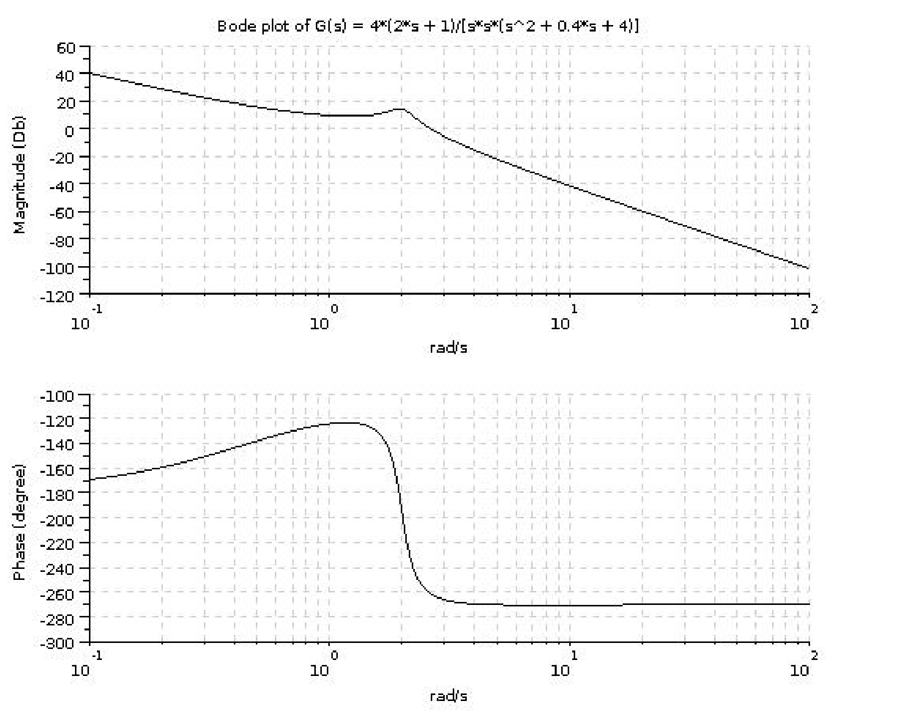

Figure 7.10: Verifying experimentally derived Transfer function

```
clear; clc;
xdel(winsid()); //close all windows

s = %s /2 /%pi; // frequencies in rad/s
G = syslin('c', 4*(2*s + 1), s*s*(s^2 + 0.4*s + 4) )
    ;
bode(G,0.1,100);
xtitle('Bode plot of G(s) = 4*(2*s + 1)/[s*s*(s^2 +
    0.4*s + 4)]','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
```

Scilab code Exa 7.a.23 Nichols plot

```
// Example A-7-23
// Nichols plot

clear; clc;
xdel(winsid()); //close all windows

s = %s;
G = syslin('c',9 , s*(s+0.5)*(s^2 + 0.6*s + 10) );
black(G);
chart([8 -4],[],list(1,0));
xgrid(color('gray'));
```

---

check Appendix AP 2 for dependency:

plotresp.sci

Scilab code Exa 7.1 Steady state sinusoidal output

```
// Example 7-1
// Steady state sinusoidal output

clear; clc;
xdel(winsid()); //close all windows

// please set the path
// cd "/<your code directory>/"
// exec("plotresp.sci")

s = %s;
w = 1;
K = 5;
T = 0.1;
```

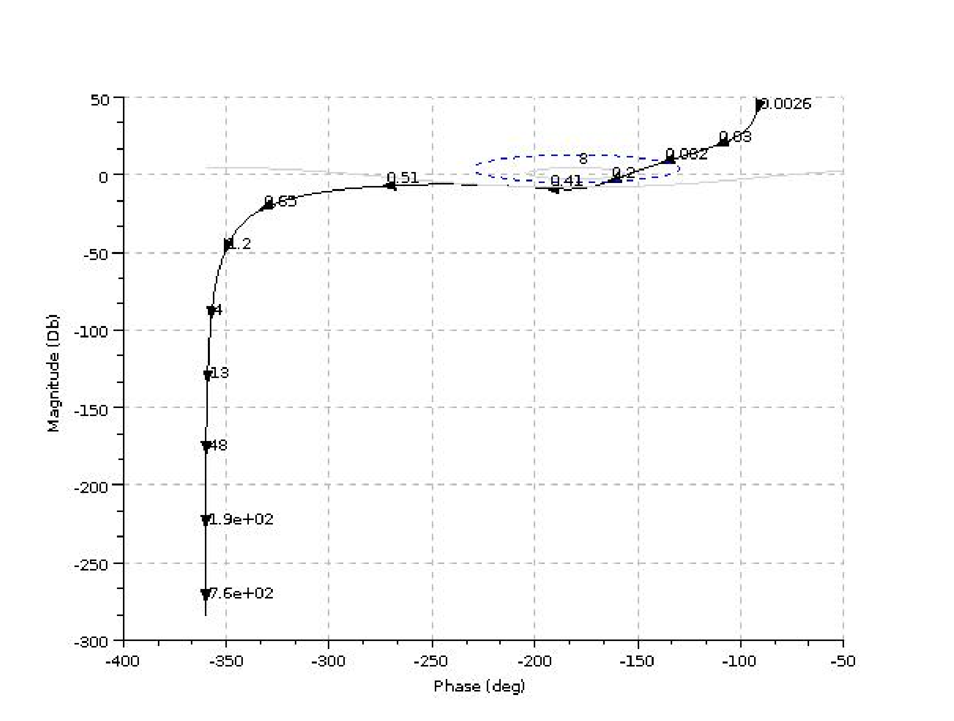

Figure 7.11: Nichols plot

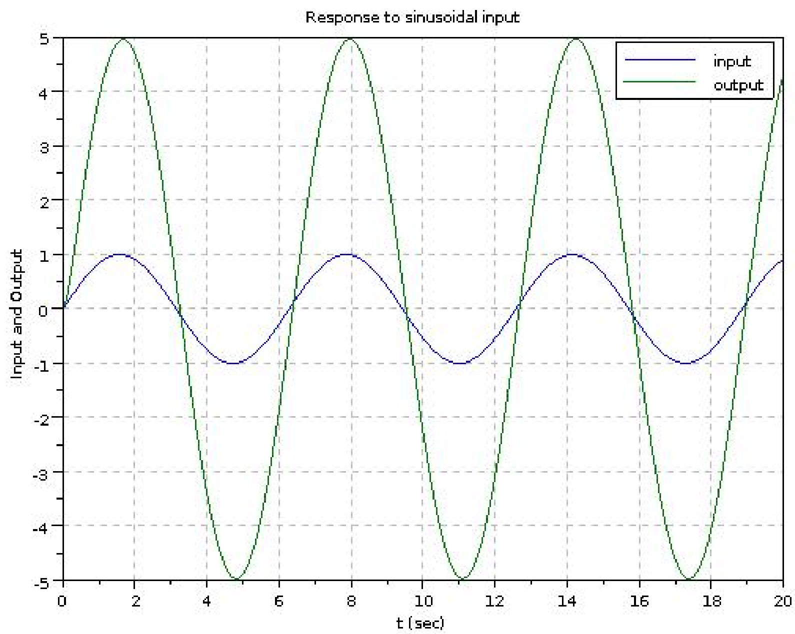

Figure 7.12: Steady state sinusoidal output

```

G = syslin('c',K,T*s + 1);
t = 0:0.1:20;
u = sin(w*t);
plotresp(u,t,G,'Response to sinusoidal input');
// as T*w is small amplitude of output is ~ K (5)
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 7.2 `Steady state sinusoidal output lag and lead`

```
// Example 7-2
// Steady state sinusoidal output lag and lead

clear; clc;
xdel(winsid()); //close all windows

// please set the path
// cd "/<your code directory>/"
// exec("plotresp.sci")

s = %s;
T1 = 1;
T2 = 5;
a = s + 1/T1;
b = s + 1/T2;
w = 1;

G1 = syslin('c',a,b);
G2 = syslin('c',b,a);
t = 0:0.1:50;
u = sin(w*t);
plotresp(u,t,G1,'Response to sinusoidal input');
plotresp(u,t,G2,'Response to sinusoidal input');
xstring(17,1.4,'Lead network T1 > T2 : lead network'
    );
xstring(17,-0.8,'Lag network T1 > T2 : lead network'
    );
```

Scilab code Exa 7.3 Bode Plot in Hz

```
// Example 7-3
// Bode Plot in Hz

```

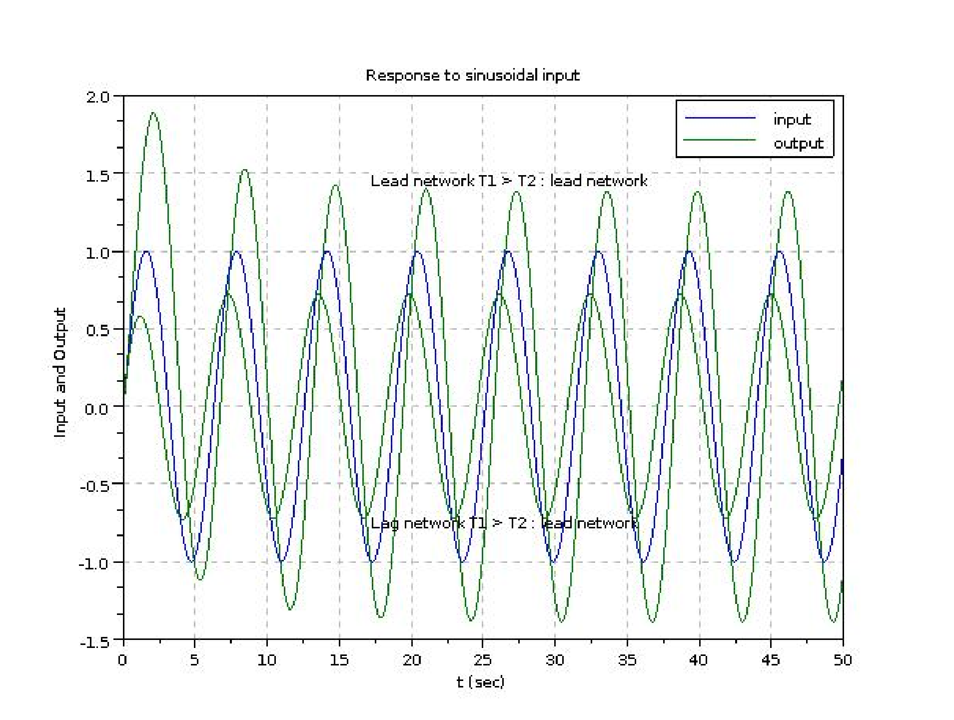

Figure 7.13: Steady state sinusoidal output lag and lead

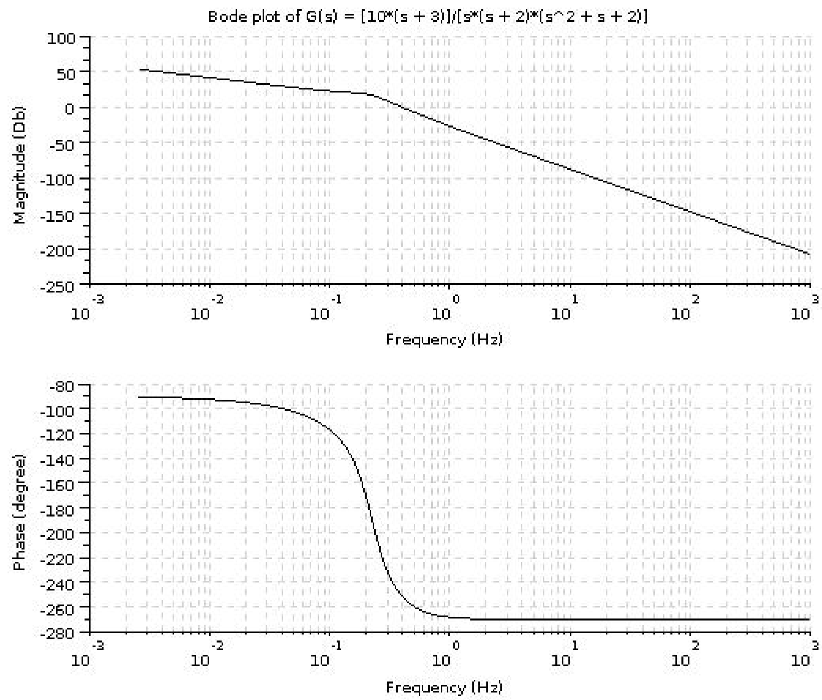

Figure 7.14: Bode Plot in Hz

```
clear; clc;
xdel(winsid()); //close all windows

s = %s;
num = 10*(s + 3);
den = s * (s + 2) * (s^2 + s + 2);
G = syslin('c',num,den);

bode(G);
xtitle('Bode plot of G(s) = [10*(s + 3)]/[s*(s + 2)
    *(s^2 + s + 2)]');
```

Scilab code Exa 7.4 Bode Plot with transport lag

```
// Example 7-4
// Bode Plot with transport lag

clear; clc;
xdel(winsid()); //close all windows

omega = logspace(-1,1,100);
repf = exp(%i*omega*-0.5) ./ (ones(1,length(omega))
   + %i*omega);

bode(omega,repf);
xtitle('Bode plot of G(s) = exp(-0.5jw) / [1 + jw]',
   'rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
```

Scilab code Exa 7.5 Bode Plot in rad per s

```
// Example 7-5
// Bode Plot in rad/s

clear; clc;
xdel(winsid()); //close all windows

s = %s;
num = 25;
den = s^2 + 4*s + 25;
G = syslin('c',num,den);

bode(G);
xtitle('Bode plot of G(s) = 25 / s^2 + 4*s + 25');

```

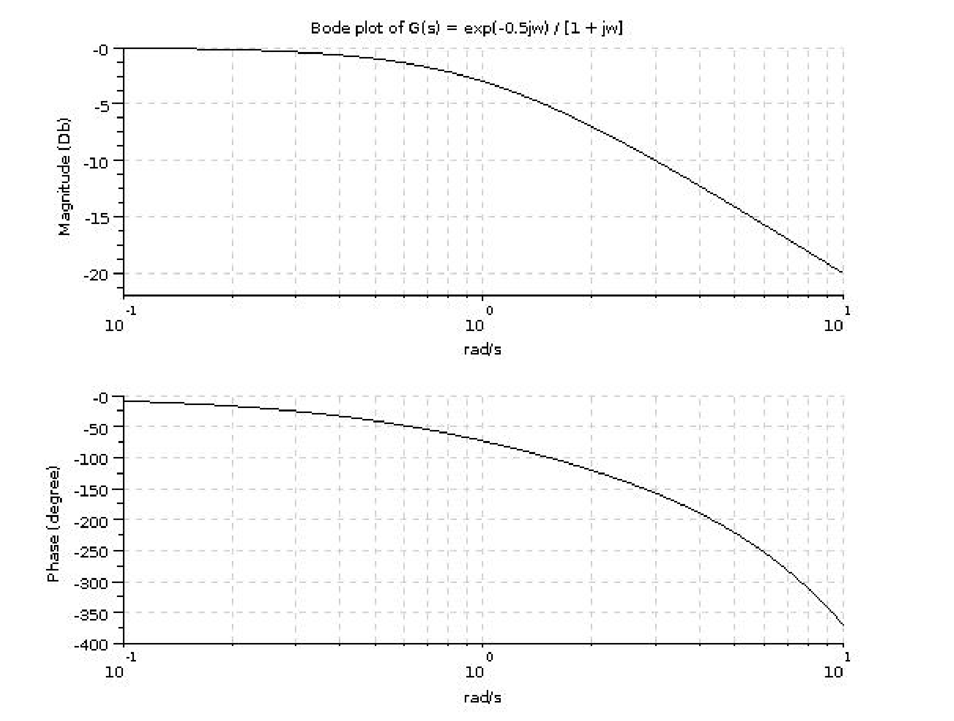

Figure 7.15: Bode Plot with transport lag

```
// Note, bode plots in Sci-Lab use the frequency in
   Hz and not in
// rad/s . If we wish to get the plot with rad/s we
   can...

omega = logspace(-2,2,50);
f = omega / 2 / %pi;
repf = repfreq(G,f);     // calculate the frequency
   response
                         // repf is a vector of
                            complex numbers
scf();
bode(omega,repf);
xtitle('Bode plot of G(s) = 25 / s^2 + 4*s + 25','
   rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
```

Scilab code Exa 7.6 Bode plot in rad per s

```
// Example 7-6
// Bode Plot in rad/s
// Plots made with angular freuqency - rad/s on the
   x-axis

clear; clc;
xdel(winsid()); //close all windows

s = %s / 2 / %pi; //correction to get frequency
   axis in rad/s
num = 9 * (s^2 + 0.2*s + 1);
den = s * (s^2 + 1.2*s + 9);
```

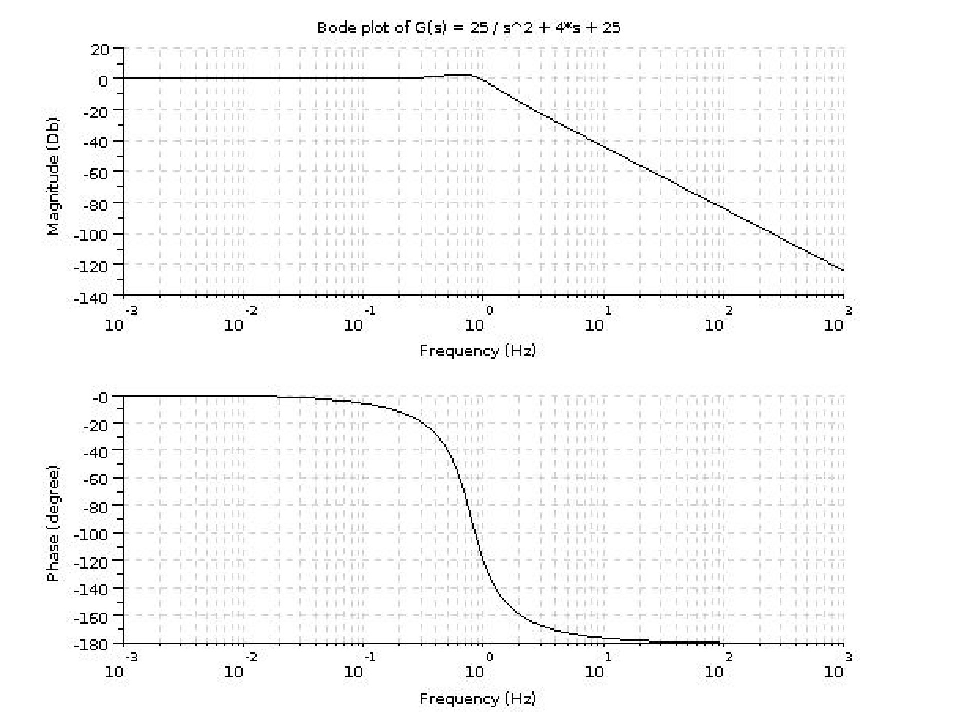

Figure 7.16: Bode Plot in rad per s

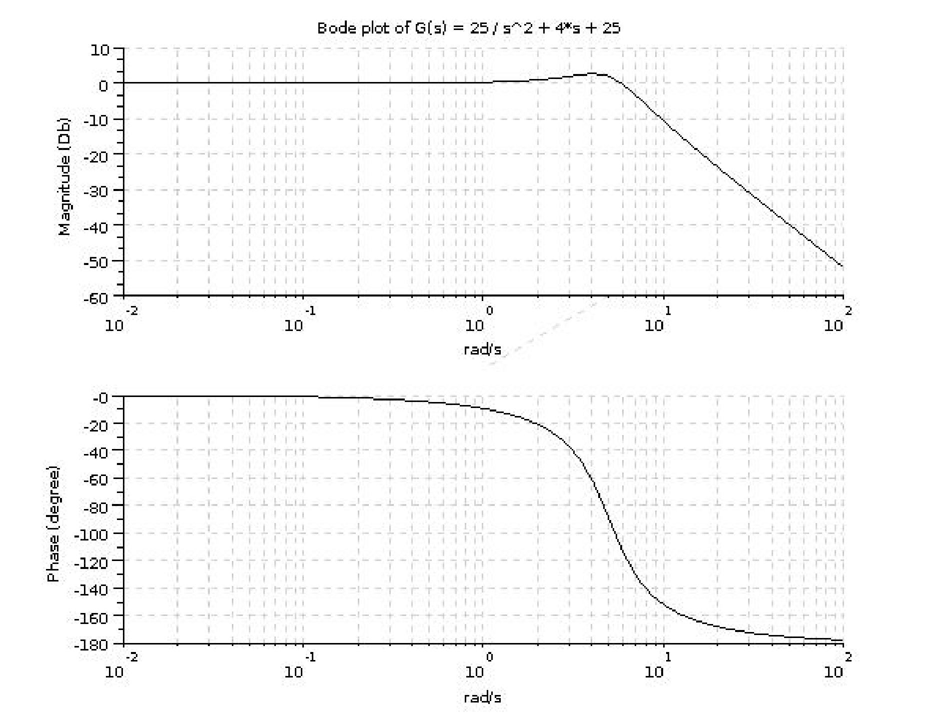

Figure 7.17: Bode Plot in rad per s

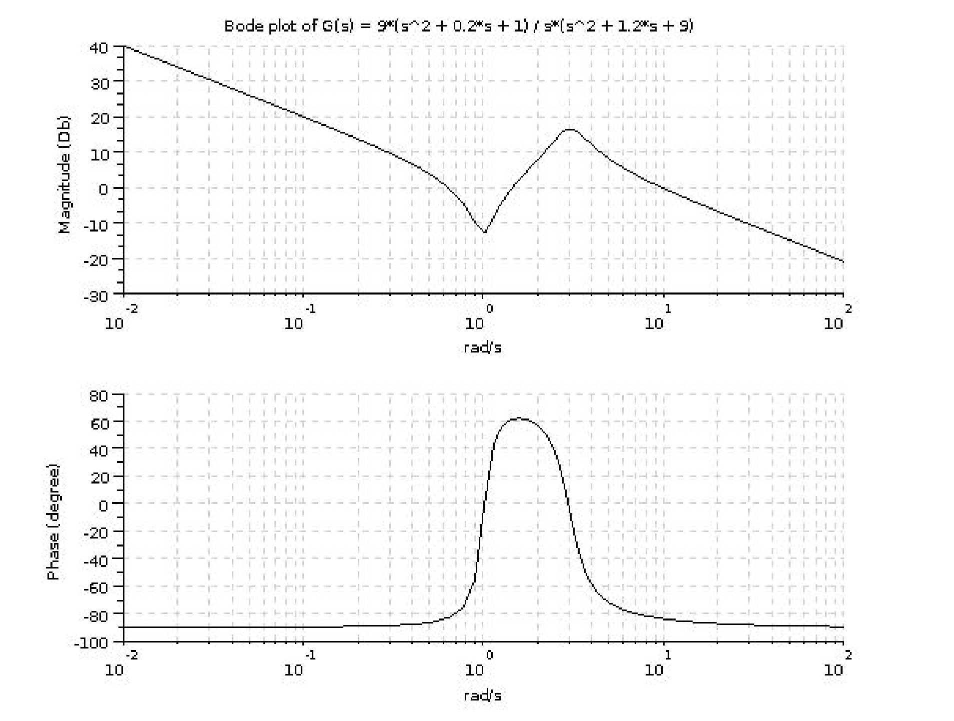

Figure 7.18: Bode plot in rad per s

```
G = syslin('c',num,den);

bode(G,0.01,100);
xtitle('Bode plot of G(s) = 9*(s^2 + 0.2*s + 1) / s
   *(s^2 + 1.2*s + 9)','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
```

Scilab code Exa 7.7 `Bode Plot for a system in State Space`

```
// Example 7-7
// Bode Plot for a system in State Space
```

```
representation

clear; clc;
xdel(winsid()); //close all windows

A = [0 1; -25 -4];
B = [0 ; 25];
C = [1 0];
D = [0];
G = syslin('c',A,B,C,D);

omega = logspace(-1,2,100);
f = omega / 2 / %pi;
repf = repfreq(G,f);    // Frequency response

bode(omega,repf);
xtitle('Bode Diagram','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
```

check Appendix AP 9 for dependency:

`spolarplot.sci`

Scilab code Exa 7.8 `Polar Plot of a linear system`

```
// Example 7-8
// Polar Plot of a linear system

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("spolarplot.sci");
```

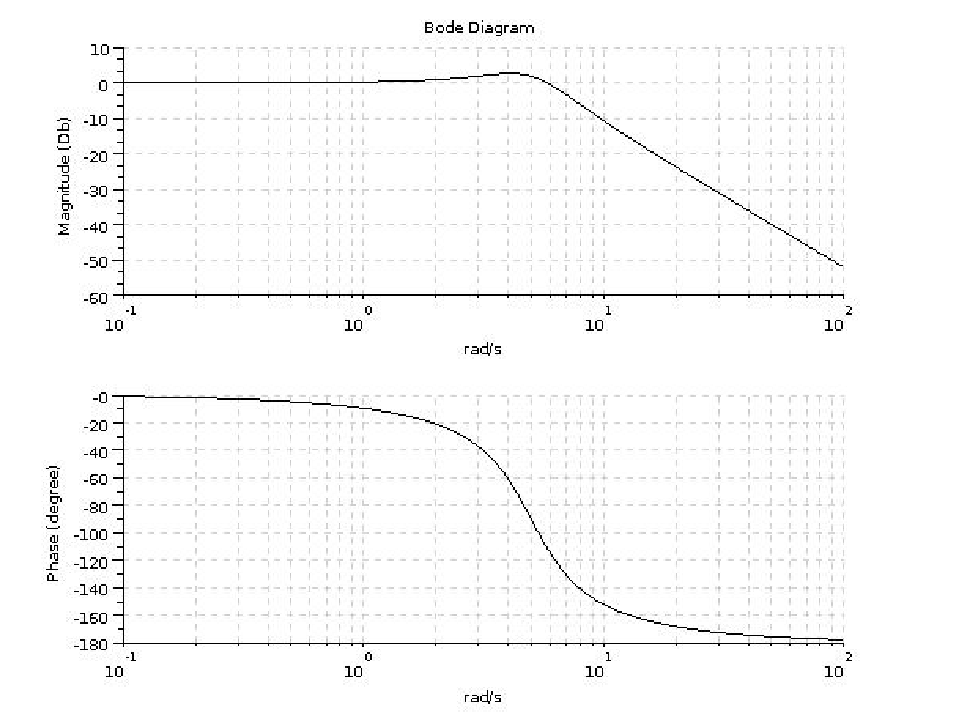

Figure 7.19: Bode Plot for a system in State Space

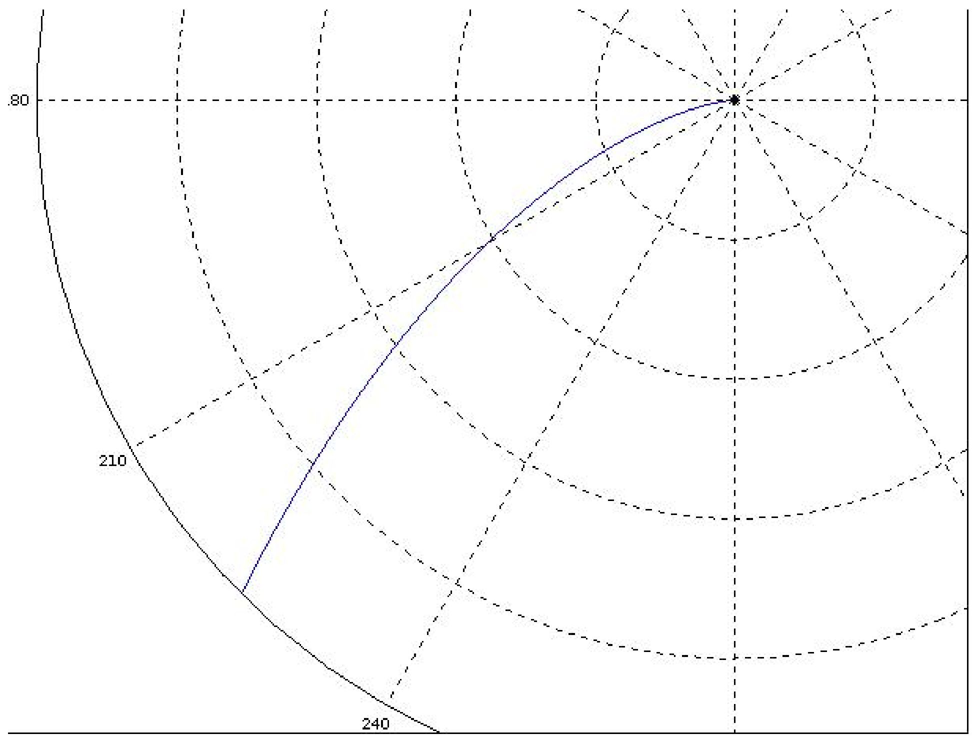

Figure 7.20: Polar Plot of a linear system

```

T = 10; s = %s;
omega = logspace(-1,3,1000);
G = syslin('c',1,s*(T*s + 1));
spolarplot();
```

Scilab code Exa 7.9 `Polar Plot with transport lag`

```
// Example 7-9
// Polar Plot with transport lag

```

```
clear; clc;
xdel(winsid());  //close all windows

T = 10;
L = 100;
omega = logspace(-1,2,1000);
s = %i * omega;
den = s .* (T*s + 1);
num = exp(-1*s*L);
repf = num ./ den;
rad = abs(repf);
theta = atan(imag(repf),real(repf));

polarplot(theta,rad,style = 2);
```

Scilab code Exa 7.10 Nyquist Plot

```
// Example 7-10
// Nyquist Plot

clear; clc;
xdel(winsid());  //close all windows

s = %s;
num = 1;
den = s^2 + 0.8*s + 1;
G = syslin('c',num,den);

nyquist(G,-1000,1000);
xgrid(color('gray'));
xtitle('Nyquist plot of G(s) = 1 / (s^2 + 0.8*s + 1)
    ');

```

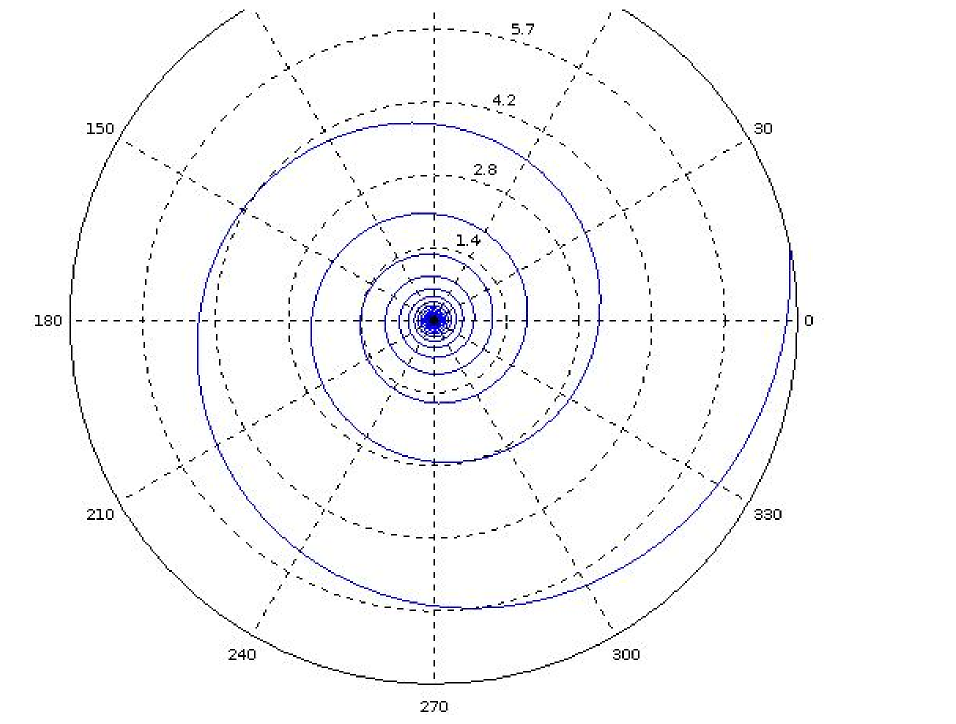

Figure 7.21: Polar Plot with transport lag

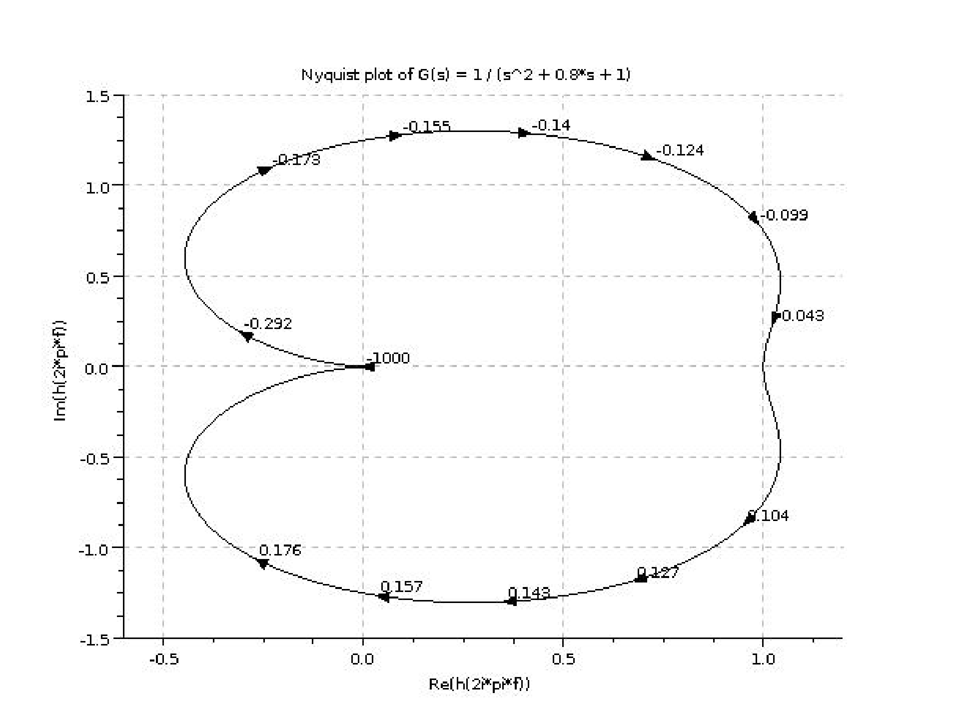

Figure 7.22: Nyquist Plot

```
// Note: nyquist function plots frequencies −1000
   and 1000 in Hz and not in rad/s
```

Scilab code Exa 7.11 `Nyquist Plot`

```
// Example 7−11
// Nyquist Plot

clear; clc;
xdel(winsid()); //close all windows

```

```
s = %s;
num = 1;
den = s * (s + 1);
G = syslin('c',num,den);

scf();
a = gca();
a.clip_state = 'on';    //clip the extra nyquist plot
nyquist(G,-1000,1000);
xgrid(color('gray'));
xtitle('Nyquist plot of G(s) = 1 / (s * (s + 1))');
a.data_bounds  = [-3 -5 ; 3 5];
a.box = 'on';
```

Scilab code Exa 7.12 Nyquist Plots of system in state space

```
// Example 7-11
// Nyquist Plot

clear; clc;
xdel(winsid());  //close all windows

s = %s;
num = 1;
den = s * (s + 1);
G = syslin('c',num,den);

scf();
a = gca();
a.clip_state = 'on';    //clip the extra nyquist plot
nyquist(G,-1000,1000);
xgrid(color('gray'));
xtitle('Nyquist plot of G(s) = 1 / (s * (s + 1))');
```

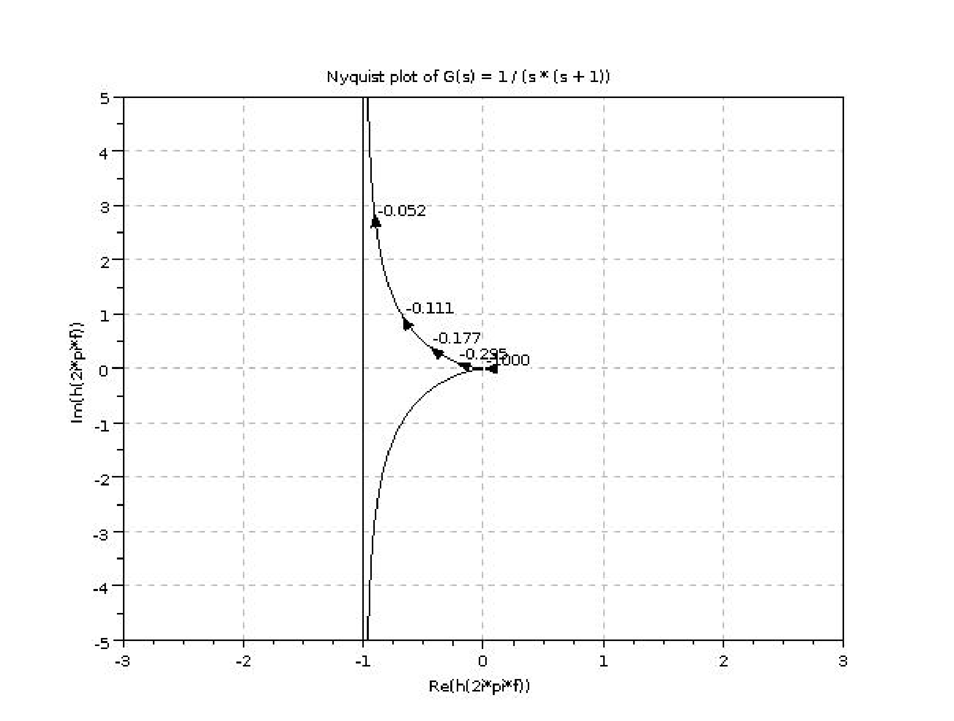

Figure 7.23: Nyquist Plot

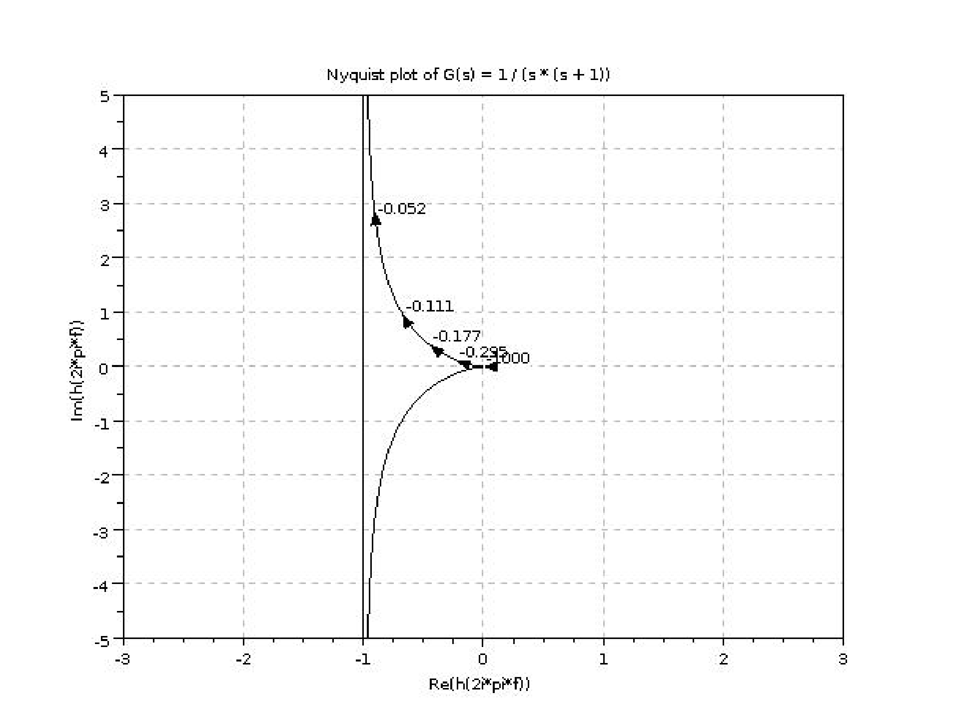

Figure 7.24: Nyquist Plots of system in state space

```
a.data_bounds  = [-3 -5 ; 3 5];
a.box = 'on';
```

Scilab code Exa 7.13 `Nyquist Plot of MIMO system`

```
// Example 7-13
// Nyquist Plot of MIMO system

clear; clc;
xdel(winsid());  //close all windows

```

```
A = [-1 -1 ; 6.5 0];
B = [1 1; 1 0];
C = [1 0; 0 1];
D = [0 0; 0 0];
G = syslin('c',A,B,C,D);
P = clean(ss2tf(G));

subplot(2,2,1);
nyquist(P(1,1),-100,100);
xgrid(color('gray'));
xtitle('Nyquist plot: From U1','Real Axis','To Y1');

subplot(2,2,2);
nyquist(P(2,1),-100,100);
xgrid(color('gray'));
xtitle('Nyquist plot: From U1','Real Axis','To Y2');

subplot(2,2,3);
nyquist(P(1,2),-100,100);
xgrid(color('gray'));
xtitle('Nyquist plot: From U2','Real Axis','To Y1');

subplot(2,2,4);
nyquist(P(2,2),-100,100);
xgrid(color('gray'));
xtitle('Nyquist plot From U2','Real Axis','To Y2');
```

Scilab code Exa 7.14 Nyquist Stability Check

```
// Example 7-14
// Nyquist Stability Check

clear; clc;
```

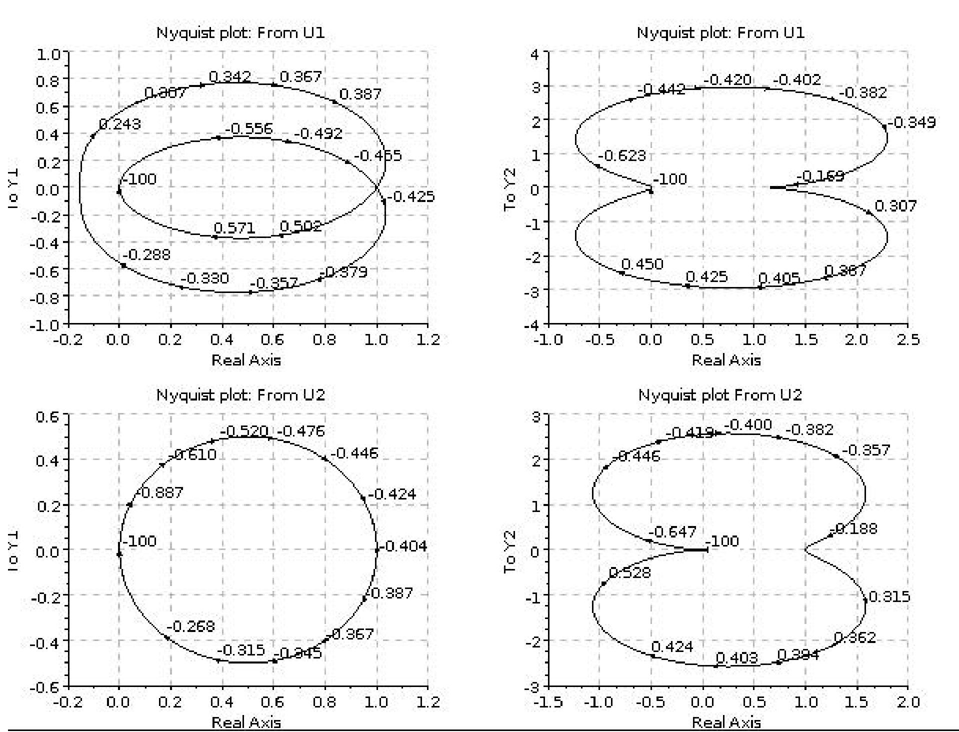

Figure 7.25: Nyquist Plot of MIMO system

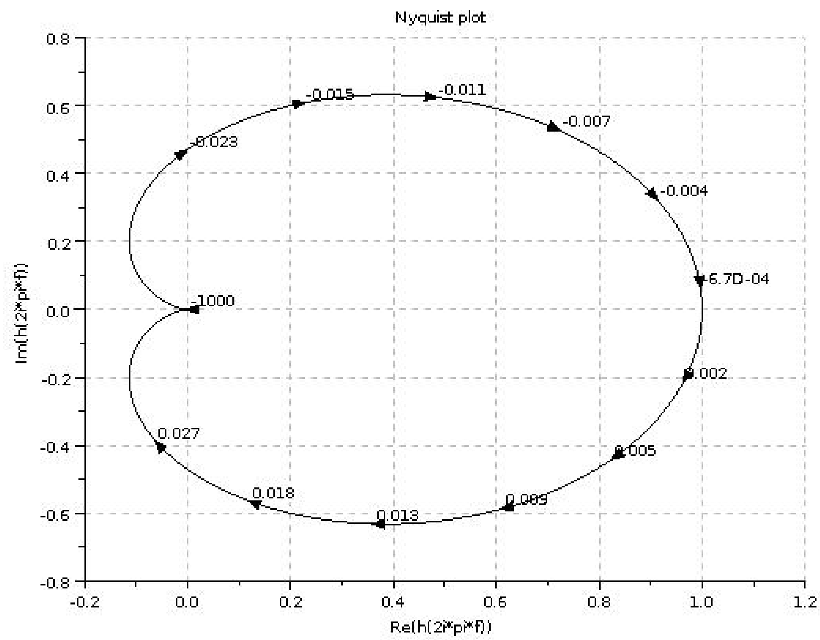

Figure 7.26: Nyquist Stability Check

```
xdel(winsid());  //close all windows

s = %s;
T1 = 5; T2 = 10;

K = 1;
den = (T1*s + 1)*(T2*s + 1);
GH = syslin('c',K,den);
nyquist(GH,-1000,1000);
xgrid(color('gray'));
```

Scilab code Exa 7.19 `Nyquist plot stability check`

```
// Example 7-19
// Nyquist plot stability check

clear; clc;
xdel(winsid()); //close all windows

s = %s;
num = s + 0.5;
den = s^3 + s^2 + 1;
disp(routh_t(den),'routh table ='); // display the routh table
GbyK = syslin('c',num,den); // open loop system

nyquist(GbyK,-1000,1000);
```

---

check Appendix AP 10 for dependency:

`shmargins.sci`

Scilab code Exa 7.20 `Gain and phase margins for different K`

```
// Example 7-20
// Gain and phase margins for different K

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("shmargins.sci");

```

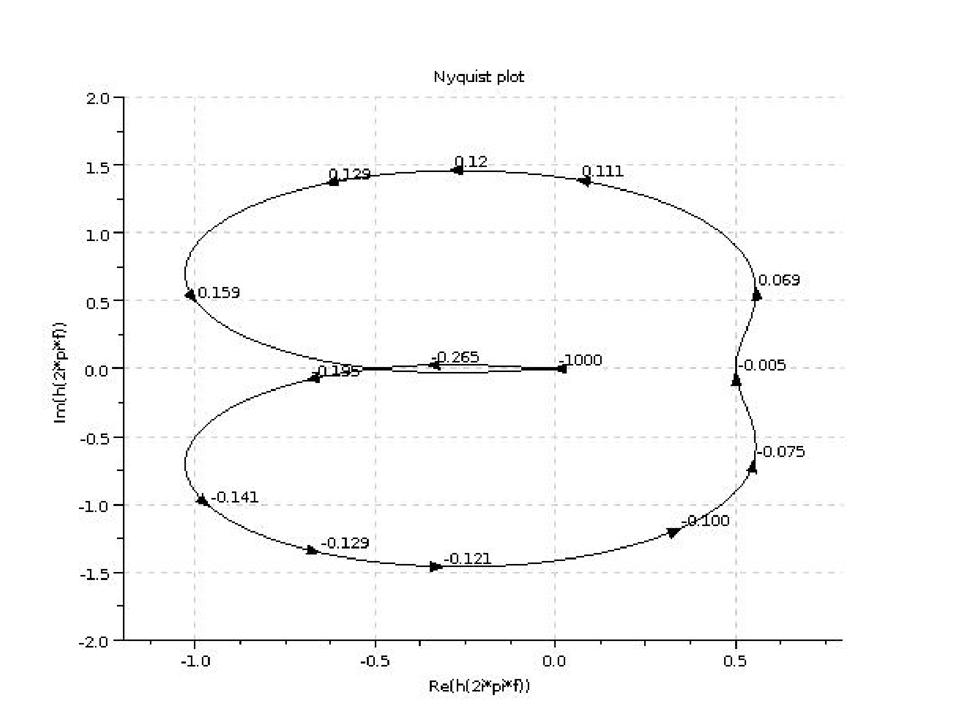

Figure 7.27: Nyquist plot stability check

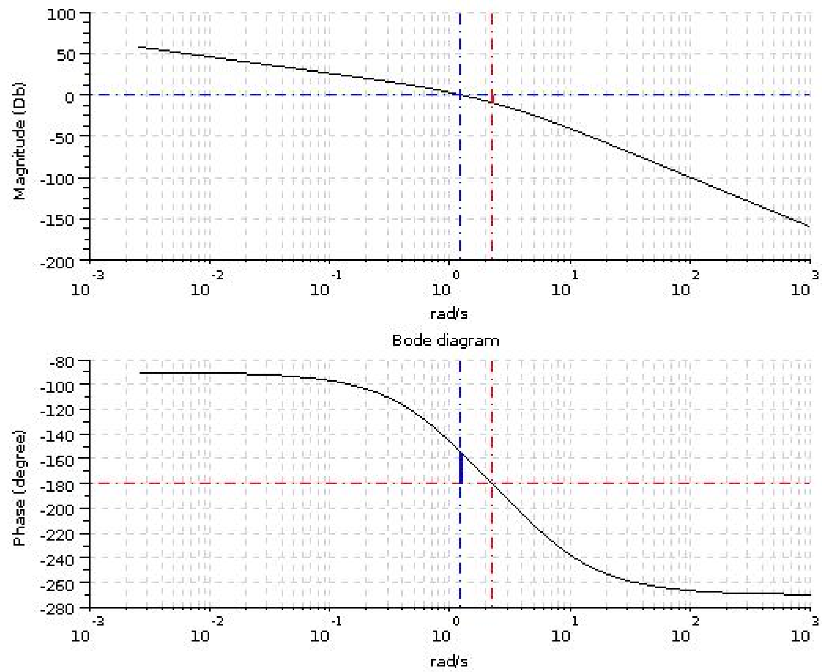

Figure 7.28: Gain and phase margins for different K

```
s = %s /2 / %pi;  // corrected for frequencies in
    rad/s
K = 10;
G = syslin('c', K, s*(s+1)*(s+5));
shmargins(G);
scf();
K = 100;
G = syslin('c', K, s*(s+1)*(s+5));
shmargins(G);
```

check Appendix AP 10 for dependency:

`shmargins.sci`

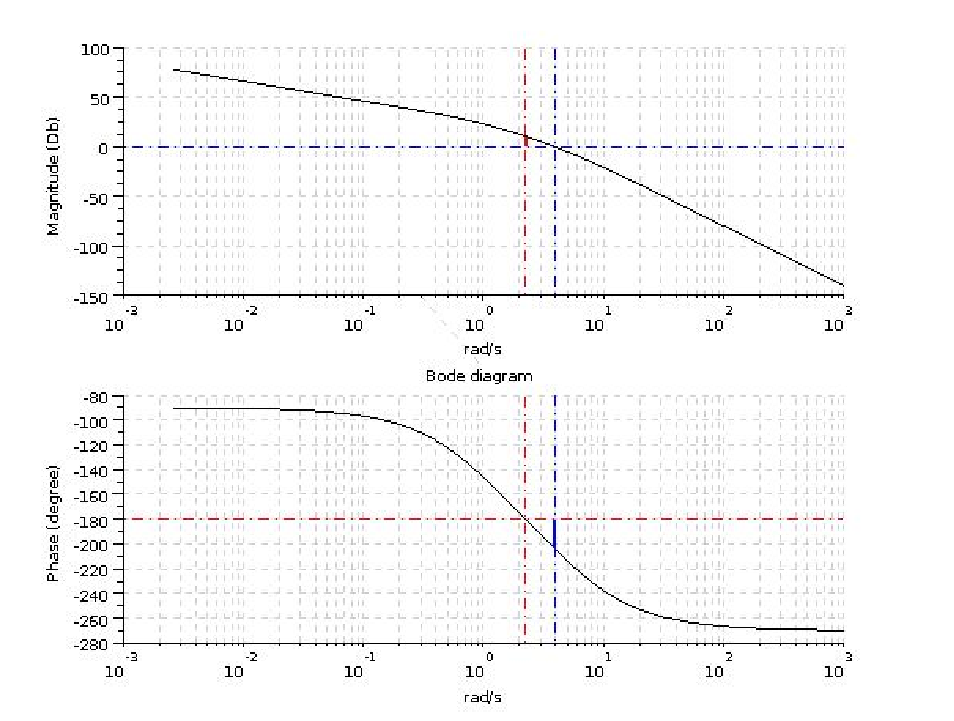

Figure 7.29: Gain and phase margins for different K

Scilab code Exa 7.21 Stability Margins

```
// Example 7-21
// Stability Margins

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("shmargins.sci");

s = %s /2 / %pi; // corrected for frequencies in
   rad/s
num = 20*(s+1);
den =s * (s + 5) * (s^2 + 2*s + 10);
G = syslin('c',num,den);
shmargins(G);
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 7.22 Correlating bandwidth and speed of response

```
// Example 7-22
// Correlating bandwidth and speed of response

clear; clc;
xdel(winsid()); //close all windows
```

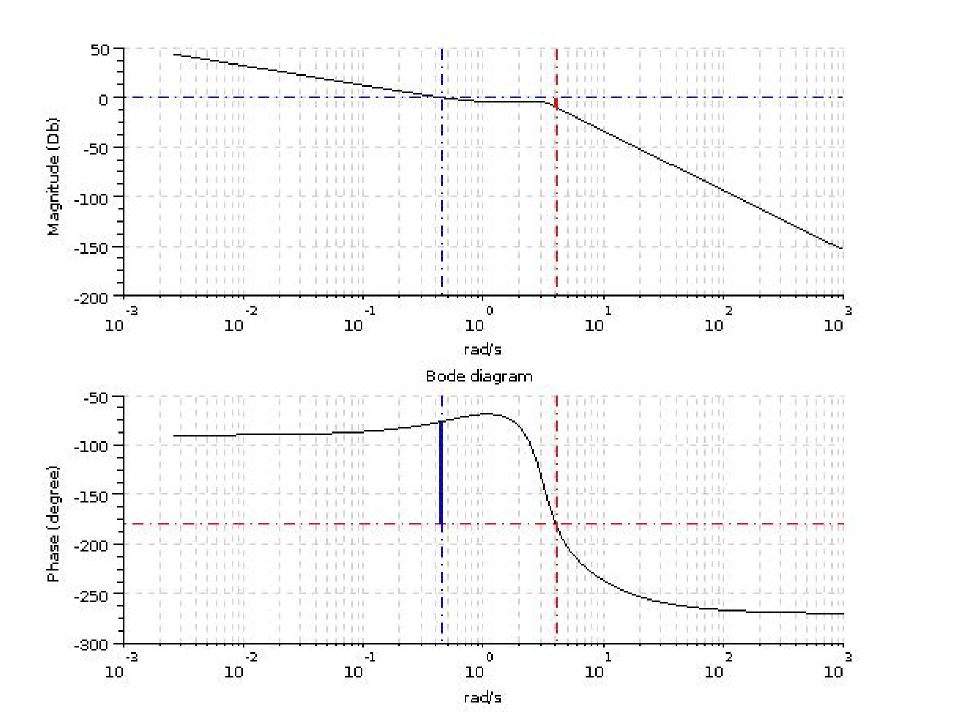

Figure 7.30: Stability Margins

```
// please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

s = %s /2 /%pi; // frequencies in rad/s
G1 = syslin('c',1,s + 1);
G2 = syslin('c',1,3*s + 1);
subplot(2,1,1);
gainplot(G1,0.1,10);
xtitle('system 1 : 1 / (s + 1)','rad/s');
subplot(2,1,2);
gainplot(G2,0.1,10);
xtitle('system 2 : 1 / (3*s + 1)','rad/s');

scf();
t = 0:0.05:1;
u = ones(1,length(t));
subplot(2,1,1);
plotresp(u,t,G1,'');
plotresp(u,t,G2,'Step response of two systems with
    different bandwidth');
xstring(0.1,0.75,'System 1');
xstring(0.35,0.4,'System 2');

subplot(2,1,2);
plotresp(t,t,G1,'');
plotresp(t,t,G2,'Ramp response of two systems with
    different bandwidth');
xstring(0.45,0.35,'System 1');
xstring(0.8,0.45,'System 2');
```

check Appendix AP 11 for dependency:

`freqch.sci`

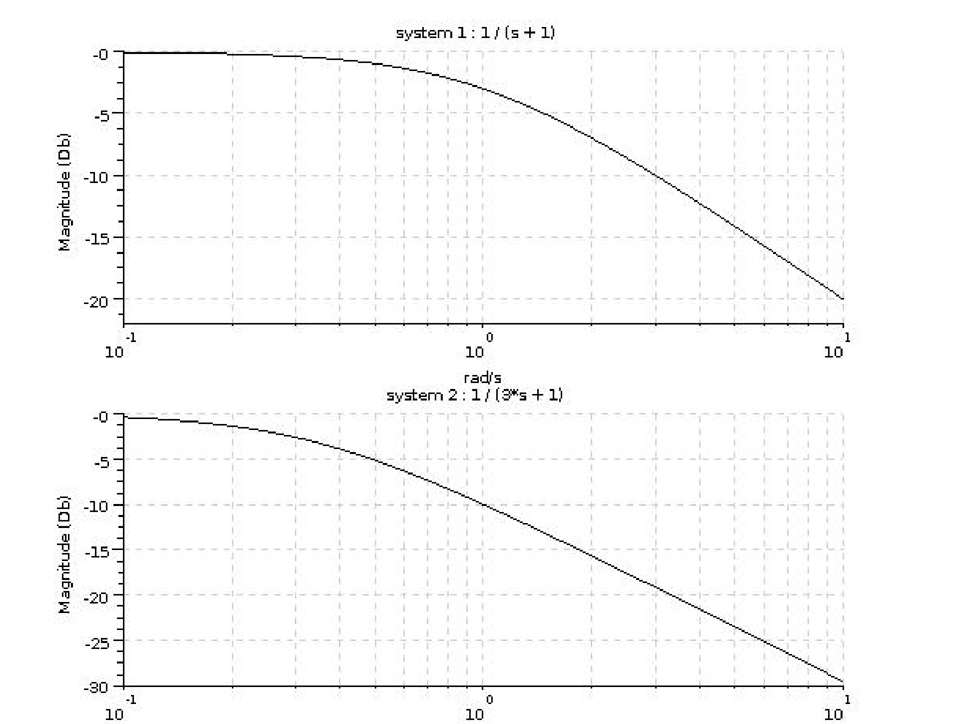

Figure 7.31: Correlating bandwidth and speed of response

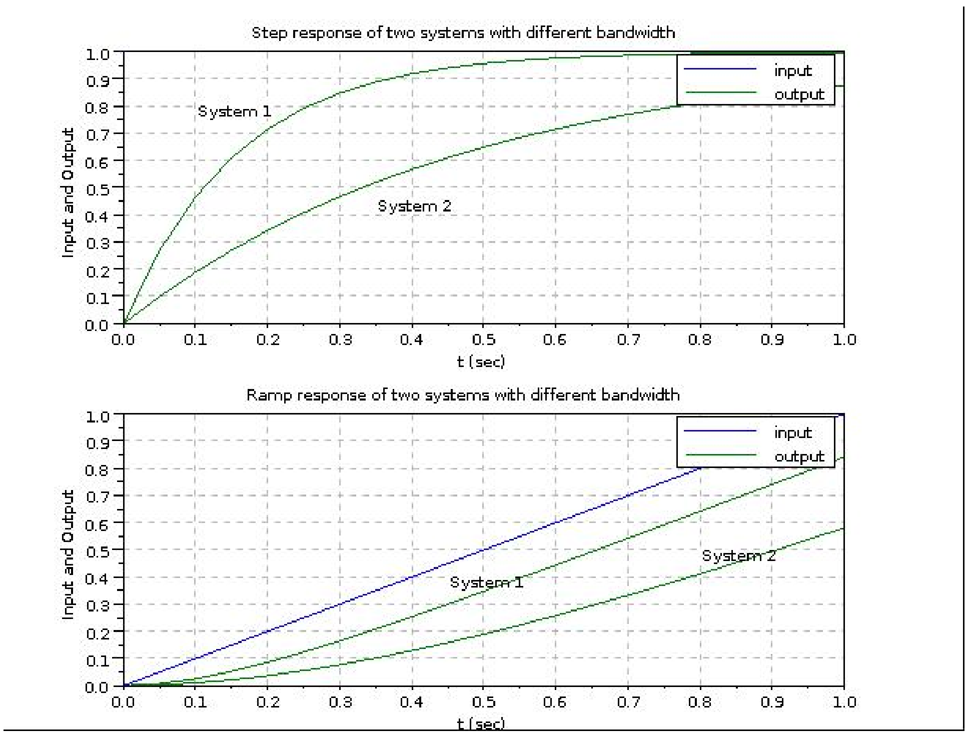

Figure 7.32: Correlating bandwidth and speed of response

Scilab code Exa 7.23 Frequency charecteristics

```
// Example 7-23
// Frequency charecteristics
clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("freqch.sci");

s = %s /2 /%pi; // frequencies in rad/s
G = 1 / (s * (0.5*s + 1) * (s + 1));
H = syslin('c',G /. 1);
omega = logspace(-1,1,200);

[Mr wr bw repf] = freqch(H,omega);
bode(omega,repf);
xtitle('Bode Diagram','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
```

check Appendix AP 9 for dependency:

spolarplot.sci

Scilab code Exa 7.24 Polar and Nichols plot with M circles

```
// Example 7-24
// Polar and Nichols plot with M circles

```

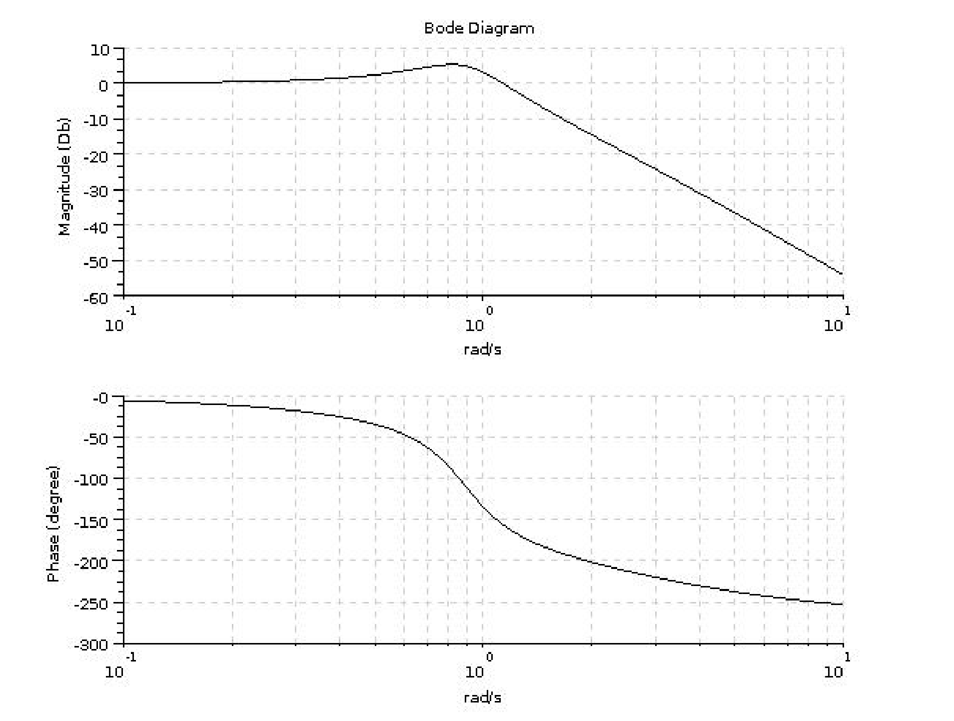

Figure 7.33: Frequency charecteristics

```
clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("spolarplot.sci");

s = %s;
G = syslin('c',1,s*(s+1));
omega = logspace(-2,2,100);
repf = spolarplot(G,omega);

scf();
black(omega,repf);
chart([1.4],[],list(1,0));
xgrid(color('gray'));
xstring(-150,8,'Mr = 1.4')
```

Scilab code Exa 7.25 Verifying experimentally derived Transfer function

```
// Example 7-25
// Verifying experimentally derived Transfer
   function

clear; clc;
xdel(winsid()); //close all windows

s = %s;
num = 320*(s + 2);
den = s * (s + 1) * (s^2 + 8*s + 64);
G = syslin('c',num,den);
```

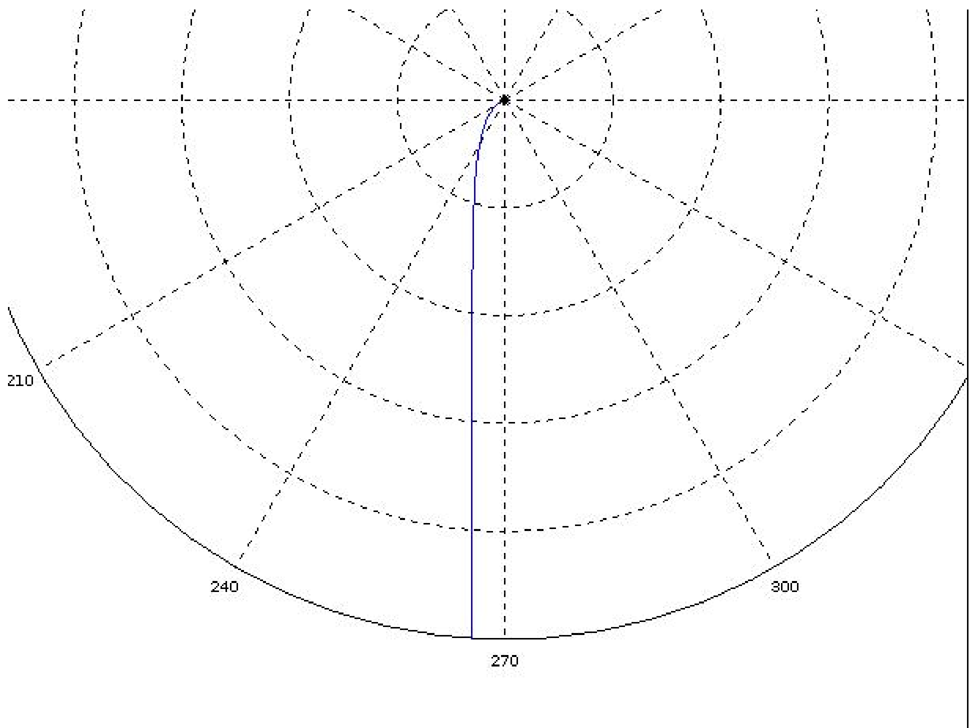

Figure 7.34: Polar and Nichols plot with M circles

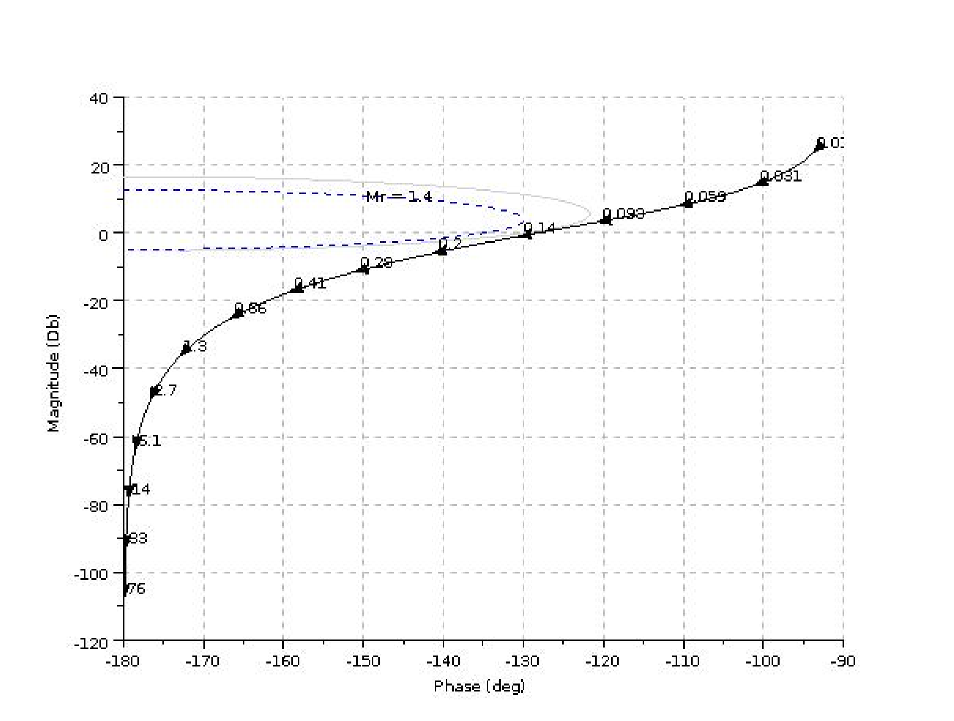

Figure 7.35: Polar and Nichols plot with M circles

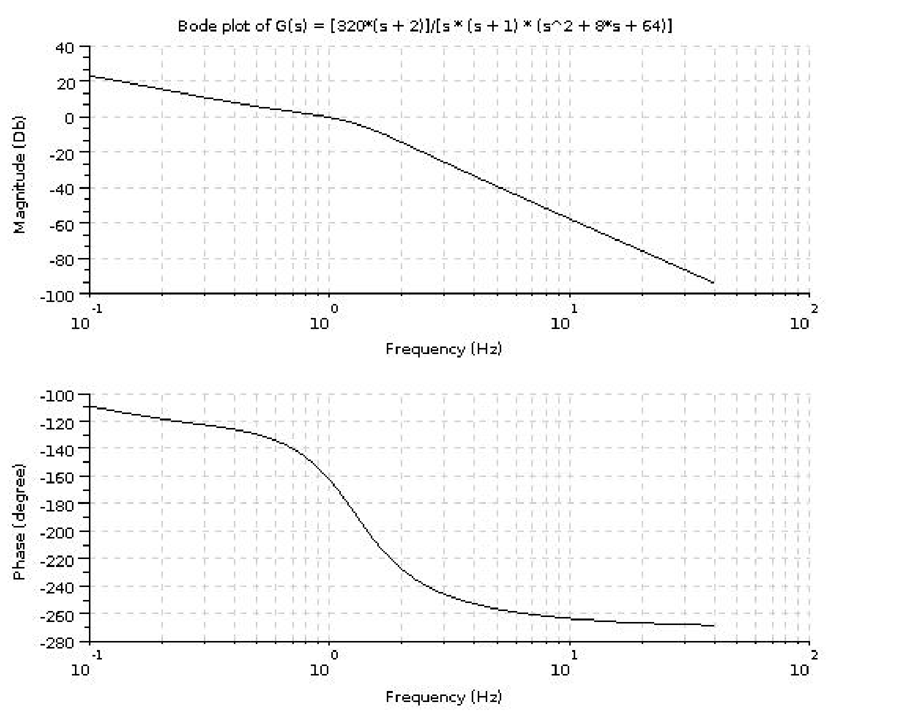

Figure 7.36: Verifying experimentally derived Transfer function

```

bode(G,0.1,40);
xtitle('Bode plot of G(s) = [320*(s + 2)]/[s * (s +
   1) * (s^2 + 8*s + 64)]');
```

Scilab code Exa 7.26.1 `Design of Lead compensator with Bode plots`

```
// Example 7-26-1
// Design of Lead compensator with Bode plots

clear; clc;
```

```
xdel(winsid());  //close all windows
mode(0);

// please edit the path
// cd "/<your code directory>/";
// exec("shmargins.sci");

s = %s/2/%pi;
G = 4 / (s * (s + 2));
Kv = 20;
K = Kv / horner(s * G,0)

GK = syslin('c',K * G);

[gm, gcrw, pm, pcrw] = shmargins(GK);
// required specification is pm = 50 degrees
phi = 50 - pm + 6  // 6 deg compensation
sn = sind(phi);
alpha = (1 - sn)/(1 + sn)

wc = 9; // new gain crossover freq.
z = wc * sqrt(alpha) // z = 1 / T
p = wc / sqrt(alpha) // p = 1 / (alpha*T)
Kc = K / alpha
disp(Kc * (%s + z)/(%s + p),'Gc = ');
Gc = Kc * (s + z)/(s + p);
GGc = syslin('c',Gc * G);
scf();
shmargins(GGc);
```

check Appendix AP 10 for dependency:

`shmargins.sci`

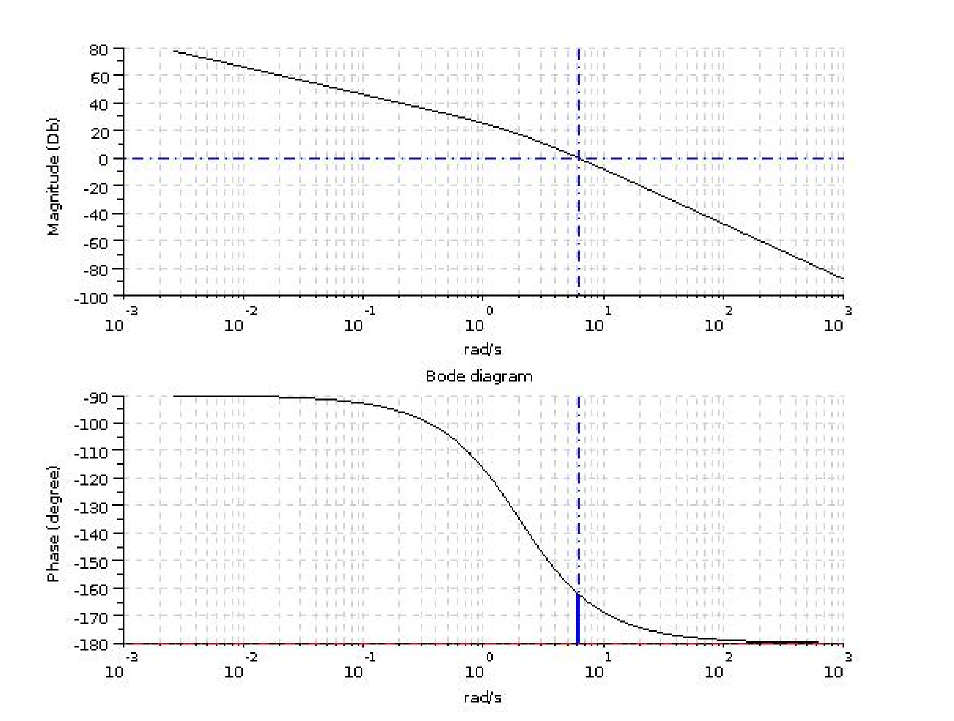

Figure 7.37: Design of Lead compensator with Bode plots

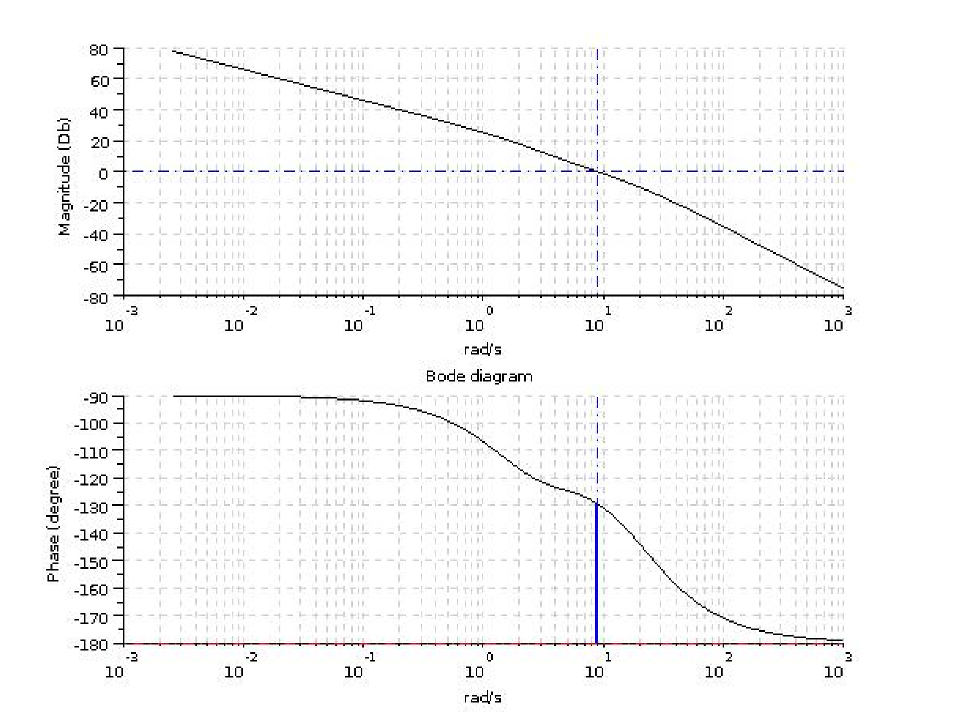

Figure 7.38: Design of Lead compensator with Bode plots

Scilab code Exa 7.26.2 Evaluating Lead compensated system

```
// Example 7-26-2
// Evaluating Lead compensated system

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory >/";
// exec("plotresp.sci");

s = %s;
G = 4 / (s * (s + 2));

Kc = 42.104125;
z  = 4.3861167;
p = 18.467361;
Gc = Kc * (s + z)/(s + p);
GGc = G*Gc;

H = syslin('c',G /. 1);
Hc = syslin('c',GGc /. 1);

t = 0:0.05:5;
u1 = ones(1,length(t)); //step response
u2 = t;                 //ramp response

subplot(2,1,1);plotresp(u1,t,H,'');
plotresp(u1,t,Hc,'Unit step response');
xstring(0.65,0.55,'uncompensated system');
xstring(0.1,1.2,'compensated system');
subplot(2,1,2);plotresp(u2,t,H,'');
plotresp(u2,t,Hc,'Unit ramp response');
```

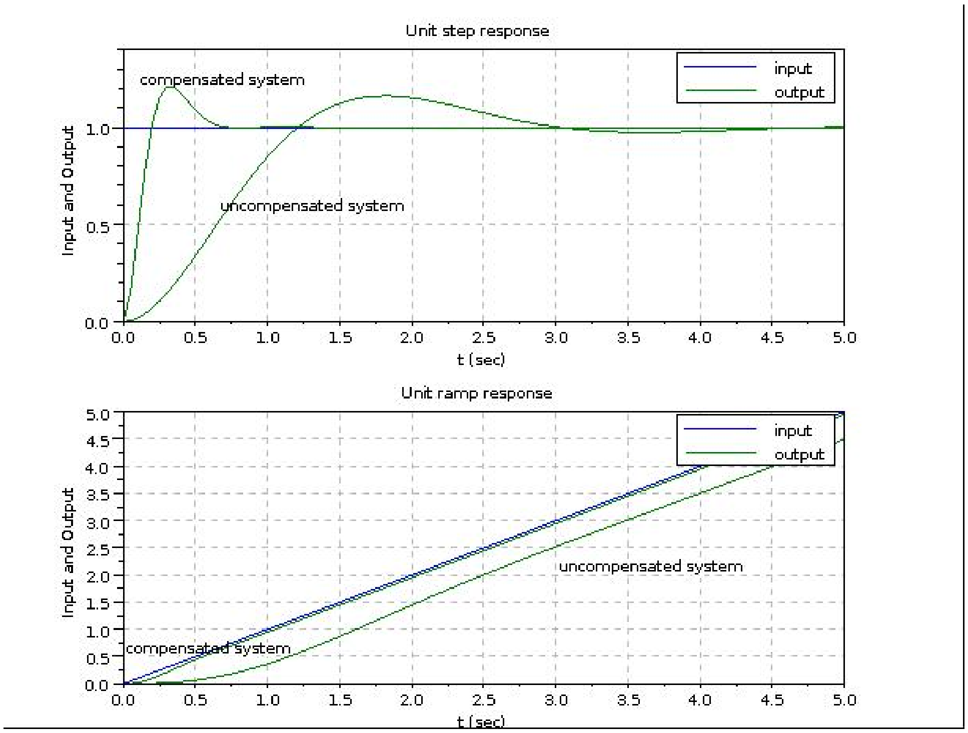

Figure 7.39: Evaluating Lead compensated system

```
xstring(3.0,2.0,'uncompensated system');
xstring(0,0.5,'compensated system');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 7.27.1 Design of Lag compensator with Bode plots

```
// Example 7-27-1
// Design of Lag compensator with Bode plots

```

```
clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "/<your code directory >/";
// exec("shmargins.sci");

s = %s/2/%pi;
G = 1 / (s * (s + 1) * (0.5*s + 1));
Kv = 5;
K = Kv / horner(s * G,0)

GK = syslin('c',K * G);

[gm, gcrw, pm, pcrw] = shmargins(GK);
// required specification is pm = 40 degrees

wc = 0.5; // new gain crossover freq.
beta = 10
z = 0.1      // z = 1 / T is chosen one octave less
p = z / beta
Kc = K / beta
disp(Kc * (%s + z)/(%s + p),'Gc = ');
Gc = Kc * (s + z)/(s + p);
GGc = syslin('c',Gc * G);
scf();
shmargins(GGc);
```

check Appendix AP 10 for dependency:

`shmargins.sci`

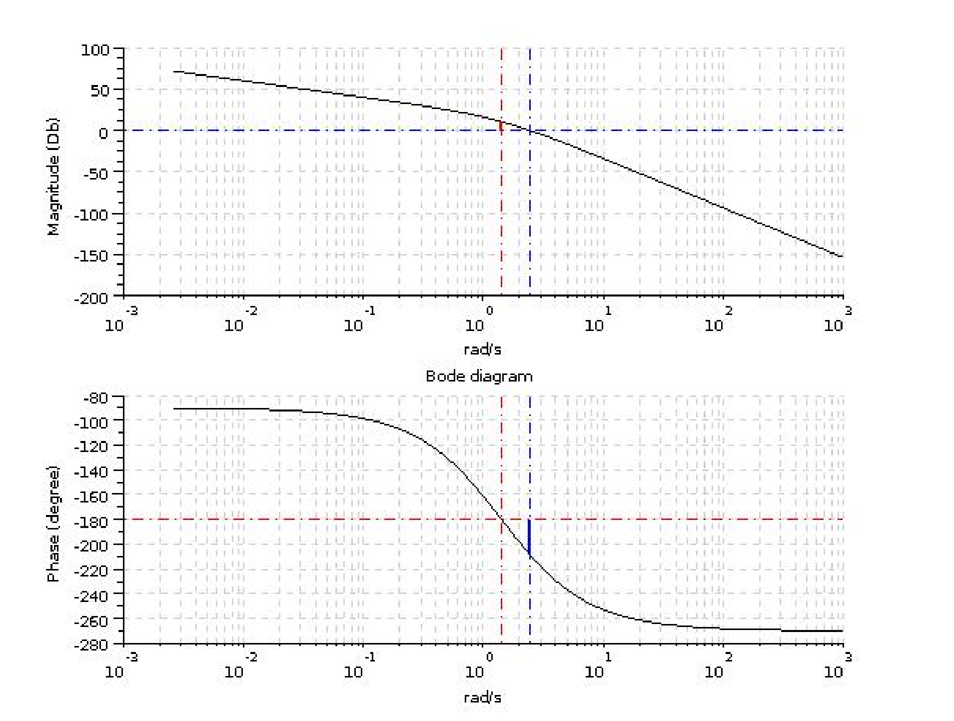

Figure 7.40: Design of Lag compensator with Bode plots

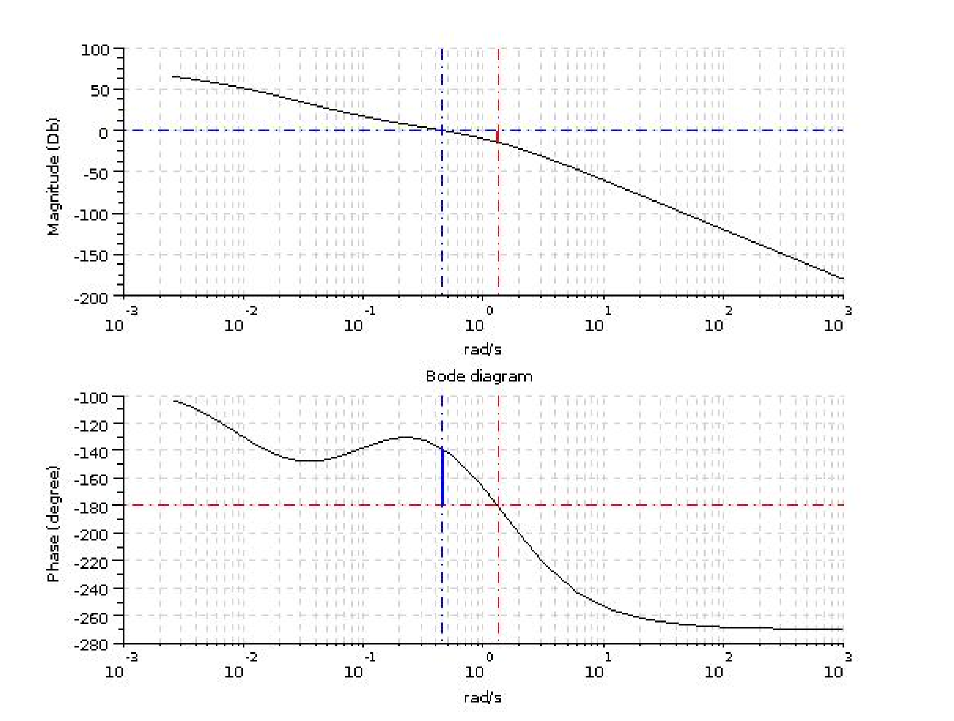

Figure 7.41: Design of Lag compensator with Bode plots

Scilab code Exa 7.27.2 Evaluating Lag compensated system

```
// Example 7-27-2
// Evaluating Lag compensated system

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

s = %s;
G = 1 / (s * (s + 1) * (0.5*s + 1));

Kc = 0.5;
z  = 0.1;
p  = 0.01;
Gc = Kc * (s + z)/(s + p);
GGc = G*Gc;

H = syslin('c',G /. 1);
Hc = syslin('c',GGc /. 1);

t = 0:0.5:40;
u1 = ones(1,length(t)); //step response

subplot(2,1,1);plotresp(u1,t,H,'');
plotresp(u1,t,Hc,'Unit step response');
xstring(2.5,0.55,'uncompensated system');
xstring(0.1,1.3,'compensated system');

t = 0:0.5:30;
u2 = t;                    //ramp response
subplot(2,1,2);plotresp(u2,t,H,'');
plotresp(u2,t,Hc,'Unit ramp response');
xstring(15,13,'uncompensated system');
xstring(14,20,'compensated system');
```

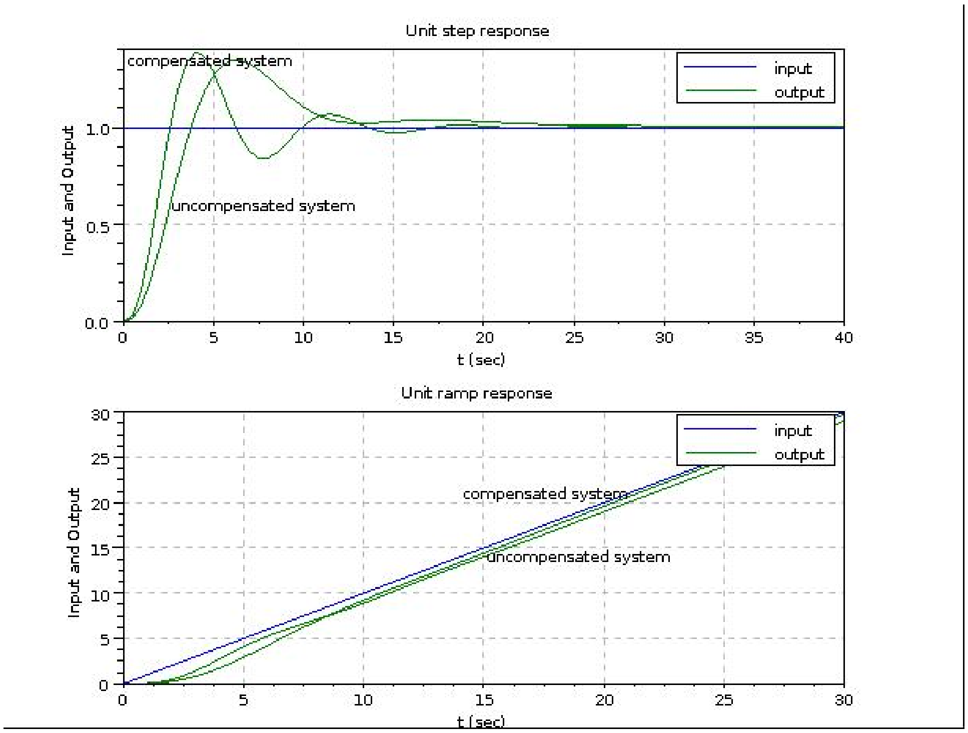

Figure 7.42: Evaluating Lag compensated system

---

check Appendix AP 2 for dependency:

plotresp.sci

Scilab code Exa 7.28.1 `Design of Lag lead compensation with Bode plots`

```
// Example 7-28-1
// Design of Lag - lead compensation with Bode plots

clear; clc;
xdel(winsid()); //close all windows
```

```
mode(0);

// please edit the path
// cd "/<your code directory>/";
// exec("shmargins.sci");

s = %s /2 /%pi ;
G = 1 / (s * (s + 1) * (s + 2));
Kv = 10;
K = Kv / horner(s * G,0)
GK = syslin('c',K * G);

[gm, gcrw, pm, pcrw] = shmargins(GK);
wc = 1.5; // new gain crossover freq.

// required specification is pm = 50 degrees
phi = 55  // 6 deg compensation
sn = sind(phi);
beta = (1 + sn)/(1 - sn)

z2 = wc /10; // z2 = 1 / T2 :1 decade below our new gain cross freq.
p2 = z2 / beta;

disp((%s + z2)/(%s + p2),'Gclead = ');
Gclead = (s + z2)/(s + p2);

z1 = 0.7 ; //corner frequencies are around w = 7 <-> -20db
p1 = 7;
disp((%s + z1)/(%s + p1),'Gclag = ');
Gclag = (s + z1)/(s + p1);

Gc = K * Gclag * Gclead;
GGc = syslin('c',Gc * G);
scf();
shmargins(GGc);
```

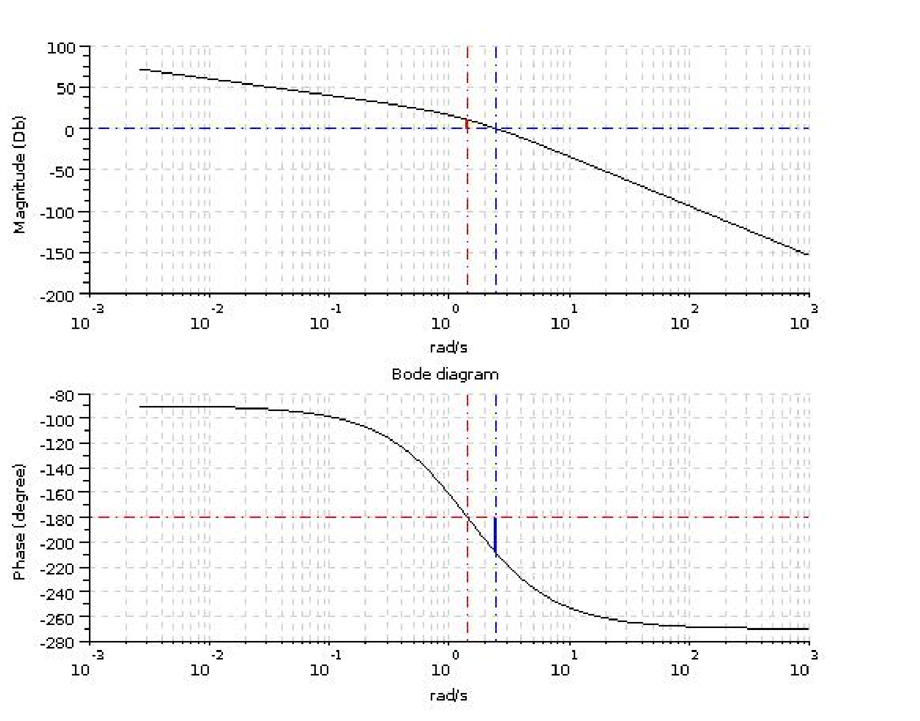

Figure 7.43: Design of Lag lead compensation with Bode plots

check Appendix AP 10 for dependency:

`shmargins.sci`

Scilab code Exa 7.28.2 `Evaluating Lag Lead compensated system`

```
// Example 7-26-2
// Evaluating Lag Lead compensated system

```

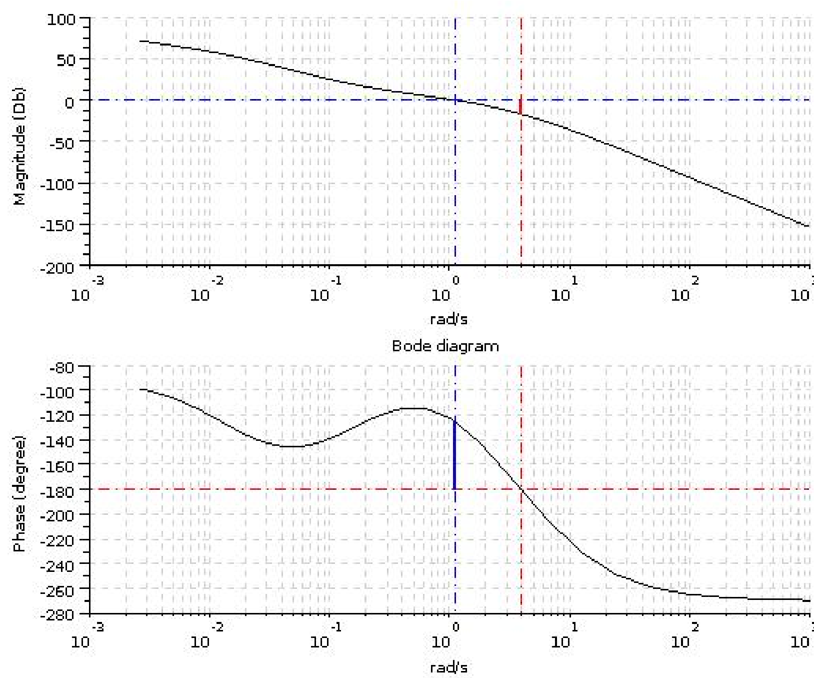

Figure 7.44: Design of Lag lead compensation with Bode plots

```
clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "/<your code directory>/";
// exec("plotresp.sci");

s = %s;
G = 1 / (s * (s + 1) * (s + 2));

Gc = 20 * (s + 0.7) * (s + 0.15) / (s + 7) / (s +
    0.015);
GGc = G*Gc;

H = syslin('c',G /. 1);
Hc = syslin('c',GGc /. 1);

t = 0:0.1:30;
u1 = ones(1,length(t)); //step response
u2 = t;                 //ramp response

subplot(2,1,1);plotresp(u1,t,H,'');
plotresp(u1,t,Hc,'Unit step response');
xstring(3,0.8,'uncompensated system');
xstring(0.7,0.6,'compensated system');
subplot(2,1,2);plotresp(u2,t,H,'');
plotresp(u2,t,Hc,'Unit ramp response');
xstring(10,7,'uncompensated system');
xstring(2,0.5,'compensated system');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

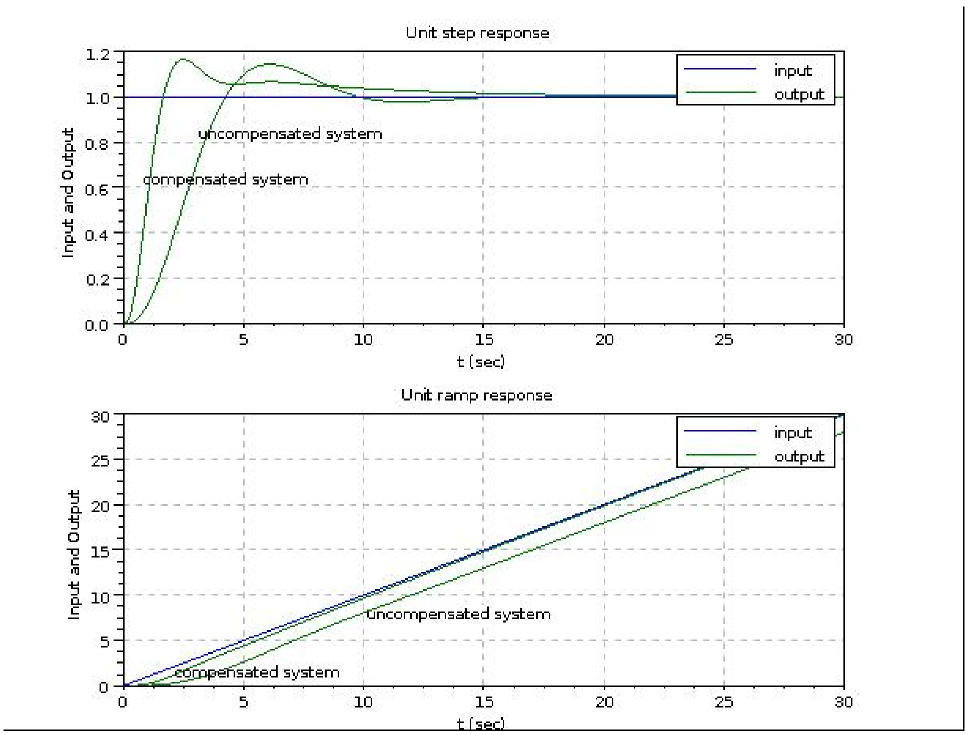

Figure 7.45: Evaluating Lag Lead compensated system

Chapter 8

PID Controllers and Modified PID Controllers

Scilab code Exa 8.i.1 `PID Design with Frequency Response`

```
// Illustration 8.1
// PID Design with Frequency Response

clear; clc;
xdel(winsid()); //close all windows
mode(0);
// please edit the path
// cd "<your code directory>";
// exec("plotresp.sci");

s = %s;
G = syslin('c',1,s^2 + 1);
Kv = 4;
K =  Kv / abs(horner(G,0))

```

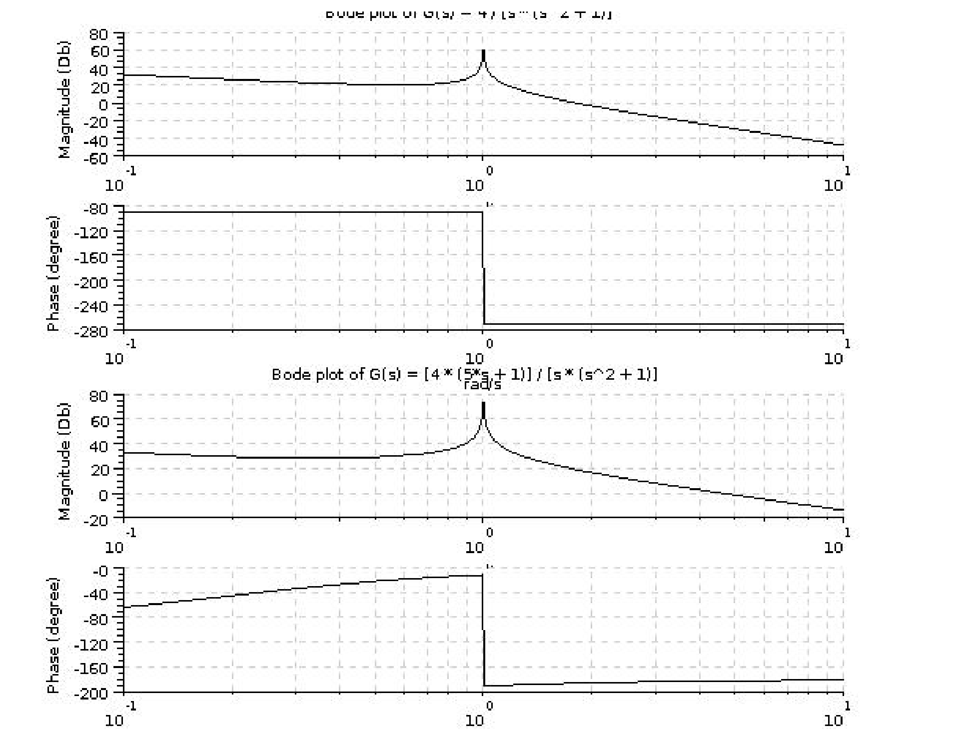

Figure 8.1: PID Design with Frequency Response

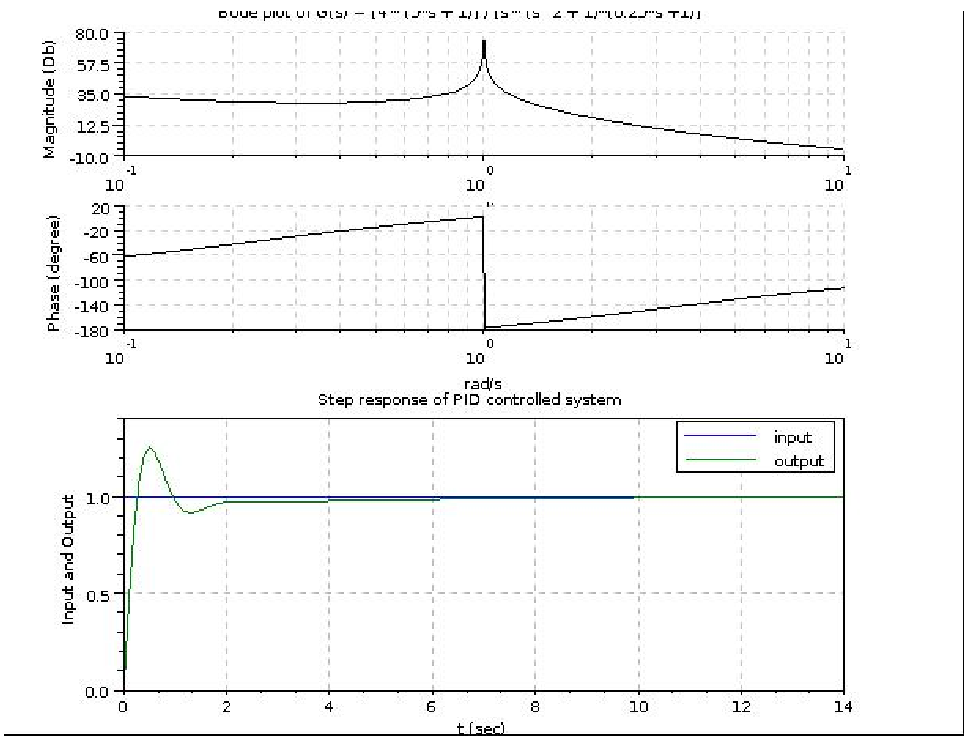

Figure 8.2: PID Design with Frequency Response

```
// Step 1 : Gain adjust
G2 = G * K / s
G2w = syslin('c', horner(G2, %s/2/%pi) );// correction for frequences in rad/s

omega = calfrq(G2w,0.1,10);    // discretises such that the peak is                                                    // well represented
[db phi] = dbphi(repfreq(G2w,omega));
phi( 53:99 ) = -270;
subplot(2,1,1);  bode(omega,db,phi);
xtitle('Bode plot of G(s) = 4 / [s * (s^2 + 1)]','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
disp(p_margin(G2w),'Phase margin of G2 =');

// Step 2:
a = 5 // a is chosen to be 5;
G3 = G2 * (a*s + 1)
G3w = syslin('c', horner(G3, %s/2/%pi) );
subplot(2,1,2);  bode(G3w,0.1,10);
xtitle('Bode plot of G(s) = [4 * (5*s + 1)] / [s * (s^2 + 1)]','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
disp(p_margin(G3w),'Phase margin of G3 =');

// Step 3
scf();
b = 0.25
G4 = G3 * (b*s + 1)
G4w = syslin('c', horner(G4, %s/2/%pi) );
subplot(2,1,1);  bode(G4w,0.1,10);
xtitle('Bode plot of G(s) = [4 * (5*s + 1)] / [s * (s^2 + 1)*(0.25*s +1)]','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
disp(p_margin(G4w),'Phase margin of G4 =');

```

```
C  = syslin('c',G4 /. 1)
disp(roots(C.den),'closed loop poles =');
t = 0:0.1:14;
u = ones(1,length(t));
subplot(2,1,2); plotresp(u,t,C,'Step response of PID
    controlled system');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 8.a.5 `PID design`

```
// Example A-8-5
// PID design

clear; clc;
xdel(winsid()); //close all windows
mode(0);
// please edit the path
// cd "";
// exec("plotresp.sci");
// exec("stepch.sci");

s = %s;
zeta = 0.5       // dominant pole charecteristics
wn = 4
sigma = zeta*wn;
ts = 4 /(zeta*wn);
disp(ts,'settling time approximate (ts) =');

D = (s + 10) * (s^2 + 2*zeta*wn*s + wn^2);
cf = coeff(D);

K = cf(1)
a_plus_b  = (cf(2) - 9) / K
```

```
ab = (cf(3) - 3.6) / K

Gc = K * (ab * s^2 + a_plus_b *s+ 1) / s
CbyD = syslin('c',s,D)

CbyR = syslin('c',numer(Gc),D)

t = 0:0.05:5;
u = ones(1,length(t));
plotresp(u,t,CbyD,'Response to step disturbance
   input');
a = gca(); a.data_bounds = [0 ,-4D-3; 5 ,14D-3];
scf();
[Mp ,tp ,tr ,ts] = stepch(CbyR,0,5,0.05,0.02);
disp(Mp,'Max overshoot =');
disp(ts,'settling time actual (ts) =');
```

check Appendix AP 2 for dependency:

plotresp.sci

check Appendix AP 8 for dependency:

stepch.sci

Scilab code Exa 8.a.6 PID design

```
// Example A-8-6
// PID Design

clear; clc;
xdel(winsid()); //close all windows
mode(0);
```

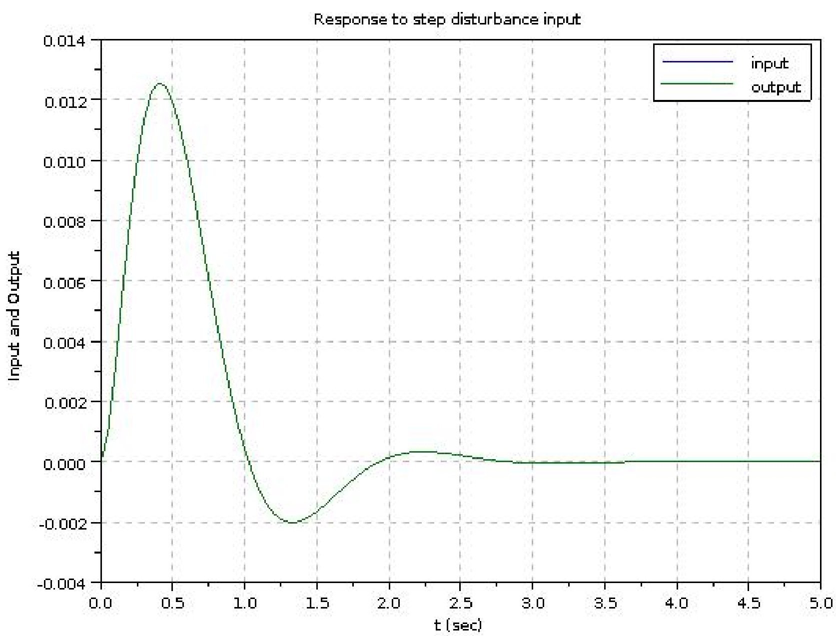

Figure 8.3: PID design

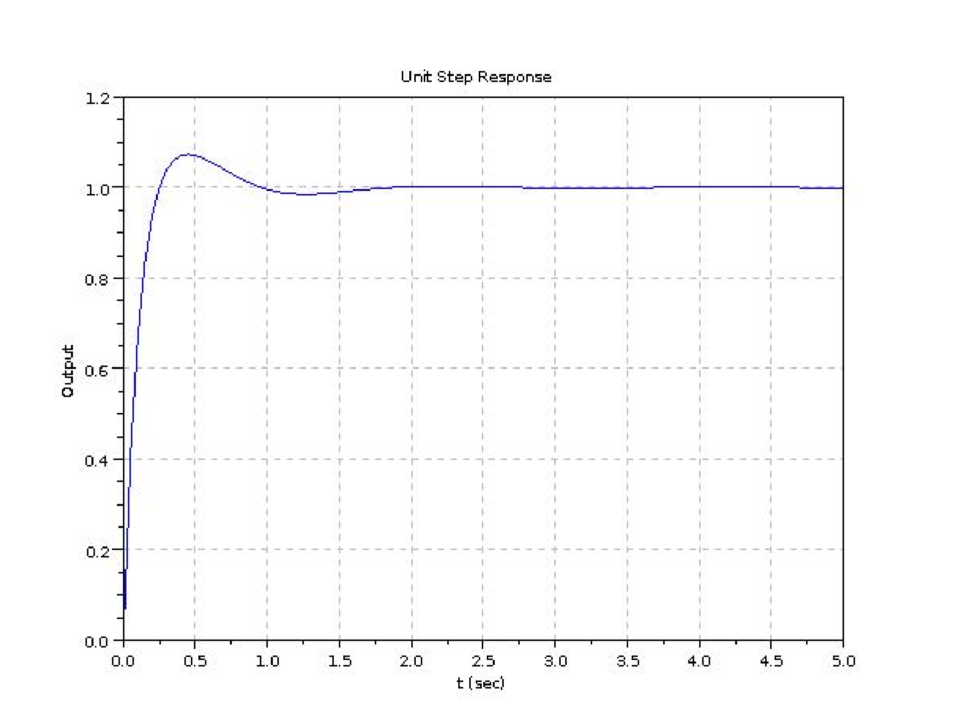

Figure 8.4: PID design

```
// please edit the path
// cd "<your code directory>";
// exec("plotresp.sci");
// exec("rootl.sci")

s = %s;
G = syslin('c',1,s^2 + 1);
dp = -1 + sqrt(3)*%i;

angdef = 180 - phasemag(horner(G*(s+1)/s,dp))
// Determining b
b = 1 + sqrt(3)*cotd(angdef)
Gc1 = (s + 1) * (s + b) / s;
K = 1/ abs(horner(G*Gc1,dp))
Gc = K * Gc1

evans(G*Gc1,50);
xgrid();
a = gca();
a.data_bounds = [-5 -3; 1 3];
a.children(1).visible = 'off';
xtitle('Root locus plot of open loop system');


C = syslin('c',G*Gc /. 1);
disp(C,'closed loop system =');
scf();
t = 0:0.05:12;
u = ones(1,length(t));
plotresp(u,t,C,'Unit step response of compensated system');
```

check Appendix AP 2 for dependency:

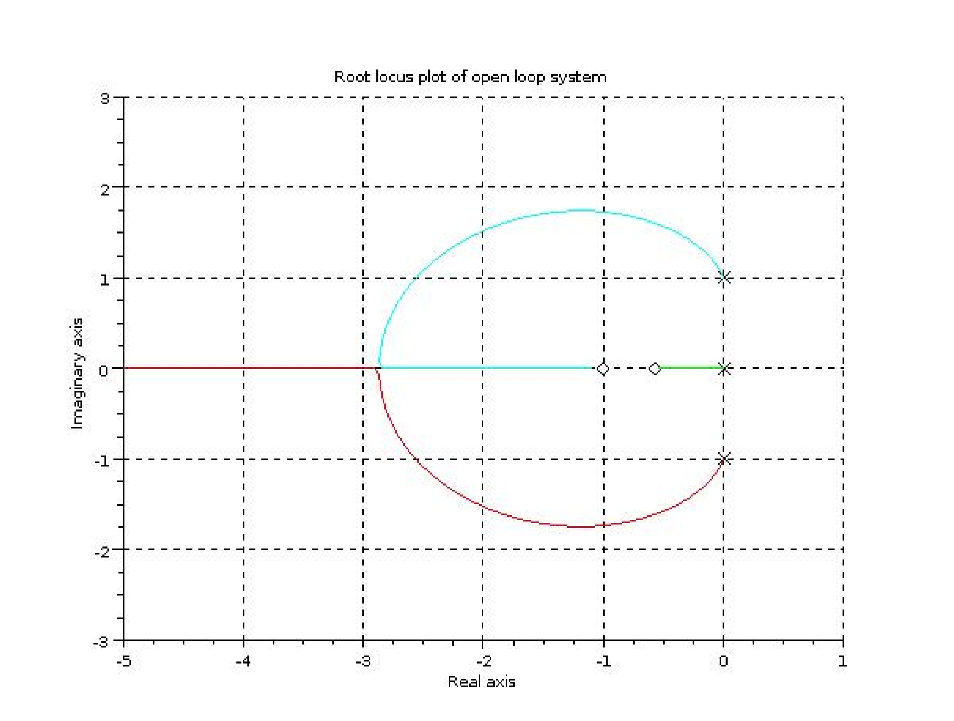

Figure 8.5: PID design

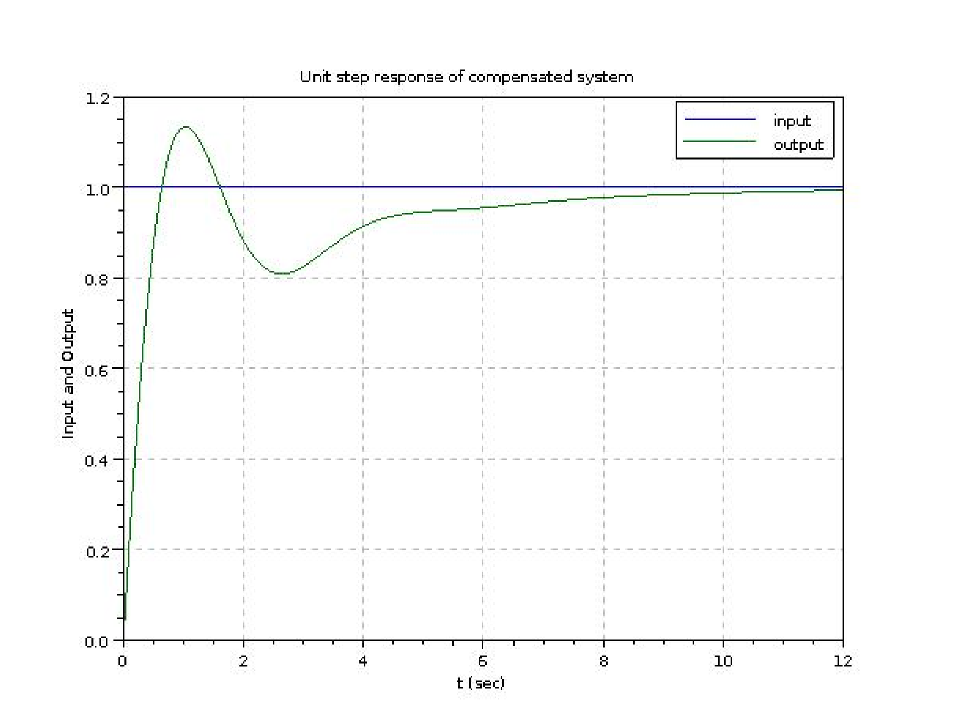

Figure 8.6: PID design

plotresp.sci

check Appendix AP 7 for dependency:

rootl.sci

Scilab code Exa 8.a.7.1 PID Design with Frequency Response

```
// Example A-8-7-1
// PID Design with Frequency Response

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "<your code directory>";
// exec("plotresp.sci");

s = %s;
Gp = syslin('c',s + 0.1,s^2 + 1);
Kv = 4;
K =  Kv / abs(horner(Gp,0))

// Step 1 : Gain adjust
G1 = Gp * K / s
G1w = syslin('c', horner(G1, %s/2/%pi) );//
    correction for frequences in rad/s


subplot(2,1,1);  bode(G1w);
xtitle('Bode plot of G(s) = 40*(s + 0.1)/ [s*(s^2 +
    1)]','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
disp(p_margin(G1w),'Phase margin of G =');

```

```
// Step 2:
a = 0.1526;
GGc = G1 * (a*s + 1)
GGcw = syslin('c', horner(GGc, %s/2/%pi) );
subplot(2,1,2);  bode(GGcw,0.1,10);
xtitle('Bode plot of G*Gc = [4 *(0.1526*s + 1)*(s +
   0.1)]/[s*(s^2 + 1)]','rad/s');
a = gcf();set(a.children(1).x_label,'text','rad/s');
disp(p_margin(GGcw),'Phase margin of G*Gc =');
disp(g_margin(GGcw),'Gain margin of G*Gc =');

scf();
C  = syslin('c',GGc /. 1)
disp(roots(C.den),'closed loop poles =');
t = 0:0.05:10;
u = ones(1,length(t));
subplot(2,1,1); plotresp(u,t,C,'Step response of PID
    controlled system');
subplot(2,1,2); plotresp(t,t,C,'Ramp response of PID
    controlled system');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 8.a.12 `Computing optimal solution`

```
// Example A-8-12
// Computing optimal solution

clear; clc;
xdel(winsid());  //close all windows
```

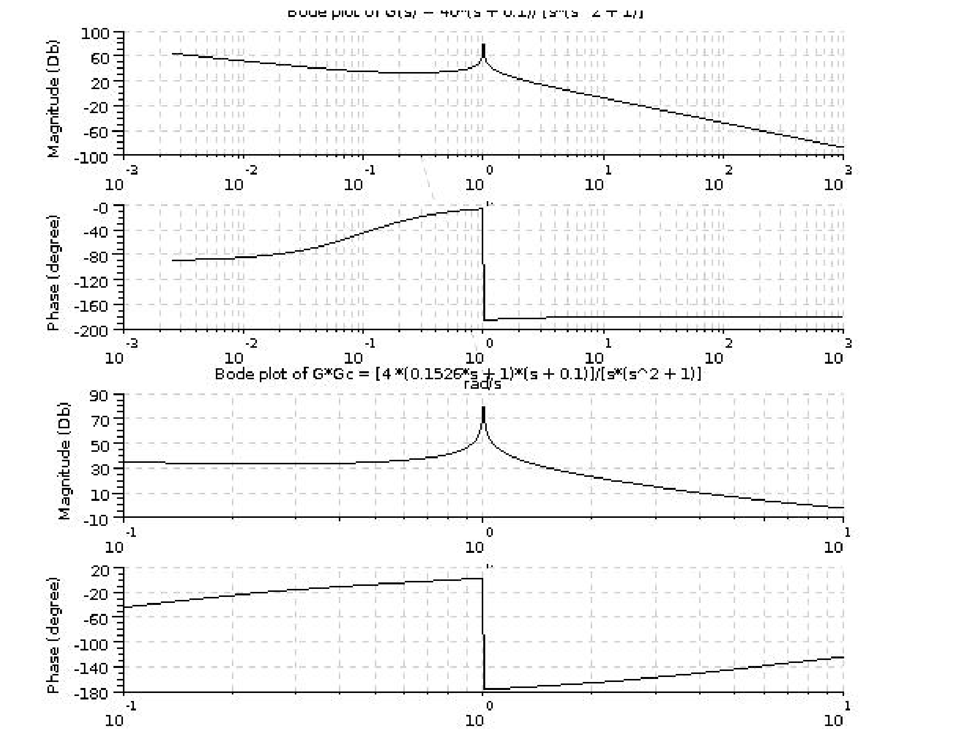

Figure 8.7: PID Design with Frequency Response

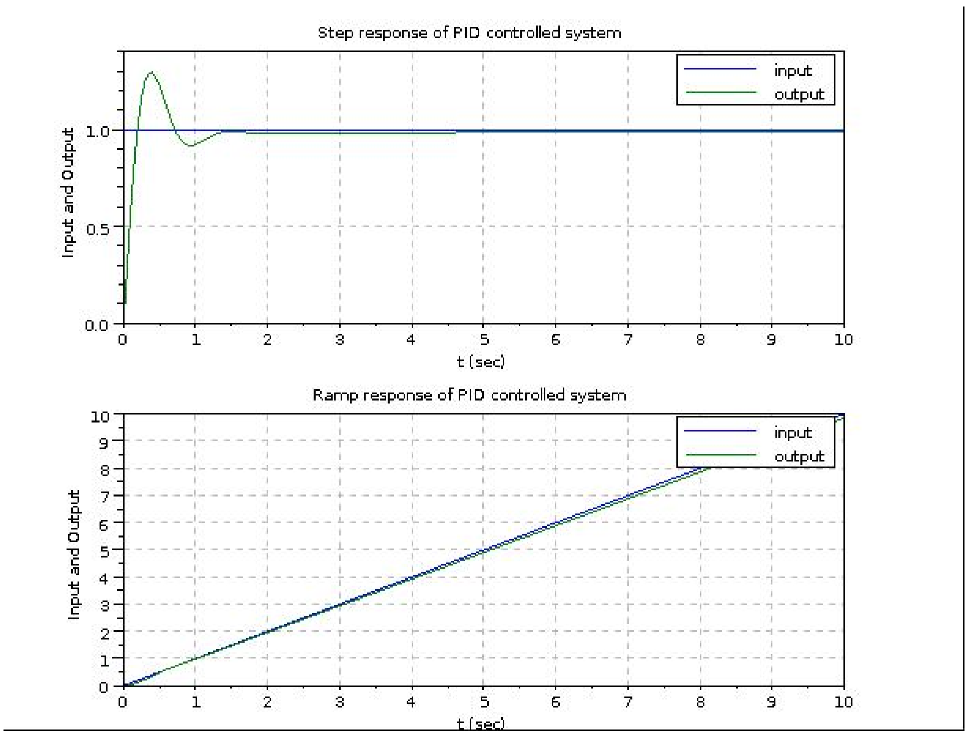

Figure 8.8: PID Design with Frequency Response

```
s = %s;
t = 0:0.1:5; u = ones(1,length(t));
t1 = 0:0.01:5;N =length(t1);  u1 = ones(1,N);

k = 0;
mprintf('Processing ...\n');
for K = 50:-1:2
  for a = 2:-0.05:0.05
    num = K * ((s + a)^2) ;
    den = s * s * (s^2 + 6*s + 5);
    G = syslin('c',num,num + den);
    y = csim(u,t,G);
    m = max(y);
    if m < 1.1 & m > 1.00 then;
      y = csim(u1,t1,G);
      if m < 1.1 & m > 1.02 then;
        l = N;
        while y(l) > 0.98 & y(l) < 1.02 ; l = l-1; end
        ts = (l-1)*0.01;
        if ts < 3.0;
          k= k + 1;
          solution(k,:) = [K a m ts];
        end
      end
    end

  end
 mprintf('completed %d%%\n',(50 - K)/48*100);
end
disp(solution,'solution = ');

// sort the solution set
[x O] = gsort(solution(:,3),'r','i');
for i = 1:k
  sortsolution(i,:) = solution(O(i),:);
end
```

```
disp(sortsolution,'sortsolution = ');

x = sortsolution(7,:); K = x(1); a = x(2)
    num = K * ((s + a)^2) ;
    den = s * s * (s^2 + 6*s + 5);
    G = syslin('c',num,num + den);
    y1 = csim('step',t1,G);

x = sortsolution(2,:); K = x(1); a = x(2)
    num = K * ((s + a)^2) ;
    den = s * s * (s^2 + 6*s + 5);
    G = syslin('c',num,num + den);
    y2 = csim('step',t1,G);
plot(t1,y1,t1,y2);
xgrid();
xtitle('Unit Step response curves','t (sec)','output
   ');
legend('K = 29 , a = 0.25', 'K = 27 , a = 0.2');
```

Scilab code Exa 8.a.13 Design of system with two degrees of freedom

```
// Example A-8-13
// Design of system with two degrees of freedom

clear; clc;
xdel(winsid()); //close all windows
mode(0);
// please edit the path
// cd "<path to dependencies";
// exec("plotresp.sci");

s = %s;
Gp = 100 /(s*(s + 1))
```

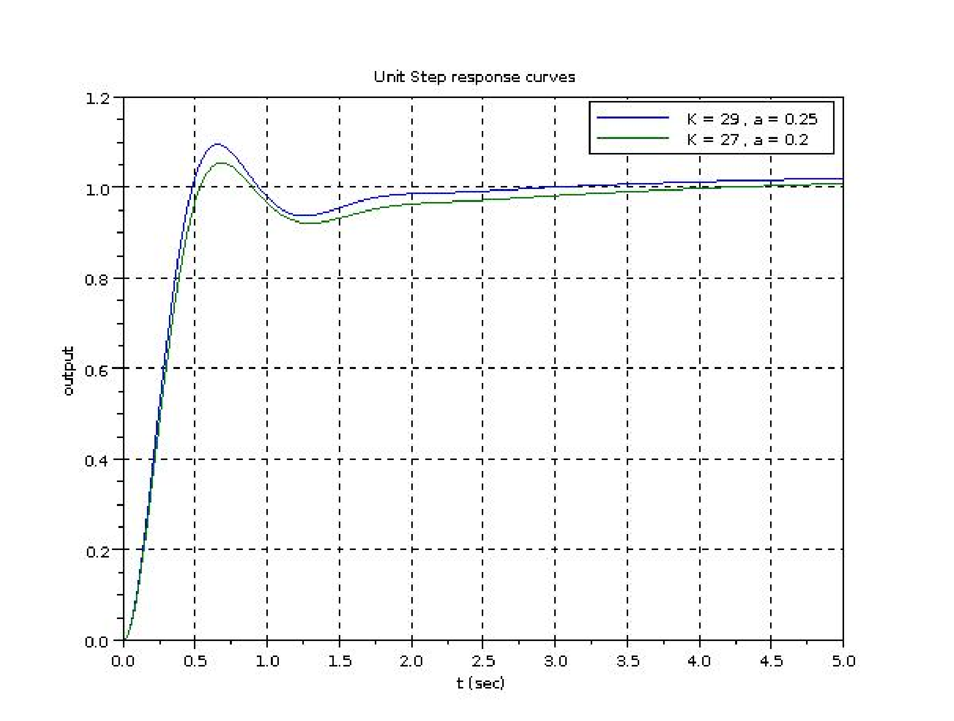

Figure 8.9: Computing optimal solution

```
dp = -5 + %i*5;

// Step 1: Design of Gc1 using root locus approach
angdef = 180 - phasemag(horner(Gp/s,dp))
angdef2 = angdef /2;
disp(angdef2,'each pole must contribute an angle of'
    );

a = 5 + 5*cotd(angdef2)
Gcx = (s + a)^2 / s;
K = 1/ abs(horner(Gcx*Gp, dp) )
Gc1 = K * (s + a)^2 / s

// determining Kp. Ti and Td
cf = coeff(numer(Gc1));
Kp = cf(2)
Ti = Kp / cf(1)
Td = cf(3) / Kp

t = 0:0.01:4;
u = ones(1,length(t));
subplot(2,1,1);
YbyD  = syslin('c',Gp / (1 + Gp * Gc1))
plotresp(u,t,YbyD,'Response to step disturbance
    input');
ax = gca();
ax.data_bounds = [0 0; 3 2];

//Step 2: Design of Gc
Gc = (YbyD.den - s^3) / 100 /  s

YbyR = syslin('c',1 - s^3 / YbyD.den )
subplot(2,1,2);
t = 0:0.01:3;
u = ones(1,length(t));
plotresp(u,t,YbyR,'Response to step reference input'
    );
scf();
```

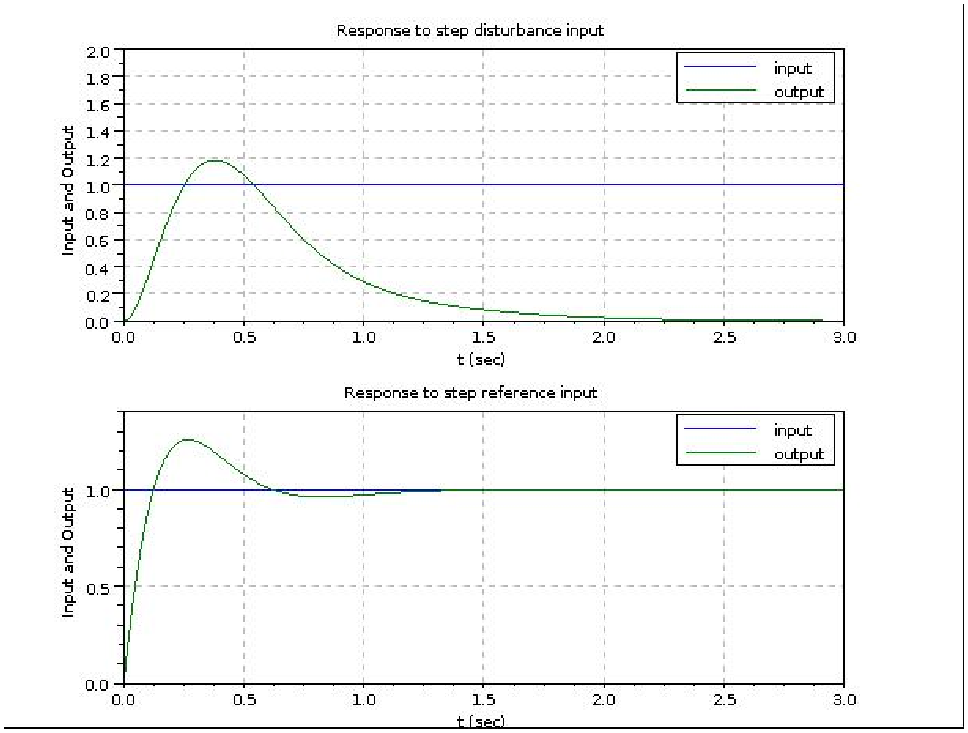

Figure 8.10: Design of system with two degrees of freedom

```
subplot(2,1,1);
plotresp(t,t,YbyR,'Response to ramp reference input'
    );
subplot(2,1,2);
t = 0:0.01:2;
u = 1/2 * t.^2;
plotresp(u,t,YbyR,'Response to acceleration
    reference input');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

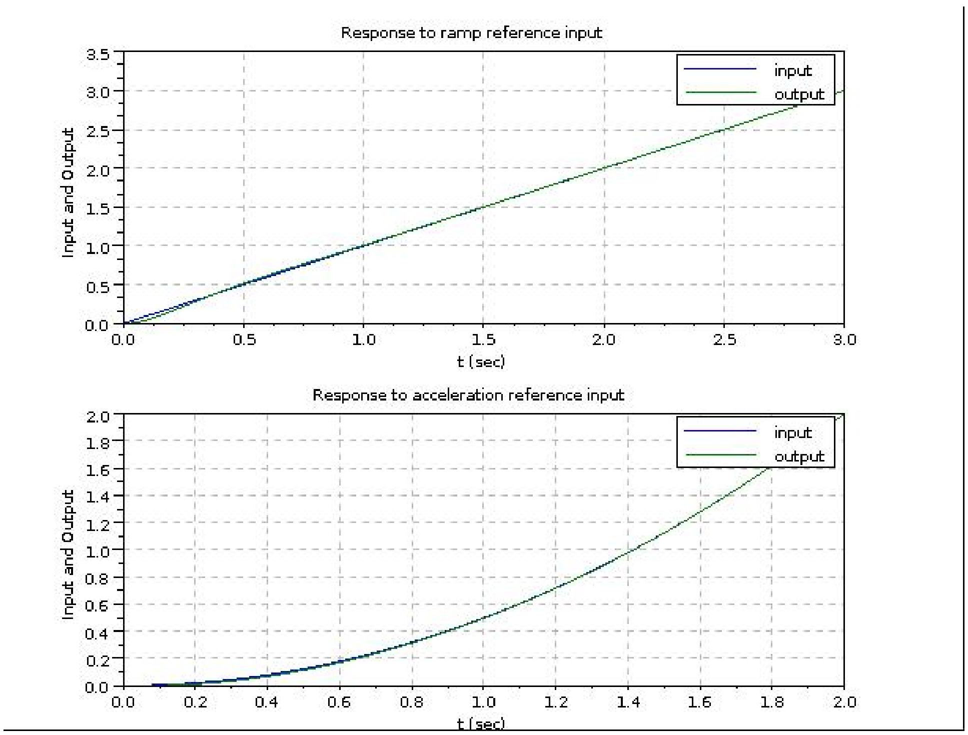

Figure 8.11: Design of system with two degrees of freedom

check Appendix AP 2 for dependency:

`plotresp.sci`

check Appendix AP 7 for dependency:

`rootl.sci`

Scilab code Exa 8.1 Tuning a PID controller using Nichols Second Rule

```
// Example 8-1
// Tuning a PID controller using Nichols Second Rule
clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd <your code directory>
// exec("plotresp.sci");
// exec("rootl.sci");

s = %s;
G = 1 / ( s * (s + 1) * (s + 5) )

// finding Kcr and wcr (omega cr)
w = poly(0,'w');
D = horner(denom(G),%i * w);
x = roots(imag(D));
wcr = abs(x(2))  // the non zero root
Kcr = -1*clean(horner(D,wcr))
Pcr = 2*%pi / wcr

Kp = 0.6 * Kcr
Ti = 0.5*Pcr
Td = 0.125*Pcr
Gc = Kp * ( s + 1/Ti + s^2*Td ) / s
```

```
GGc = syslin('c',G*Gc);
H = syslin('c',GGc /. 1);
disp(H,'closed loop system =');

rootl(GGc,0,'Root locus of open loop system');
sgrid([0.3],[]);
a = gca(); a.data_bounds = [-7 -4; 2 4];
xstring(-1,1,'zeta = 0.3');

scf();
t = 0:0.1:14;
u = ones(1,length(t));
plotresp(u,t,H,'');
// unacceptably large maximum overshoot

// new system
Kp2 = 39.42
Ti2 = 3.077
Td2 = 0.7692
Gc2 = Kp2 * ( s + 1/Ti2 + s^2*Td2 ) / s
GGc2 = syslin('c',G*Gc2);
H2 = syslin('c',GGc2 /. 1);
disp(H2,'closed loop system2 =');
plotresp(u,t,H2,'Step Response to a PID controlled system');
xstring(1.5,1.65,'System 1');
xstring(0.5,1.3,'System 2');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

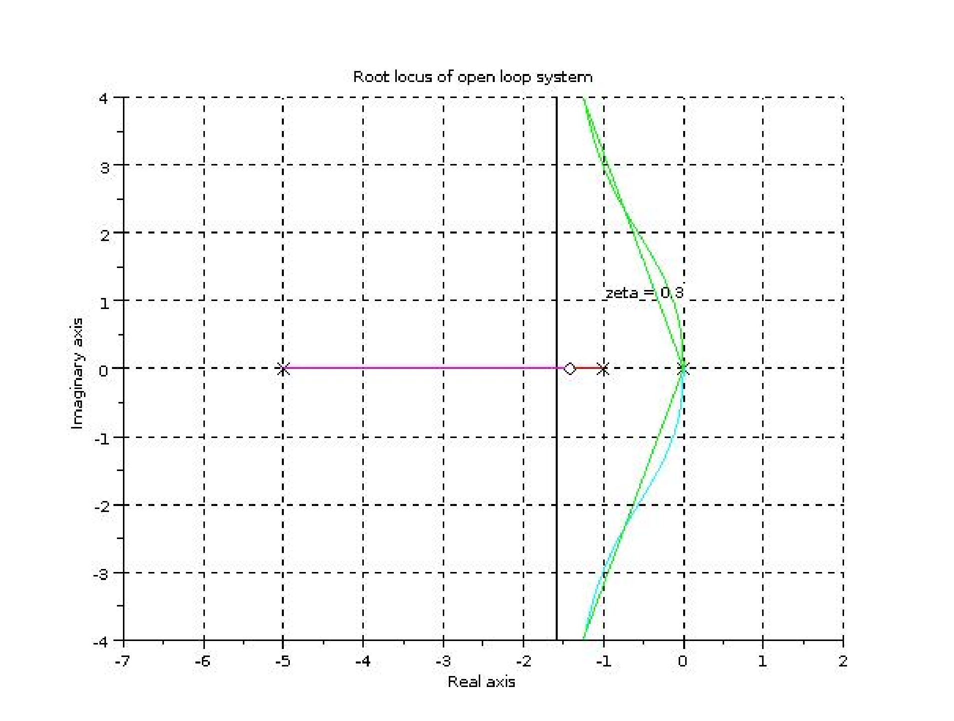

Figure 8.12: Tuning a PID controller using Nichols Second Rule

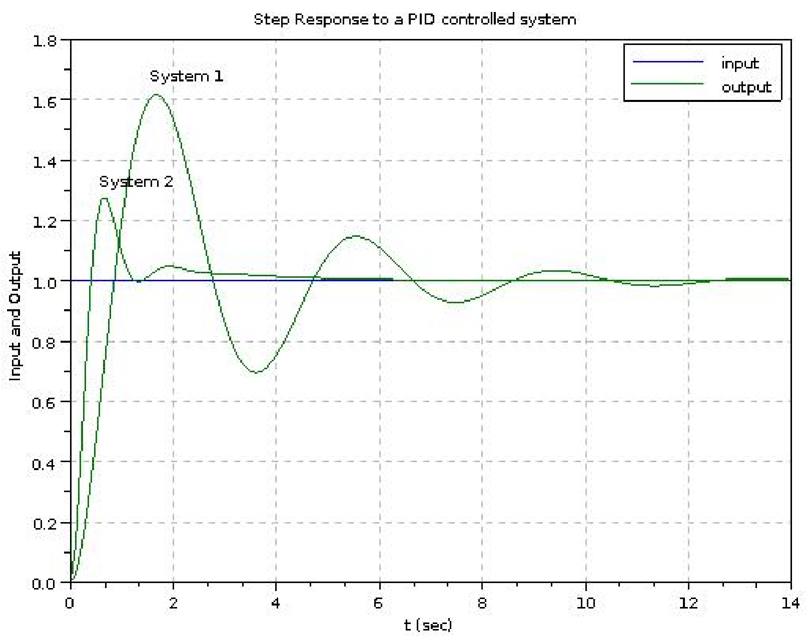

Figure 8.13: Tuning a PID controller using Nichols Second Rule

Scilab code Exa 8.2 `Computation of Optimal solution 1`

```
// Example 8-2
// Computation of Optimal solution 1

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "";
// exec("plotresp.sci");

s = %s;
G = 1.2 / ( 0.36*s^3+ 1.86*s^2 + 2.5*s + 1);

K = 2.0 : 0.2 : 3.0;
a = 0.5 : 0.2 : 1.5;

t = 0:0.1:5; u = ones(1,length(t));
// lesser points for a rough check
t1 = 0:0.01:5; u1 = ones(1,length(t1));
// more points for a rigorous check

k = 0;
for i = 1:6
  for j = 1:6
    Gc = K(i) * (s + a(j))^2 / s;
    H = G * Gc;
    H = syslin('c', H /. 1);
    y = csim(u,t,H);
    m = max(y);
    if m < 1.1 then
      y = csim(u1,t1,H);
      m = max(y);
      if m  < 1.1 then
        k = k + 1;
        solution(k,:) = [K(i) a(j) m];
      end
```

```
    end
  end
end
disp(solution,'solution [K a m] = ');
// to sort the matrix
[x O] = gsort(solution(:,3),'r','i');
// re order the matrix
for i = 1:k
  sortsolution(i,:) = solution( O(i) , :);
end
disp(sortsolution,'sortsolution [K a m] = ');

// Response with largest overshoot above 10%
x = sortsolution(k,:);
K = x(1); a = x(2);
Gc = K * (s + a)^2 / s;
H = G * Gc;
H = syslin('c', H /. 1);
plotresp(u1,t1,H,'Step Response with 10% overshoot')
   ;
disp(Gc,'Gc = ');
disp(H,'H = ');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 8.3 `Computation of Optimal solution 2`

```
// Example 8-3
// Computation of Optimal solution 2

clear; clc;
xdel(winsid()); //close all windows
```

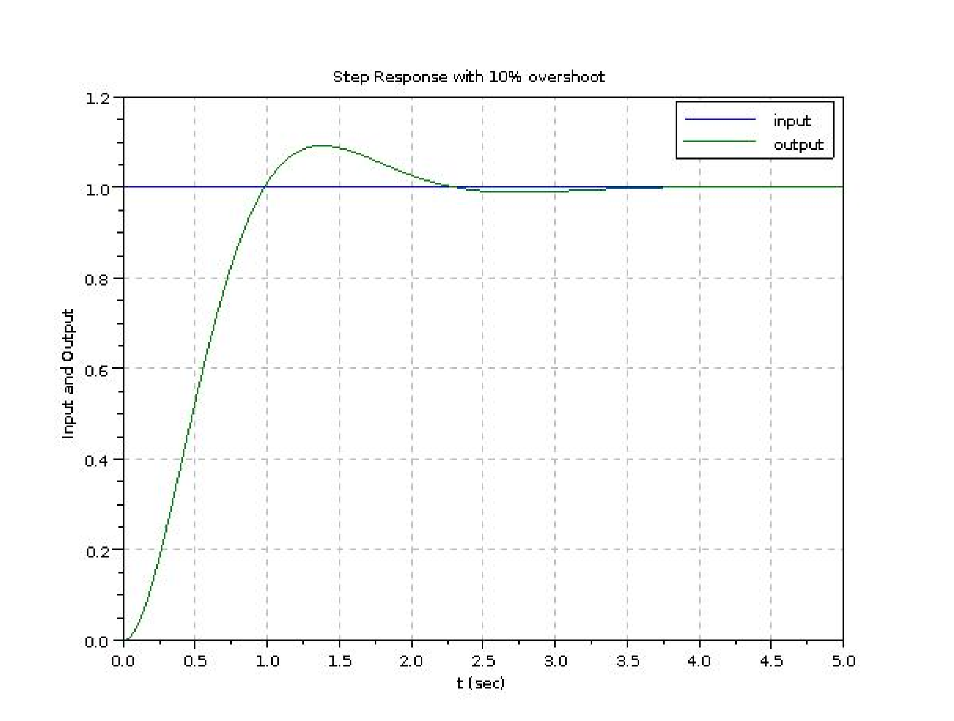

Figure 8.14: Computation of Optimal solution 1

```

// please edit the path
// cd "";
// exec("plotresp.sci");

s = %s;
G = 4 / ( s^3+ 6*s^2 + 8*s + 4);

t = 0:0.1:8; u = ones(1,length(t));
// lesser points for a rough check
t1 = 0:0.01:8; u1 = ones(1,length(t1));
// more points for a rigorous check

k = 0;
mprintf('Processing...\n');

for K = 3:0.2:6
  for a = 0.1:0.1:3
    Gc = K * (s + a)^2 / s;
    H = G * Gc;
    H = syslin('c', H /. 1);
    y = csim(u,t,H);
    m = max(y);
    if m < 1.15 & m > 1.08 then
       // give  a margin of 0.02 for the rough check - 1.08
       y = csim(u1,t1,H);
       m = max(y);
       if m < 1.15 & m > 1.10 then
         // check for settling time
         l =length(t1);
         while y(l) > 0.98 & y(l) < 1.02 ; l = l-1; end
            ts = (l-1) * 0.01;
         if ts < 3.00 then
            k = k + 1;
            solution(k,:) = [K a m ts];
          end
```

```
          end
        end

      end
    if modulo(K*10,2) == 0 then mprintf(' completed
       %d%%\n', (K - 3)/3*100)
   end
  end

disp(solution,'solution [K a m ts] = ');

[x O] = gsort(solution(:,3),'r','i');
for i = 1:k
  sortsolution(i,:) = solution( O(i) , :);
end
disp(sortsolution,'sortsolution [K a m ts] = ');

// Response with smallest overshoot
x = sortsolution(1,:);
K = x(1); a = x(2);
Gc = K * (s + a)^2 / s;
H = G * Gc;
H = syslin('c', H /. 1);
plotresp(u,t,H,'Step Response with smallest
    overshoot');
disp(Gc,'Gc = ');
disp(H,'H = ');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 8.4 `Design of system with two degrees of freedom`

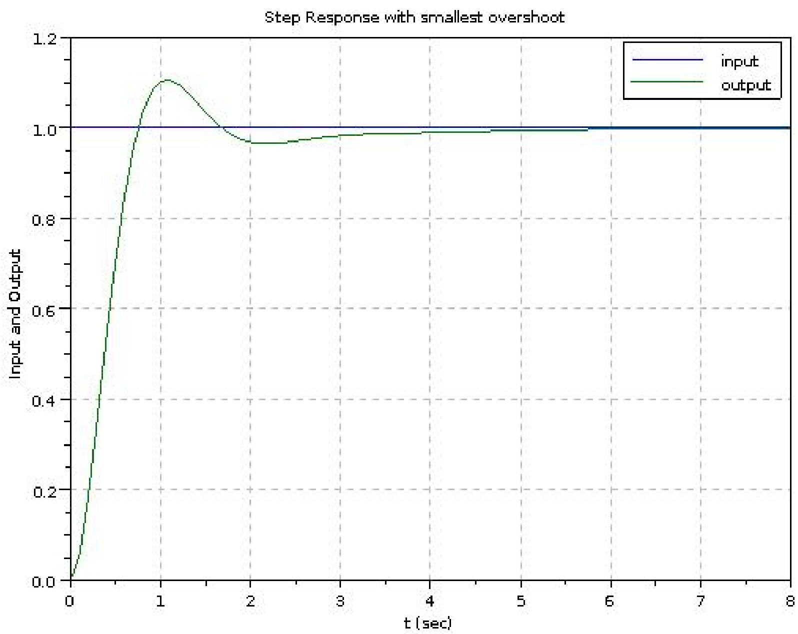

Figure 8.15: Computation of Optimal solution 2

```
// Example 8-4
// Design of system with two degrees of freedom

clear; clc;
xdel(winsid()); //close all windows
mode(0);
// please edit the path
// cd "";
// exec("plotresp.sci");

s = %s;
// Design Step 1: choosing a, b and c.

t = 0:0.1:4;
u = ones(1,length(t));

t1 = 0:0.01:4;
N = length(t1);
u1 = ones(1,N);
// N = N - 3

k = 0;
mprintf('Processing...\n');

for i = 1:21
  a = 6.2 - 0.2*i;
  for j = 1:21
    b = 6.2 - 0.2*j;
    for h = 1:21
      c = 12.2 - 0.2*h;
      num = (2*a + c)*s^2 + (a*a + b*b + 2*a*c)*s +
         (a*a + b*b)*c;
      den = s^3 + num;
      G = syslin('c',num,den);
      y = csim(u,t,G);
      m = max(y);
      if m < 1.19 & m > 1.00 then
        y = csim(u1,t1,G);
```

```
        m = max(y);
        if m < 1.19 & m > 1.02 then
          l = N;
          while y(l) > 0.98 & y(l) < 1.02 ; l = l-1; end
            ts = (l-1) * 0.01;
          if ts < 1.0 then
             k = k + 1;
             solution(k,:) = [a b c m ts];
           end
         end
       end

     end
  end
 mprintf('completed %d%%\n', (6 - a)/4*100);
end

disp(solution,'solution = ');

K = solution(1,:);
a = K(1); b = K(2); c = K(3);
num = (2*a + c)*s^2 + (a*a + b*b + 2*a*c)*s + (a*a + b*b)*c;
den = s^3 + num;
YbyR = syslin('c',num,den);disp(YbyR,'Y(s)/R(s) =');
subplot(2,1,1);
plotresp(u1,t1,YbyR,'Step response for a = 4.2 ,b = 2 ,c =12');

cf  = coeff(den);
K = (cf(3) - 1) / 10
alpha_plus_beta = cf(2) / K /10
alphabeta = cf(1) / K / 10
Gc = K * (s^2 + alpha_plus_beta*s + alphabeta) / s
YbyD = syslin('c',10*s,den);
disp(YbyD,'Y(s)/D(s) = ');
subplot(2,1,2);
```

```
plotresp(u1,t1,YbyD,'Response to step disturbance
    input for a = 4.2 ,b = 2 ,c =12');
a = gca(); a.data_bounds = [0 -0.01; 4 0.07];

// Design Step 2
scf();
Gc1 = (YbyR.num / 10) / s
Gc2 = Gc - Gc1

// response to reference inputs
y1 = csim(t,t,YbyR); u = 1/2 * t.^2;
y2 = csim(u,t,YbyR);

subplot(2,1,1);
plotresp(t,t,YbyR,'Response to unit ramp input');
subplot(2,1,2);
plotresp(u,t,YbyR,'Response to unit acceleration
    input');
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 8.5 `Design of system with two degrees of freedom 2`

```
// Example 8-5
// Design of system with two degrees of freedom 2

clear; clc;
xdel(winsid()); //close all windows
mode(0);
// please edit the path
```

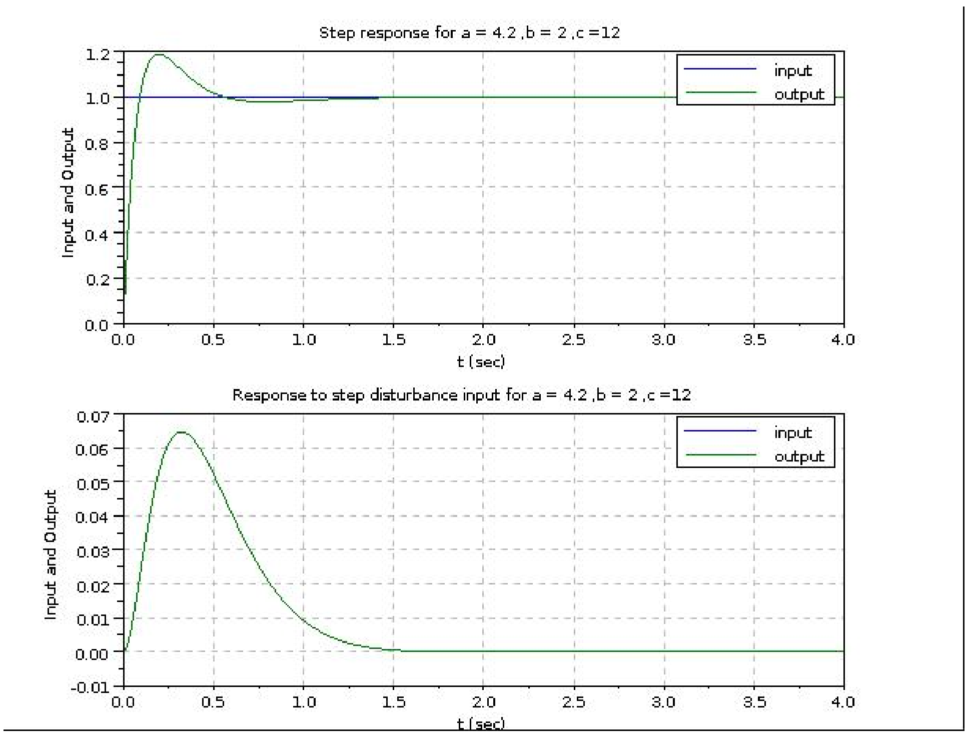

Figure 8.16: Design of system with two degrees of freedom

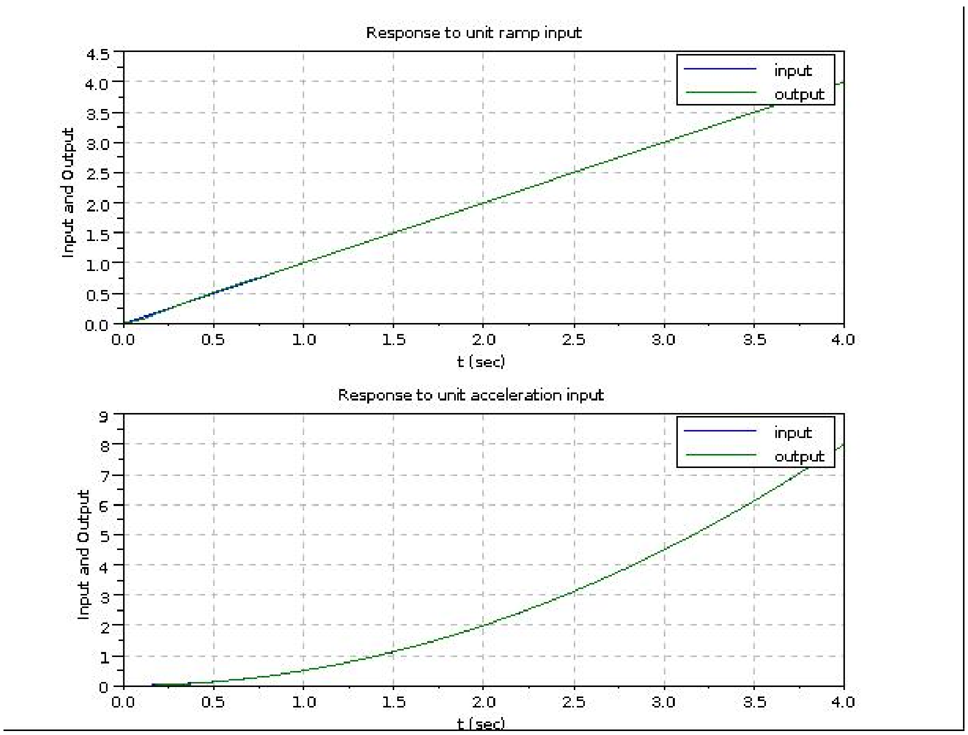

Figure 8.17: Design of system with two degrees of freedom

```
// cd "";
// exec("plotresp.sci");

s = %s;
Gp = 5 / (s+1) / (s+5)
t = 0:0.01:3;
u = ones(1,length(t));

// Step 1: Design of Gc1
a = sqrt(13)
K = 4 / (5*a - 15)
Gc1 = K * (s + a)^2 / s

// determining Kp. Ti and Td
cf = coeff(numer(Gc1));
Kp = cf(2)
Ti = Kp / cf(1)
Td = cf(3) / Kp

subplot(2,1,1);
YbyD  = syslin('c',Gp / (1 + Gp * Gc1))
plotresp(u,t,YbyD,'Response to step disturbance
   input');
a = gca();
a.data_bounds = [0 0; 3 0.1];

//Step 2: Design of Gc2
cf = coeff(YbyD.den);
Kp2 = (cf(2) - 47.63 ) / 5
Td2 = (cf(3) - 6.6051) / 5 / Kp2

Gc2 = Kp2 * ( 1 + Td2*s)

YbyR = syslin('c',1 - s^3 / YbyD.den )
subplot(2,1,2);
t = 0:0.05:2;
u = ones(1,length(t));
plotresp(u,t,YbyR,'Response to step reference input'
```

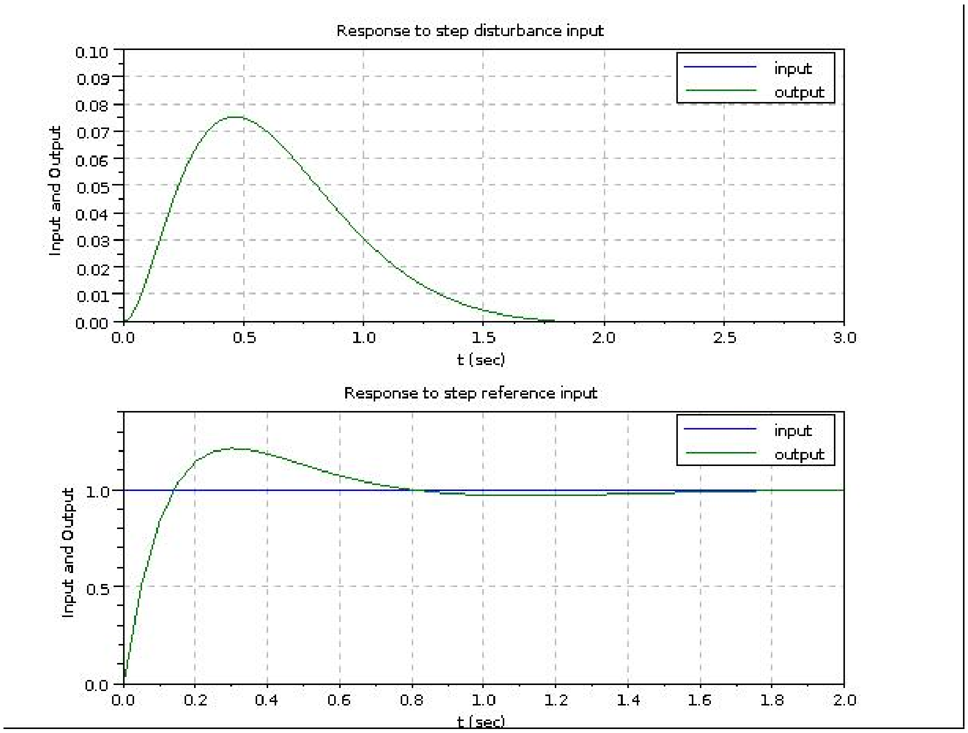

Figure 8.18: Design of system with two degrees of freedom 2

```
    );
scf();
subplot(2,1,1);
plotresp(t,t,YbyR,'Response to ramp reference input'
    );
subplot(2,1,2);
u = 1/2 * t.^2;
plotresp(u,t,YbyR,'Response to acceleration
    reference input');
```

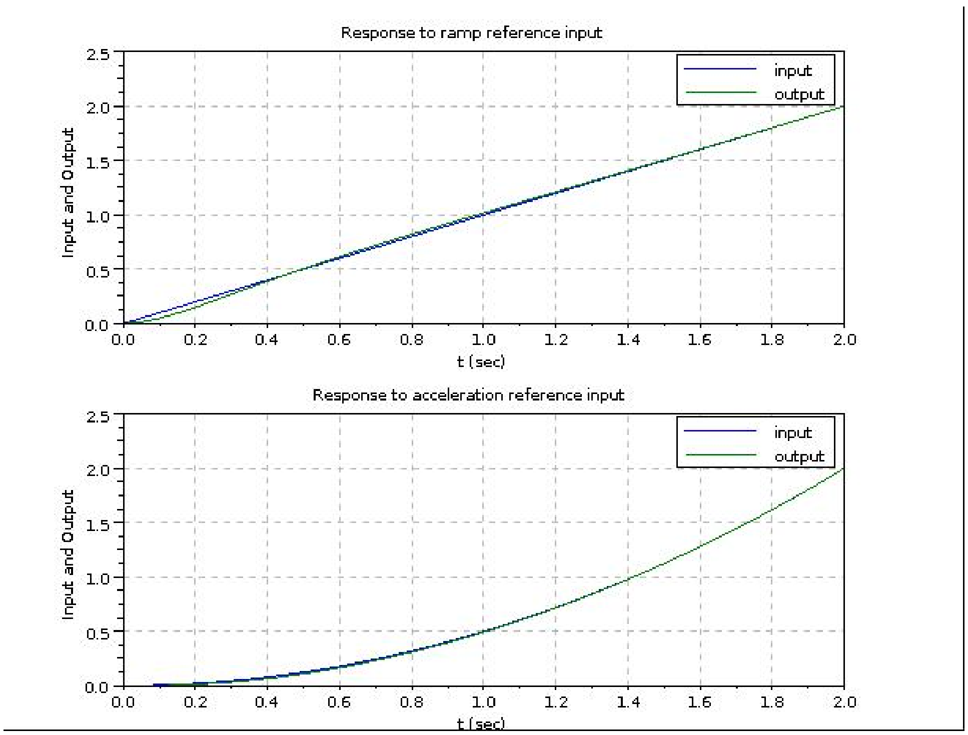

Figure 8.19: Design of system with two degrees of freedom 2

Chapter 9

Control Systems Analysis in State Space

Scilab code Exa 9.b.3 `Obtaining canonical form`

```
// Exercise B-9-3
// Obtaining canonical form

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "<path to dependencies>";
// exec("transferf.sci");

A = [1 2; -4 -3];
B = [1;2];
C = [1 1];
D = 0;

[Ac Bc U ind] = canon(A,B);
U = -1*U;    // a correction
Cc = C*U;
disp(clean(Ac),'Ac = ');
```

```
disp(clean(Bc),'Bc = ');
disp(clean(Cc),'Cc = ');
disp(U,'transformation matrix U = ');
// Ac=inv(U)*A*U, Bc=inv(U)*B

// check
Htf1 = transferf(A,B,C,D);
Htf2 = transferf(Ac,Bc,Cc,D);
disp(Htf1,'Htf1 = ');
disp(Htf2,'Htf2 = ');
```

check Appendix AP 4 for dependency:

`transferf.sci`

Scilab code Exa 9.a.5 `Conversion from transfer function model to state space model`

```
// Example A-9-5
// Conversion from transfer function model to state
   space model

clear; clc;
xdel(winsid()); //close all windows

s = %s;
num = 25.04*s + 5.008;
den = poly( [5.008 25.1026 5.03247 1],'s','c');

Hss = cont_frm(num,den);
disp(Hss,'Hss = ');
```

Scilab code Exa 9.a.16 `Controllability and pole zero cancellation`

```
// Example A-9-16
// Controllability and pole zero cancellation

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "<path to dependencies>";
// exec("transferf.sci");


A = [-3 1; -2  1.5];
B = [1; 4];
C = [1 0];
D = 0;
Cc = cont_mat(A,B); disp(Cc,'state controllability
    matrix =');
disp(det(Cc), 'det(Cc) = ');

Htf = transferf(A,B,C,D); disp(Htf,'Reduced transfer
     function =');
e = spec(A);                disp(e,'Eigen values = ');
D = poly(e,'s');    disp(D,'actual denominator (
    characteristic poly) =');
```

check Appendix AP 4 for dependency:

`transferf.sci`

Scilab code Exa 9.a.17 `Controllability observability and pole zero cancellation`

```
// Example A-9-17
// Controllability observability and pole zero
    cancellation

clear; clc;
```

```
xdel(winsid());   //close all windows


A = [0  1; -0.4  -1.3];
B = [0; 1];
C = [0.8 1];
D = [0];
G1 = syslin('c',A,B,C,D); ssprint(G1);

G2 = syslin('c',A', C',B',D); ssprint(G2);

Cc1 = cont_mat(A,B); disp(Cc1,'state controllability matrix 1 =');
disp(det(Cc1), 'det(Cc1) = ');
Ob1 = obsv_mat(A,C); disp(Ob1,'observability matrix 1 =');
disp(det(Ob1),'det(Ob1)');

Cc2 = cont_mat(A',C'); disp(Cc2,'state controllability matrix 2 =');
disp(det(Cc2), 'det(Cc2) = ');
Ob2 = obsv_mat(A',B'); disp(Ob2 ,'observability matrix 2 =');
disp(det(Ob2),'det(Ob1)');

Htf = ss2tf(G1); disp(Htf,'Reduced transfer function =');
e = spec(A);                  disp(e,'Eigen values = ');
D = poly(e,'s');   disp(D,'actual denominator (characteristic poly) =');
```

check Appendix AP 6 for dependency:

`pf_residu.sci`

Scilab code Exa 9.1 Transfer function to controllable observable and jordon canoni

```
// Example 9-1
// Transfer function to controllable, observable and jordon canonical forms

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "<path for the dependencies>";
// exec("pf_residu.sci");

s = %s;
N = s + 3;
D = s^2 + 3*s + 2;

Hc = cont_frm(N,D);
disp('controllable form ='); ssprint(Hc);

Ho =syslin('c', (Hc.A)' , (Hc.C)', (Hc.B)' ,Hc.D);
disp('observable form ='); ssprint(Ho);

A = diag(roots(D));
B = [1;1];
C = pf_residu(N,D)';
D = Hc.D;                    // in this case : b0 = 0
Hj = syslin('c',A,B,C,D);
disp('jordon canonical form =');ssprint(Hj);

// This example will work for any proper transfer function
// with all distinct poles or eigen values
```

Scilab code Exa 9.2 `Transformations in state space`

```
// Example 9-2
```

```
// Transformations in state space

clear; clc;
xdel(winsid()); //close all windows
mode(0);

A = [0 1 0; 0 0 1; -6 -11 -6];
B = [0; 0; 0];
C = [1 0 0];
D = [0];
H = syslin('c',A,B,C,D);
disp('non standard form =');ssprint(H);

e = spec(A)'   // eigen values
P = [ones(1,3); e; e.^2] // P is the transformation
    matrix
A1 = diag(e);
B1 = inv(P)* B;
C1 = C * P;
D1 = D;
H1 = syslin('c',A1,B1,C1,D1);
disp('standard form =');ssprint(H1);
```

check Appendix AP 4 for dependency:

`transferf.sci`

Scilab code Exa 9.3 `Conversion from state space to transfer function model`

```
// Example 9-3
// Conversion from state space to transfer function
    model

clear; clc;
xdel(winsid()); //close all windows

```

```
// please edit the path
// cd "<path to your dependencies>";
// exec("transferf.sci");

A = [0 1 0; 0 0 1; -5.008 -25.1026 -5.03247];
B = [0; 25.04; -121.005];
C = [1 0 0];
D = [0];

H = transferf(A,B,C,D);
disp(H,'H =');
```

check Appendix AP 4 for dependency:

`transferf.sci`

Scilab code Exa 9.4 `Conversion from state space to transfer function model`

```
// Example 9-4
// Conversion from state space to transfer function
   model

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "<path to your dependencies>";
// exec("transferf.sci");

A = [0 1; -25 -4];
B = [1 1; 0 1];
C = [1 0; 0 1];
D = [0 0; 0 0];

H = transferf(A,B,C,D);
disp(H,'H =');
```

```

// Htf is the tranfer function matrix with four
   transfer functions
// Htf(y1,u1),Htf(y1,u2)
// Htf(y2,u1),Htf(y2,u2)
```

check Appendix AP 5 for dependency:

`ilaplace.sci`

check Appendix AP 6 for dependency:

`pf_residu.sci`

Scilab code Exa 9.5 `State transition matrix`

```
// Example 9-5
// State transition matrix

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "<path for the dependencies>";
// exec("pf_residu.sci");
// exec("ilaplace.sci");

s = %s;
A = [0 1; -2 -3];
L = inv(s*eye(2,2) - A);
disp(L, 'inv(sI - A) =');

// Find the Inverse Laplace transform
for i = 1:2
  for j = 1:2
    phi(i,j) = ilaplace(L(i,j));
  end;
```

```
end;

disp(phi,'state transition matrix =');
// ilaplace may not work for systems with repeated
   poles
```

check Appendix AP 5 for dependency:

`ilaplace.sci`

check Appendix AP 6 for dependency:

`pf_residu.sci`

Scilab code Exa 9.7 Finding e to the power At using laplace transforms

```
// Example 9-7
// Finding e to the power At using laplace
   transforms

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "<path for the dependencies>";
// exec("pf_residu.sci");
// exec("ilaplace.sci");

s = %s;
A = [0 1; 0 -2];
L = inv(s*eye(2,2) - A);
disp(L, 'inv(sI - A) =');

// Find the Inverse Laplace transform
for i = 1:2
  for j = 1:2
    phi(i,j) = ilaplace(L(i,j));
```

```
  end;
end;
disp(phi,'e^At =');
```

Scilab code Exa 9.9 Linear dependence of vectors

```
// Example 9-9
// Linear dependence of vectors

clear; clc;
xdel(winsid()); //close all windows
mode(0)

x1 = [1; 2; 3]
x2 = [1; 0; 1]
x3 = [2; 2; 4]
A = [x1 x2 x3];
disp(A,'[x1:x2:x3] =');
disp(clean(det(A)),'det([x1:x2:x3]) ='); // singular

x3 = [2;2;2]
A = [x1 x2 x3];
disp(A,'[x1:x2:x3] =');
disp(det(A),'det([x1:x2:x3]) =');// non singular
```

Scilab code Exa 9.14 State and ouput controllability and observability

```
// Example 9-14
// State and ouput controllability and observability

clear; clc;
xdel(winsid()); //close all windows

```

```
A = [1 1; -2 -1];
B = [0;1];
C = [1 0];
D = [0];
G =syslin('c',A,B,C,D); ssprint(G);

Cc = cont_mat(A,B); disp(Cc,'state controllability
    matrix =');
c = [C*B C*A*B];  disp(c,'output controllability
    matrix =');
Ob = obsv_mat(A,C); disp(Ob,'observability matrix ='
    );
```

Scilab code Exa 9.15 Observability

```
// Example 9-15
// Observability

clear; clc;
xdel(winsid());   //close all windows

A = [0 1 0; 0 0 1; -6 -11 -6];
B = [0; 0; 1];
C = [4 5 1];

Ob = obsv_mat(A,C);
disp(Ob,'observability matrix =');
disp(clean(det(Ob)) , 'det(Ob) =');
// system is not completely observable
```

Chapter 10

Control Systems Design in State Space

Scilab code Exa 10.i.1 Designing a regulator using a minimum order observer

```
// Illustration 10.1
// Designing a regulator using a minimum order
   observer

// Section 10-6 of the book

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "<path to dependencies>";
// exec("minorder.sci");

function smallplot(i)
```

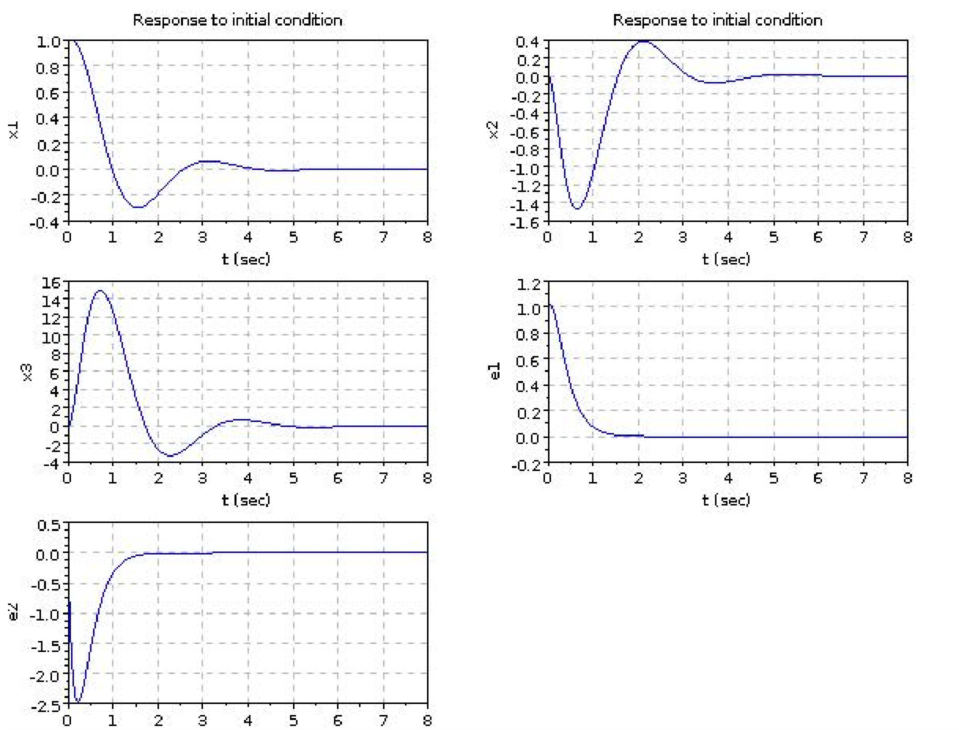

Figure 10.1: Designing a regulator using a minimum order observer

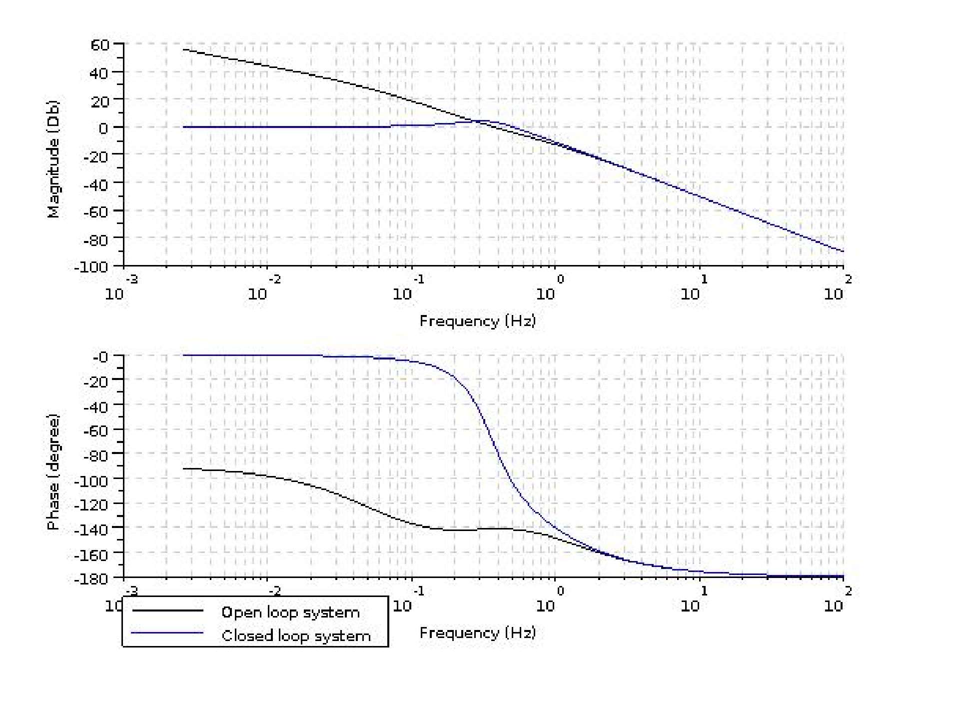

Figure 10.2: Designing a regulator using a minimum order observer

```
  subplot(3,2,i);xgrid(color('gray'));
  plot(t,x(i,:));
endfunction


A = [0 1 0; 0 0 1; 0 -24 -10];
B = [0; 10; -80];
C = [1 0 0];
D = [0];
Gp = syslin('c',A,B,C,D);

// Trial 1
disp('trial 1')
P = [-1 + %i*2,-1 - %i*2,-5 ]
Q = [-10 -10]  // observer poles

// Determining gains K and Ke
// Determining observer controller transfer function
[K Ke Go ch] = minorder(A,B,P,Q);
K
Ke
disp(Go,'observer controller transfer function =');
disp(ch,'overall system characteristic equation =');
disp(roots(Go.den),'observer controller has unstable
    root!');

disp('trial 2'); // Trial 2;
P
Q = [-4.5 -4.5]; // change Q
[K Ke Go ch AA] = minorder(A,B,P,Q);
K
Ke
disp(Go,'observer controller transfer function =');
disp(ch,'overall system characteristic equation =');
disp(roots(Go.den),'observer controller has all
   stable roots!');

// system response to initial conditions
```

```
x0 = [1; 0; 0; 1; 0];
G = syslin('c',AA,[1 ;0 ;0 ;0 ;0],[1 0 0 0 0],[0],x0);

t = 0:0.01:8;
u = zeros(1,length(t));
[y x] = csim(u,t,G);

smallplot(1);
xtitle('Response to initial condition','t (sec)','x1');
smallplot(2);
xtitle('Response to initial condition','t (sec)','x2');
smallplot(3);
xtitle('','t (sec)','x3');
smallplot(4);
xtitle('','t (sec)','e1');
smallplot(5);
xtitle('','t (sec)','e2');

scf();
// Bode diagram
O = Go*Gp; C = O /. 1;
bode([O;C],0.001,100,['Open loop system'; 'Closed loop system']);
disp(p_margin(O),'Phase margin');
```

check Appendix AP 1 for dependency:

`minorder.sci`

Scilab code Exa 10.i.2 Designing a control system with a minimum order observer

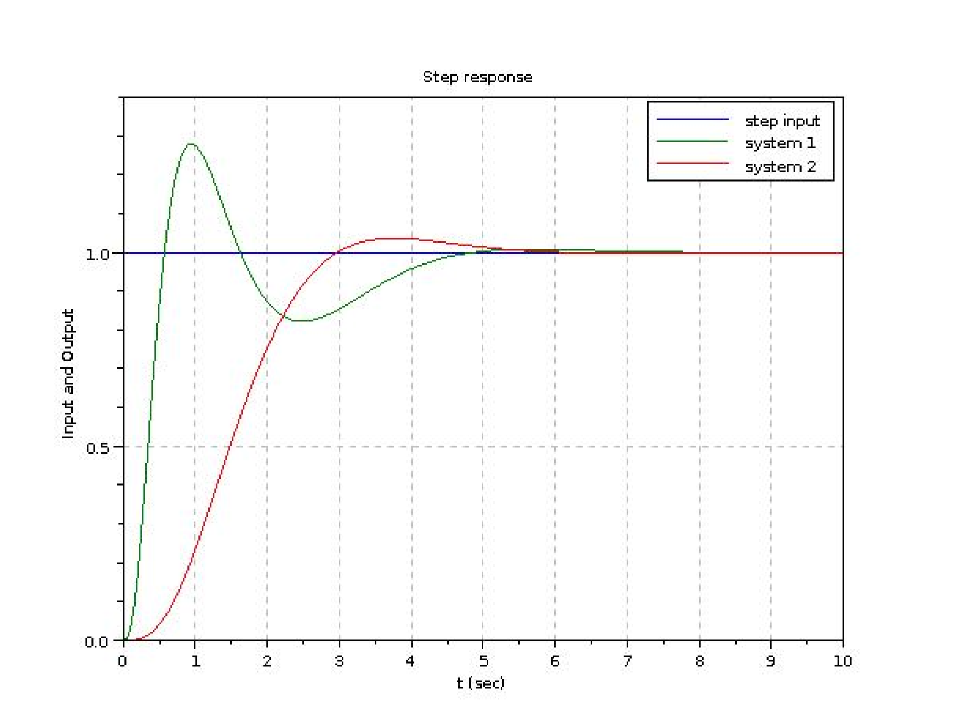

Figure 10.3: Designing a control system with a minimum order observer

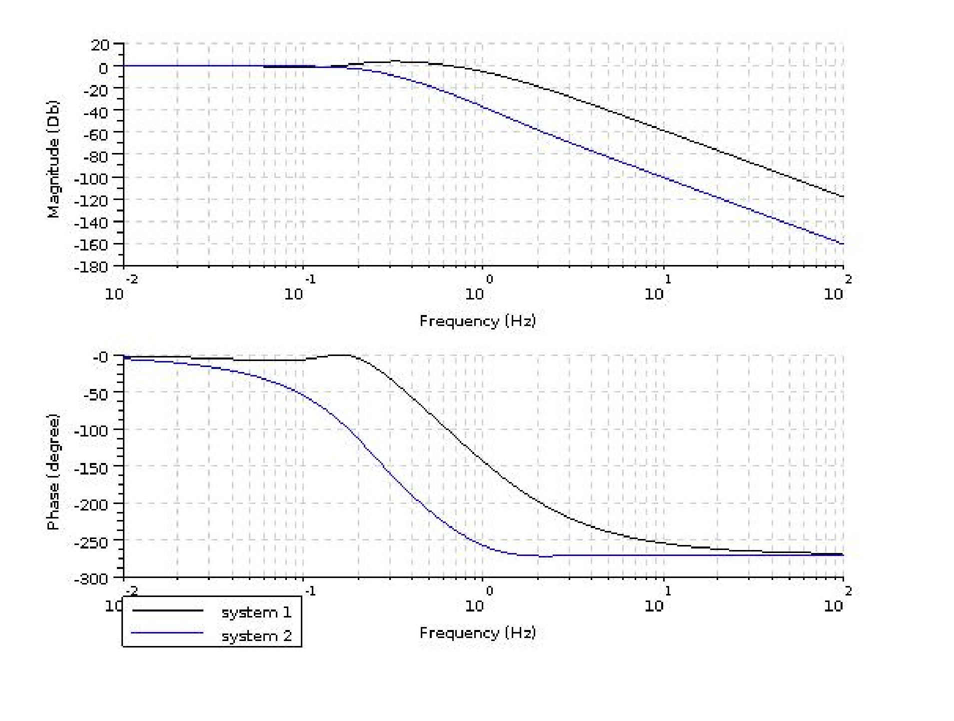

Figure 10.4: Designing a control system with a minimum order observer

```
// Illustration 10.2
// Designing a control system with a minimum order
   observer

// Section 10-7 of the book

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "<path to dependencies>";
// exec("minorder.sci");
// exec("plotresp.sci");

s = %s;
t = 0:0.05:10;
u = ones(1,length(t));
Gp = syslin('c',1,s*(s^2 + 1));
Gs = cont_frm(1,s*(s^2 + 1));
A = Gs.A;
B = Gs.B;
C = Gs.C;
D = Gs.D;

// designing the observer controller
P = [-1 + %i,-1 - %i,-8 ]
Q = [-4 -4]  // observer poles
[K Ke Go] = minorder(A,B,P,Q);
K
Ke
disp(Go,'observer controller transfer function =');

// First configuration
C1 = Go*Gp /. 1;
disp(C1,'closed loop system of first configuration =
   ');
plotresp(u,t,C1,'Step response');
```

```

// Secoond Configuration
C = Gp /. Go;
N = 1 / horner(C,0)
C2 = syslin('c',N*C);
y = csim(u,t,C2);
disp(C2,'closed loop system of second configuration
   =');
plot(t,y,'r');
legend('step input','system 1','system 2');

// Bode diagram
scf();
bode([C1;C2],0.01,100,['system 1';'system 2']);
// frequency in Hz
```

check Appendix AP 1 for dependency:

`minorder.sci`

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 10.a.5 `Feedback gain for moving eigen values`

```
// Example A-10-5
// Feedback gain for moving eigen values

clear; clc;
xdel(winsid());  //close all windows
mode(0);

s = %s;
A = [0 1;-2 -3];
B = [0; 2];
C = [1 0];
```

```
E = [-3 -5]; // new eigen values

ch = det(s*eye(2,2) - A)
cf = coeff(ch);
a = cf(1: $-1)

chd = poly(E,'s');
cf2 =  coeff(chd);
alpha = cf2(1: $-1)

M = cont_mat(A,B)
W = [cf(2:$) ; 1 0]
T = M*W

Ti = inv(T); disp(Ti,'inv(T)');
K = (alpha - a) * Ti
```

Scilab code Exa 10.a.6 Gain matrix determination

```
// Example A-10-6
// Gain matrix determination

clear; clc;
xdel(winsid());  //close all windows


A = [0 1 0; 0 0 1;-6 -11 -6];
B = [0; 0; 10];

P = [-2 + %i*2*sqrt(3) , -2 - %i*2*sqrt(3), -10];
K = ppol(A,B,P);   disp(K,'K =');
```

Scilab code Exa 10.a.9 Transforming to canonical form

```
// Example A-10-9
// Transforming to canonical form

clear; clc;
xdel(winsid()); //close all windows
mode(0);

s = %s;
A = [1 1;-4 -3];
B = [0; 2];
C = [1 1];

ch = det(s*eye(2,2) - A)
cf = coeff(ch);
a = cf(1: $-1)


N = obsv_mat(A,C)'
W = [cf(2:$) ; 1 0]
Qi = W*N'
Q = inv(Qi)

A1 = Qi*A*Q
B1 = Qi*B
```

Scilab code Exa 10.a.13 Designing a regulator using a minimum order observer

```
// Example A-10-13
// Designing a regulator using a minimum order
   observer

clear; clc;
xdel(winsid()); //close all windows
mode(0);

```

```
function smallplot(i)
  subplot(3,2,i);xgrid(color('gray'));
  plot(t,x(i,:));
endfunction


A = [0 0 1 0; 0 0 0 1; -36 36 -0.6 0.6; 18 -18 0.3
    -0.3];
B = [0; 0; 1; 0];
C = [1 0 0 0; 0 1 0 0];
D = [0;0];
Gp = syslin('c',A,B,C,D);

Aab = A(1:2,3:$);
Abb = A(3:$,3:$);

P = [-2 + %i*2*sqrt(3),-2 - %i*2*sqrt(3),-10,-10 ]
Q = [-15 -16]  // observer poles

K = ppol(A,B,P)
Ke = ppol(Abb',Aab',Q)'
Kb = K(3:$);

AA = [A - B*K , B*Kb; zeros(2,4) , Abb - Ke*Aab]

// system response to initial conditions
x0 = [0.1; 0; 0; 0; 0.1; 0.05];
G = syslin('c',AA,zeros(6,1),zeros(1,6),[0],x0);

t = 0:0.01:4;
u = zeros(1,length(t));
[y x] = csim(u,t,G);

smallplot(1);
xtitle('Response to initial condition','t (sec)','x1
    ');
smallplot(2);
xtitle('Response to initial condition','t (sec)','x2
```

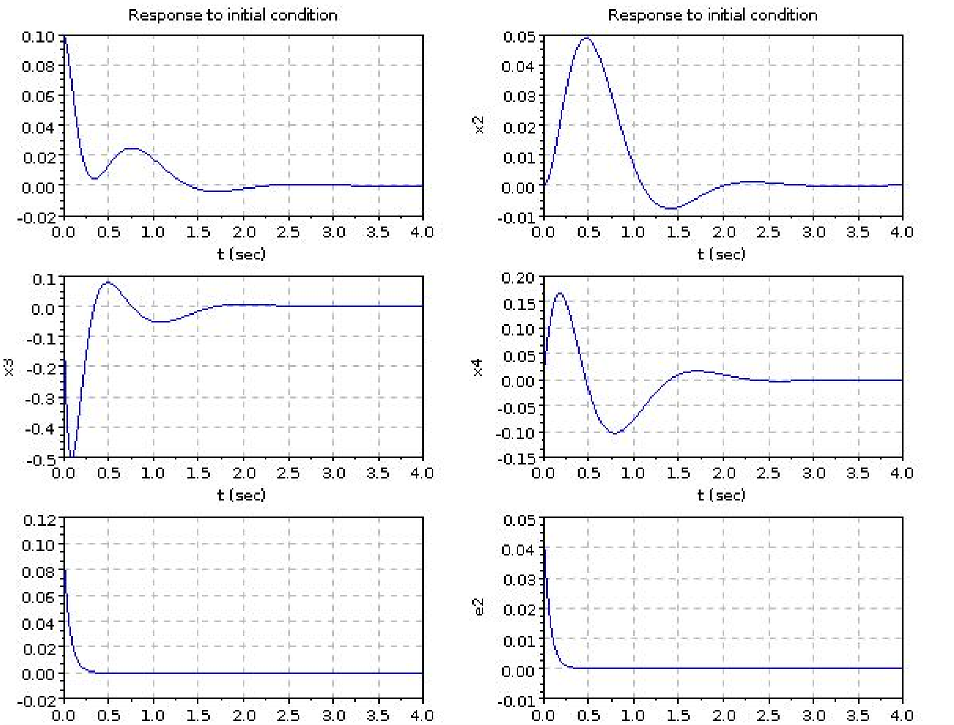

Figure 10.5: Designing a regulator using a minimum order observer

```
    ');
smallplot(3);
xtitle('','t (sec)','x3');
smallplot(4);
xtitle('','t (sec)','x4');
smallplot(5);
xtitle('','t (sec)','e1');
smallplot(6);
xtitle('','t (sec)','e2');
```

Scilab code Exa 10.a.14 `Designing a regulator using a minimum and full order obser`

```
// Example A-10-14
// Designing a regulator using a minimum and full order observer

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "<path to dependencies>";
// exec("minorder.sci");

function smallplot(i)
  subplot(2,2,i);xgrid(color('gray'));
  plot(t,x(i,:));
endfunction

A = [0 1; 0 -2];
B = [0; 4];
C = [1 0];
D = [0];
Gp = syslin('c',A,B,C,D);

P = [-2 + %i*2*sqrt(3),-2 - %i*2*sqrt(3)]
Q1 = [-8 -8 ]
Q2 = [-8];

disp('full order obssrver -');
K1 = ppol(A,B,P)
Ke1 = ppol(A',C',Q1)'

Go1 =transferf(A-B*K1-Ke1*C,Ke1,K1,[0]);
disp(Go1,'full order observer controller transfer function =');

// system response to initial conditions
AA1 = [A - B*K1, B*K1; zeros(2,2), A - Ke1*C];
x0 = [1; 0; 1; 0];
```

```
G = syslin('c',AA1,zeros(4,1),zeros(1,4),[0],x0);

t = 0:0.05:8;
u = zeros(1,length(t));
[y x] = csim(u,t,G);
smallplot(1);
xtitle('Response to initial condition (Full order)',
    't (sec)','x1');
smallplot(2);
xtitle('Response to initial condition (Full order)',
    't (sec)','x2');
smallplot(3);
xtitle('','t (sec)','e1');
smallplot(4);
xtitle('','t (sec)','e2');

disp('minimal order observer -');
P
Q2
[K2 Ke2 Go2 ch AA2] = minorder(A,B,P,Q2);
K2
Ke2
disp(Go2,'minimal order observer controller transfer
     function =');

x0 = [1; 0; 1;];
G = syslin('c',AA2,zeros(3,1),zeros(1,3),[0],x0);

t = 0:0.05:8;
u = zeros(1,length(t));
[y x] = csim(u,t,G);
scf();
smallplot(1);
xtitle('Response to initial condition (minimal order
    )','t (sec)','x1');
smallplot(2);
xtitle('Response to initial condition (minimal order
    )','t (sec)','x2');
```

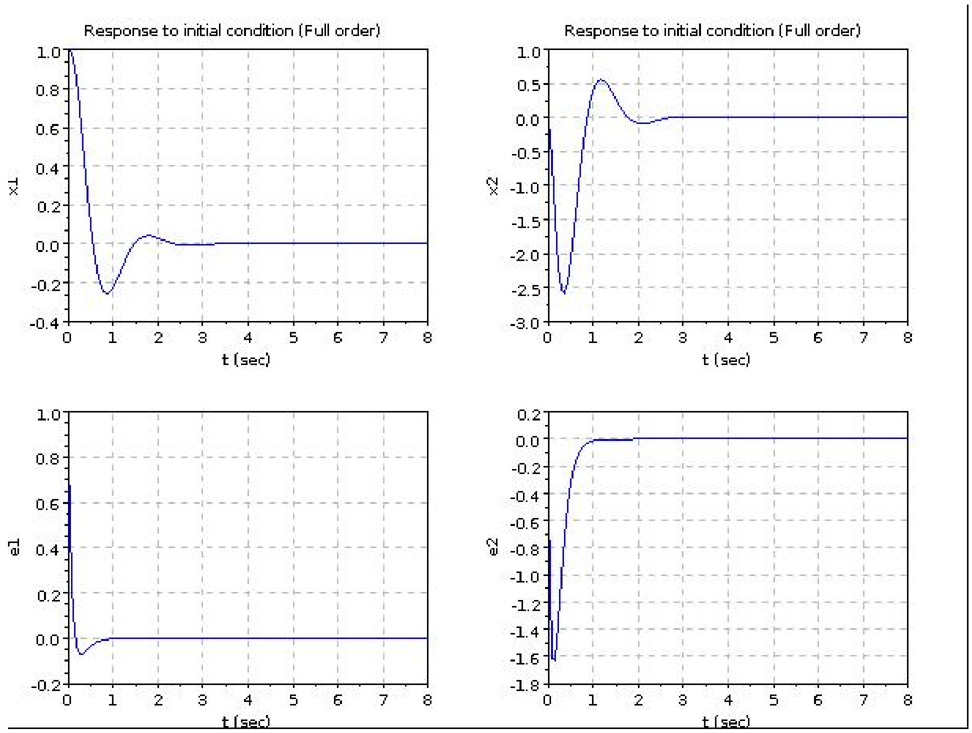

Figure 10.6: Designing a regulator using a minimum and full order observer

```
smallplot(3);
xtitle('','t (sec)','e');

scf();
// Bode diagram
C1 = Go1*Gp /. 1;
C2 = Go2*Gp /. 1;
bode([C1 ;C2],0.1,100,['System 1';'System 2']);
```

check Appendix AP 1 for dependency:

`minorder.sci`

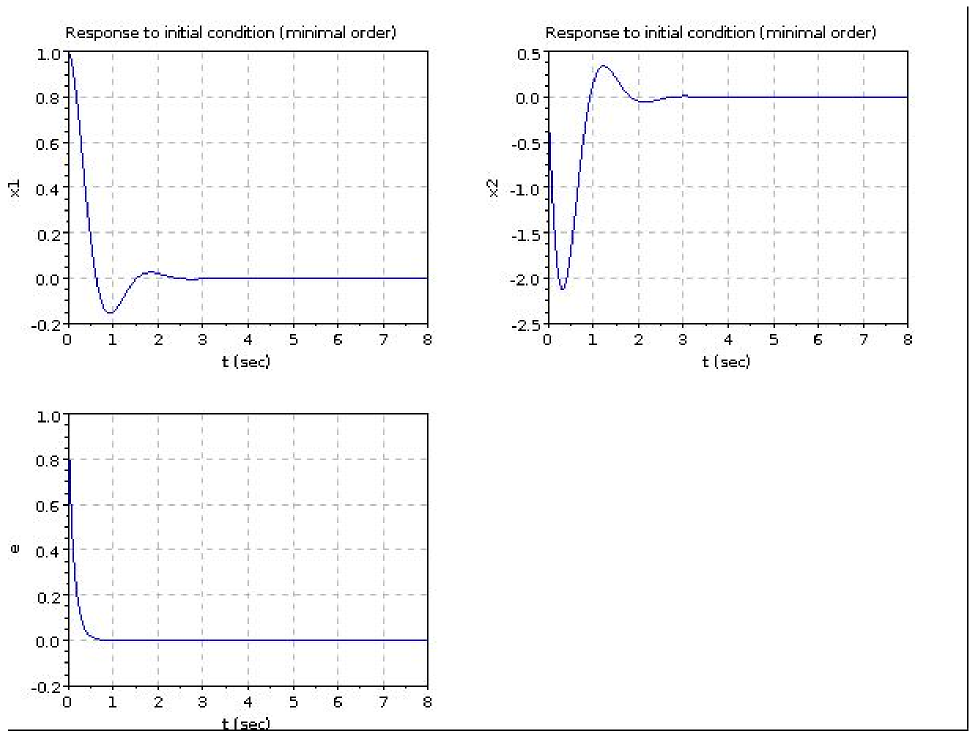

Figure 10.7: Designing a regulator using a minimum and full order observer

Scilab code Exa 10.a.17 Design of quadratic optimal regulator system and finding t

```
// Example A-10-17
// Design of quadratic optimal regulator system and
   finding the response

clear; clc;
xdel(winsid()); //close all windows
mode(0);

function smallplot(i)
  subplot(3,2,i);xgrid(color('gray'));
  plot(t,x(i,:));
endfunction

A = [0 1 0 0; 20.601 0 0 0; 0 0 0 1; -0.4905 0 0 0];
B = [0; -1; 0; 0.5];
C = [0 0 1 0];

Ahat = [A zeros(4,1); -C 0]
Bhat = [B ; 0]

Q = eye(5,5);Q(1,1) = 100
R = [0.01]

// solve the riccati equation
P = riccati(Ahat, Bhat*inv(R)*Bhat', Q,'c');
K = inv(R)*Bhat'*P
k1 = -K($);

AA = Ahat - Bhat*K
G = syslin('c',AA,[zeros(4,1); 1] , [C 0], [0]);
t = 0:0.05:10;
u = ones(1,length(t));
```

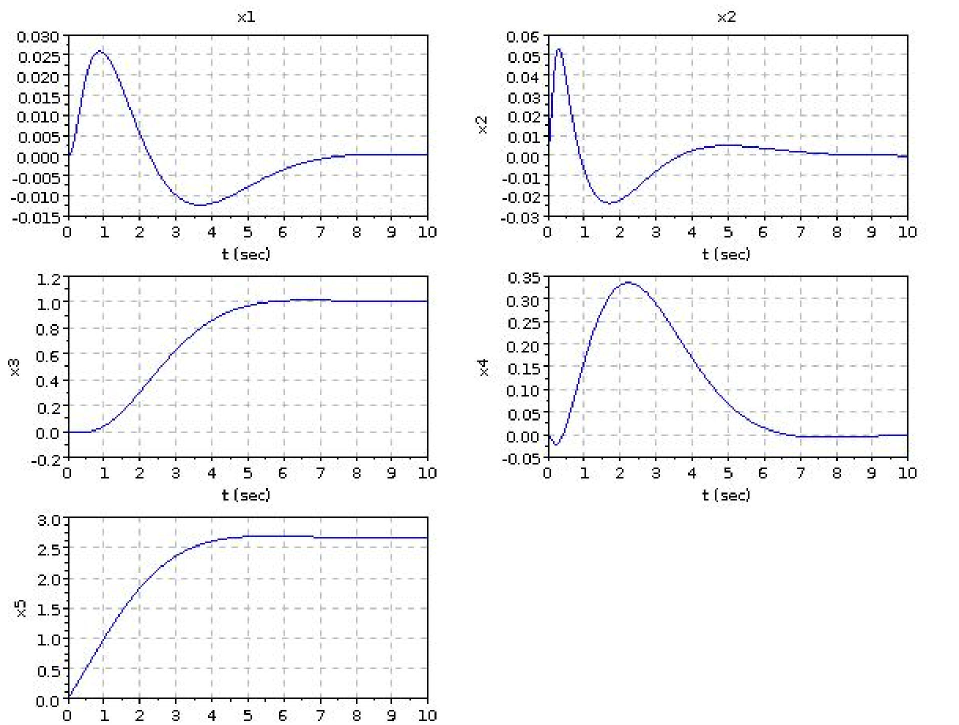

Figure 10.8: Design of quadratic optimal regulator system and finding the response

```
[y,x] = csim(u,t,G);smallplot(1);

xtitle('x1','t (sec)','');
smallplot(2);
xtitle('x2','t (sec)','x2');
smallplot(3);
xtitle('','t (sec)','x3');
smallplot(4);
xtitle('','t (sec)','x4');
smallplot(5);
xtitle('','t (sec)','x5');
```

check Appendix AP 3 for dependency:

ackermann.sci

Scilab code Exa 10.1 Gain matrix using characteristic eq and Ackermanns formula

```
// Example 10-1
// Gain matrix using characteristic eq and Ackermanns formula

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "<path to dependencies>";
// exec("ackermann.sci");

A = [0 1 0; 0 0 1;-1 -5 -6];
B = [0; 0; 1];
P = [-2 + %i*4 , -2 - %i*4, -10];

// Method 1
phi = poly(spec(A),'s');
disp(phi,'Given systems characteristic eq = ');
cf = coeff(phi);
a = cf(1:$-1)

phid = poly(P,'s');
disp(phid,'Desired characteristic eq = ');
cf = coeff(phid);
alpha = cf(1:$-1)

T = eye(3,3) // in this case
K = (alpha - a) * inv(T)

// Method 2
```

```
[K, phiA] = ackermann(A,B,P);
disp(cont_mat(A,B), ' controllability matrix = ');
disp(phiA,'phi(A) =');
disp(K,'using ackermanns formula K = ');
```

check Appendix AP 3 for dependency:

`ackermann.sci`

Scilab code Exa 10.2 Gain matrix using ppol and Ackermanns formula

```
// Example 10-2
// Gain matrix using ppol and Ackermanns formula

clear; clc;
xdel(winsid()); //close all windows

// please edit the path
// cd "<path to dependencies>";
// exec("ackermann.sci");

A = [0 1 0; 0 0 1;-1 -5 -6];
B = [0; 0; 1];
P = [-2 + %i*4 , -2 - %i*4, -10];
K = ackermann(A,B,P);disp(K,'using ackermanns
    formula K = ');
K = ppol(A,B,P);   disp(K,'using ppol function K = '
    )

// ackermann's formula is computationally tedious
// and hence avoided
```

Scilab code Exa 10.3 Response to initial condition

```
// Example 10-3
// Response to initial condition

clear; clc;
xdel(winsid()); //close all windows

A = [0 1 0; 0 0 1;-1 -5 -6];
B = [0; 0; 1];
C = [0 0 0];
D = 0;
K = [199 55 8];
x0 = [1; 0; 0]; // initial state

G = syslin('c',(A - B*K),C',C,D,x0);
t = 0:0.01:4;
u = zeros(1,length(t)); // zero input response
[y x] = csim(u,t,G);

xtitle('Response to initial condition','t (sec)','x1
    ');
subplot(3,1,1);xgrid(color('gray'));
plot(t,x(1,:));
subplot(3,1,2);xgrid(color('gray'));
xtitle('','t (sec)','x2');
plot(t,x(2,:));
subplot(3,1,3);xgrid(color('gray'));
xtitle('','t (sec)','x3');
plot(t,x(3,:));
```

check Appendix AP 2 for dependency:

`plotresp.sci`

Scilab code Exa 10.4 `Design of servo system with integrator in the plant`

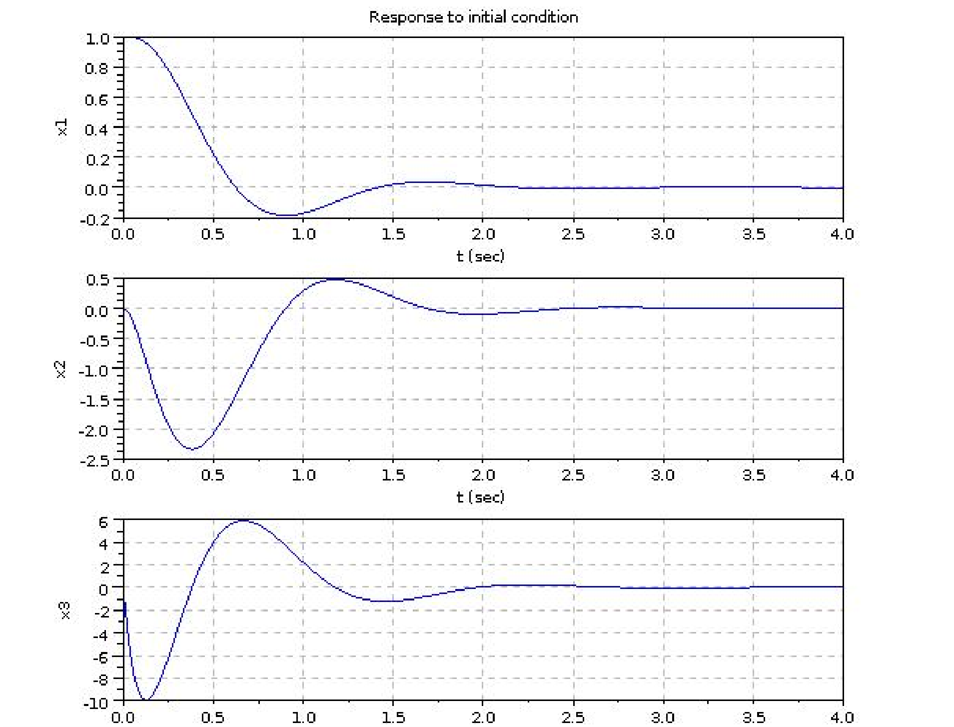

Figure 10.9: Response to initial condition

```
// Example 10-4
// Design of servo system with integrator in the
   plant

clear; clc;
xdel(winsid()); //close all windows
mode(0)

// please edit the path
// cd "<path to dependencies>";
// exec("plotresp.sci");

s = %s;
Gp = cont_frm( 1, s*(s+1)*(s+2));
A = Gp.A
B = Gp.B
J = [-2 + %i*2*sqrt(3) , -2 - %i*2*sqrt(3), -10];
K = ppol(A,B,J)

A1 = A - B*K;
B1 = [0; 0; 160];
C1 = [1 0 0];
D1 = [0];

G = syslin('c',A1,B1,C1,D1); ssprint(G);

t = 0:0.01:5;
u = ones(1,length(t));
plotresp(u,t,G,'Unit-Step Response of servo system')
   ;
```

Scilab code Exa 10.5 Design of servo system without integrator in the plant

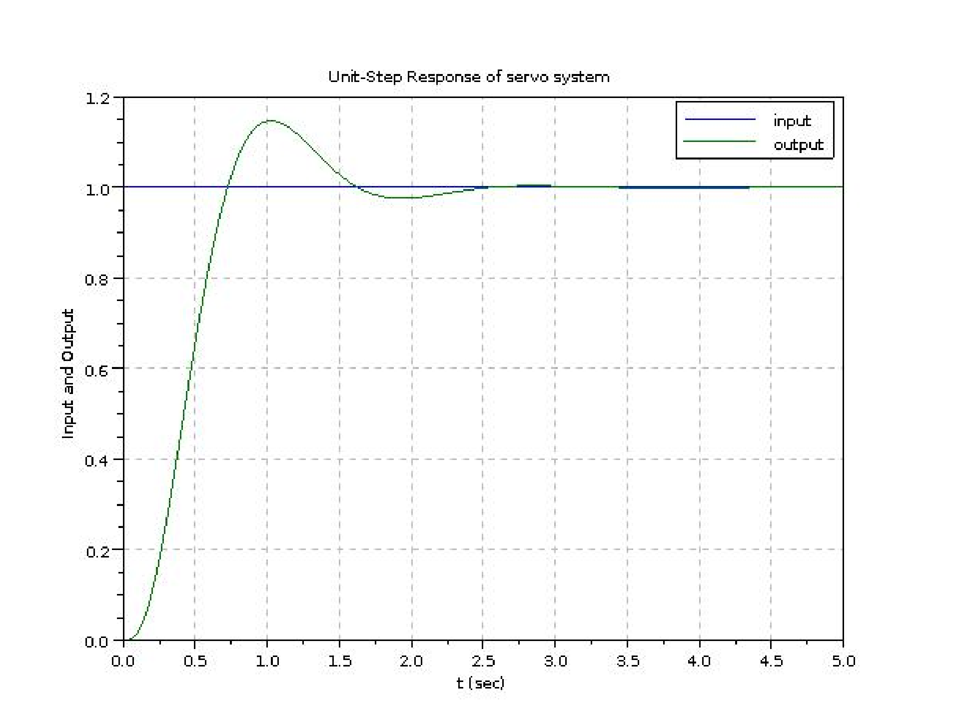

Figure 10.10: Design of servo system with integrator in the plant

```
// Example 10-5
// Design of servo system without integrator in the
   plant

clear; clc;
xdel(winsid()); //close all windows
mode(0);

function smallplot(i)
  subplot(3,2,i);xgrid(color('gray'));
  plot(t,x(i,:));
endfunction

// Plant
A = [0 1 0 0; 20.601 0 0 0; 0 0 0 1; -0.4905 0 0 0];
B = [0; -1; 0; 0.5];
C = [0 0 1 0];
J = [-1 + %i*sqrt(3) , -1 - %i*sqrt(3), -5, -5, -5];


// Error dynamics with the error as a state variable

Ahat = [A zeros(4,1); -C 0];
Bhat = [B ; 0];
Khat = ppol(Ahat,Bhat,J)
K = Khat(1: $-1)
k1 = -Khat($)

// Over all system with the error as a state
   variable
A1 = Ahat - Bhat*Khat;
B1 = [zeros(4,1); 1];
C1 = [C , 0];
D1 = [0];
G = syslin('c',A1,B1,C1,D1);

t = 0:0.02:6;
u = ones(1,length(t));
```

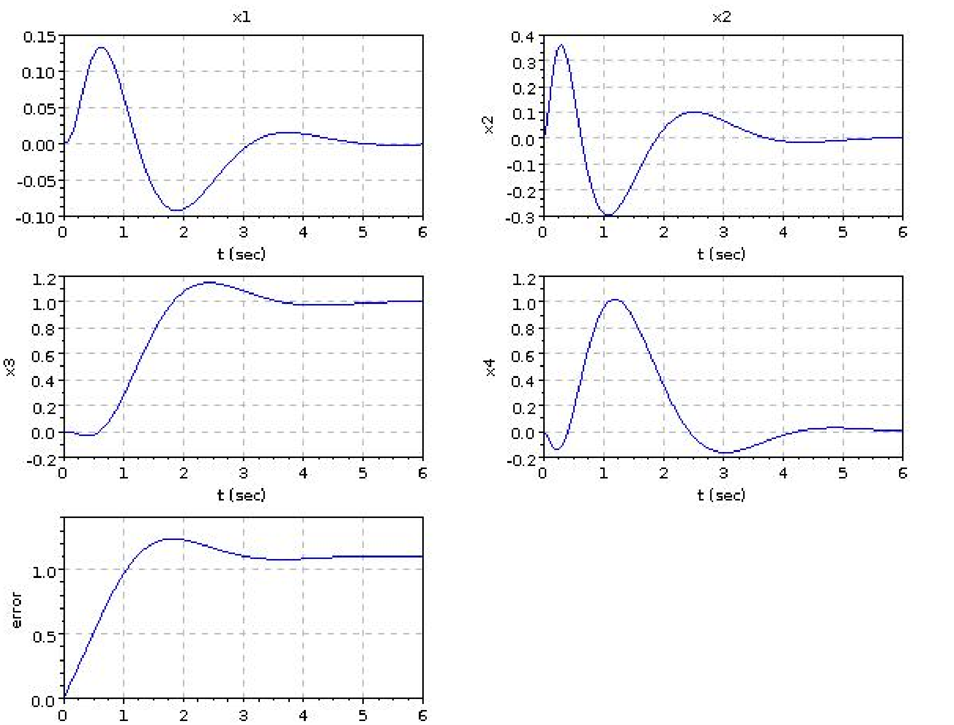

Figure 10.11: Design of servo system without integrator in the plant

```
[y ,x] = csim(u,t,G);

smallplot(1);
xtitle('x1','t (sec)','');
smallplot(2);
xtitle('x2','t (sec)','x2');
smallplot(3);
xtitle('','t (sec)','x3');
smallplot(4);
xtitle('','t (sec)','x4');
smallplot(5);
xtitle('','t (sec)','error');
```

check Appendix AP 3 for dependency:

ackermann.sci

Scilab code Exa 10.6 Observer Gain matrix using ch eq and Ackermanns formula

```
// Example 10-6
// Observer Gain matrix using ch eq and Ackermanns
   formula

clear; clc;
xdel(winsid()); //close all windows
mode(0);

// please edit the path
// cd "<path to dependencies>";
// exec("ackermann.sci");

A = [0 20.6; 1 0];
C = [0 1];
P = [-10 -10];

// Method 1
phi = poly(spec(A),'s');
disp(phi,'Given systems characteristic eq = ');
cf = coeff(phi);
a = cf(1:$-1)'

phid = poly(P,'s');
disp(phid,'Desired characteristic eq = ');
cf = coeff(phid);
alpha = cf(1:$-1)'

T = eye(2,2) // in this case
Ke = inv(T) * (alpha - a)

// Method 2
```

```
[Ke, phiA] = ackermann(A',C',P);
disp(obsv_mat(A,C), 'observability matrix = ');
disp(phiA','phi(A) =');
disp(Ke','using ackermanns formula Ke = ');
```

Scilab code Exa 10.7 Designing a controller using a full order observer

```
// Example 10-7
// Designing a controller using a full order
   observer

clear; clc;
xdel(winsid()); //close all windows
mode(0);

function smallplot(i)
  subplot(2,2,i);xgrid(color('gray'));
  plot(t,x(i,:));
endfunction

s = %s;
A = [0 1; 20.6 0];
B = [0; 1];
C = [1 0];
D = [0];
P = [-1.8 + %i*2.4 ,-1.8 - %i*2.4 ];
Q = [-8 -8]; // observer poles

K = ppol(A,B,P)
Ke = ppol(A',C',Q)'

// The transfer function of observer controller
A1 = A - B*K - Ke*C
M = s*eye(A1) - A1
UbyE = K * inv(M) * Ke;
```

```
disp(UbyE, 'U(s) / E(s) =');

// Plant dynamics
Gp = syslin('c',A,B,C,D);
disp('plant  dynamics'); ssprint(Gp);
YbyU = ss2tf(Gp)

// Observer controller dynamics
disp('observer controller dynamics (x = xbar) ,(u = y), (y = u)');
Goc = syslin('c',A1,Ke,-K,[0]);
ssprint(Goc);

// Overall System transfer funtion

GsFullsystem = UbyE * YbyU /. 1

// Overall System
x0 = [1; 0; 0.5; 0];  // initial state
As = [A-B*K, B*K ; zeros(2,2) , A-Ke*C];
Gss = syslin('c',As,[1;0;0;0], [1 0 0 0], [0],x0);

// Unit step response
t = 0:0.01:4;
u = zeros(1,length(t));
[y x] = csim(u,t,Gss);

smallplot(1);
xtitle('Response to initial condition','t (sec)','x1');
smallplot(2);
xtitle('Response to initial condition','t (sec)','x2');
smallplot(3);
xtitle('','t (sec)','e1');
smallplot(4);
xtitle('','t (sec)','e2');
```

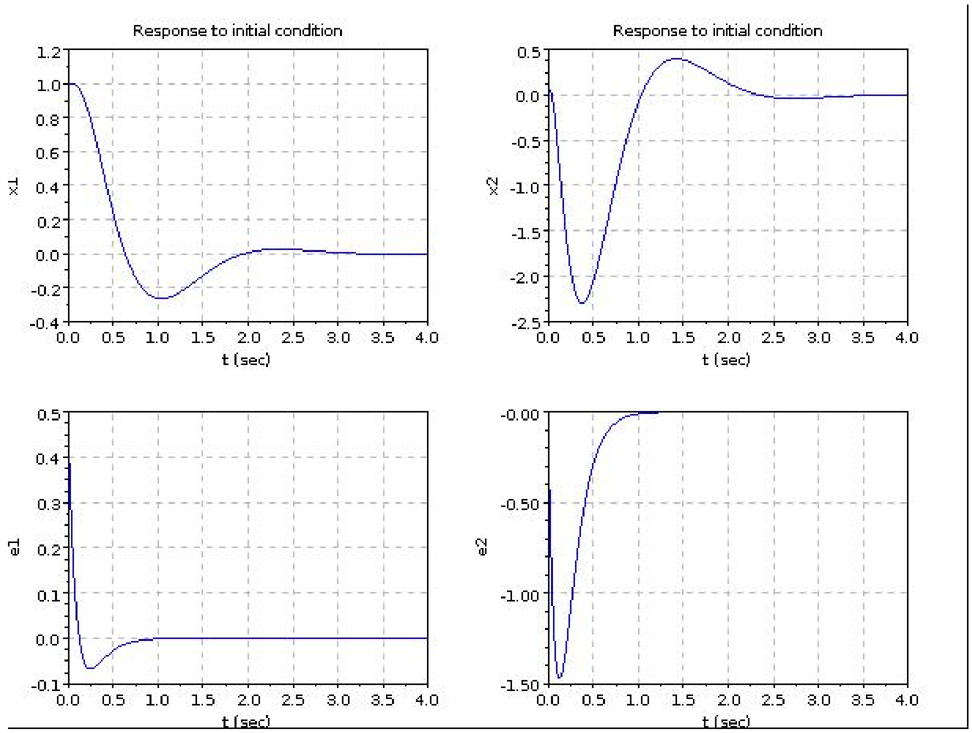

Figure 10.12: Designing a controller using a full order observer

Scilab code Exa 10.8 `Designing a controller using a minimum order observer`

```
// Example 10-8
// Designing a controller using a minimum order observer

clear; clc;
xdel(winsid()); //close all windows
mode(0);

```

```
A = [0 1 0; 0 0 1; -6 -11 -6];
B = [0; 0; 1];
C = [1 0 0];
D = [0];
P = [-2 + %i*2*sqrt(3),-2 - %i*2*sqrt(3),-6];
Q = [-10 -10];  // observer poles

K = ppol(A,B,P)

// Observer design
Aaa = A(1,1)
Aab = A(1,2:$)
Aba = A(2:$,1)
Abb = A(2:$,2:$)

Ke = ppol(Abb',Aab',Q)'

Ba = B(1,1)
Bb = B(2:$,1)

Ahat = Abb - Ke*Aab;
disp(Ahat,'Ahat = Abb - Ke*Aab =');
Bh = Aba - Ke*Aaa;
disp(Bh,'Aba - Ke*Aaa =');
Chat = [zeros(1,2); eye(2,2)]
Dhat = [1; Ke]
Fhat = Bb - Ke*Ba;
disp(Fhat,'Fhat = Bb - Ke*Ba =');
```

Scilab code Exa 10.9 Design of quadratic optimal regulator system

```
// Example 10.9
// Design of quadratic optimal regulator system

clear; clc;
```

```
xdel(winsid());  //close all windows
mode(0);

A = [0 1;0 0];
B = [0;1];
Q = [1 0; 0 1];
R = [1];

// solve the riccati equation
P = riccati(A, B*inv(R)*B', Q,'c')
K = inv(R)*B'*P
E = spec(A - B*K) // eigen values
```

Scilab code Exa 10.10 Design of quadratic optimal regulator system

```
// Example 10-10
// Design of quadratic optimal regulator system

clear; clc;
xdel(winsid());  //close all windows
mode(0);

A = [-1 1;0 2];
B = [1;0];
Q = [1 0; 0 1];
R = [1];

// solve the riccati equation
P = riccati(A, B*inv(R)*B', Q,'c')
K = inv(R)*B'*P
E = spec(A - B*K) // eigen values
// when a solution does not exist
// a different method is used - least square
   solution
```

Scilab code Exa 10.11 Design of quadratic optimal regulator system

```
// Example 10-11
// Design of quadratic optimal regulator system

clear; clc;
xdel(winsid()); //close all windows
mode(0);

A = [0 1;0 -1];
B = [0;1];
Q = [1 0; 0 1];
R = [1];

// solve the riccati equation
P = riccati(A, B*inv(R)*B', Q,'c')
K = inv(R)*B'*P
E = spec(A - B*K) // eigen values
```

Scilab code Exa 10.12 Design of quadratic optimal regulator system and finding the

```
// Example 10-12
// Design of quadratic optimal regulator system and
   finding the response

clear; clc;
xdel(winsid()); //close all windows
mode(0);

A = [0 1 0; 0 0 1; -35 -27 -9];
B = [0; 0; 1];
Q = [1 0 0; 0 1 0; 0 0 1];
```

```
R = [1];

// solve the riccati equation
P = riccati(A, B*inv(R)*B', Q,'c')
K = inv(R)*B'*P
E = spec(A - B*K) // eigen values

x0 = [1; 0; 0];  // initial state

G = syslin('c',(A - B*K),[0;0;0],[0 0 0],[0], x0);
t = 0:0.01:8;
u = zeros(1,length(t));
[y x] = csim(u,t,G);

xtitle('Response to initial condition','t (sec)','x1');
subplot(3,1,1);xgrid(color('gray'));
plot(t,x(1,:));

subplot(3,1,2);xgrid(color('gray'));
xtitle('','t (sec)','x2');
plot(t,x(2,:));

subplot(3,1,3);xgrid(color('gray'));
xtitle('','t (sec)','x3');
plot(t,x(3,:));
```

Scilab code Exa 10.13 Design of quadratic optimal regulator system and finding the

```
// Example 10-13
// Design of quadratic optimal regulator system

clear; clc;
```

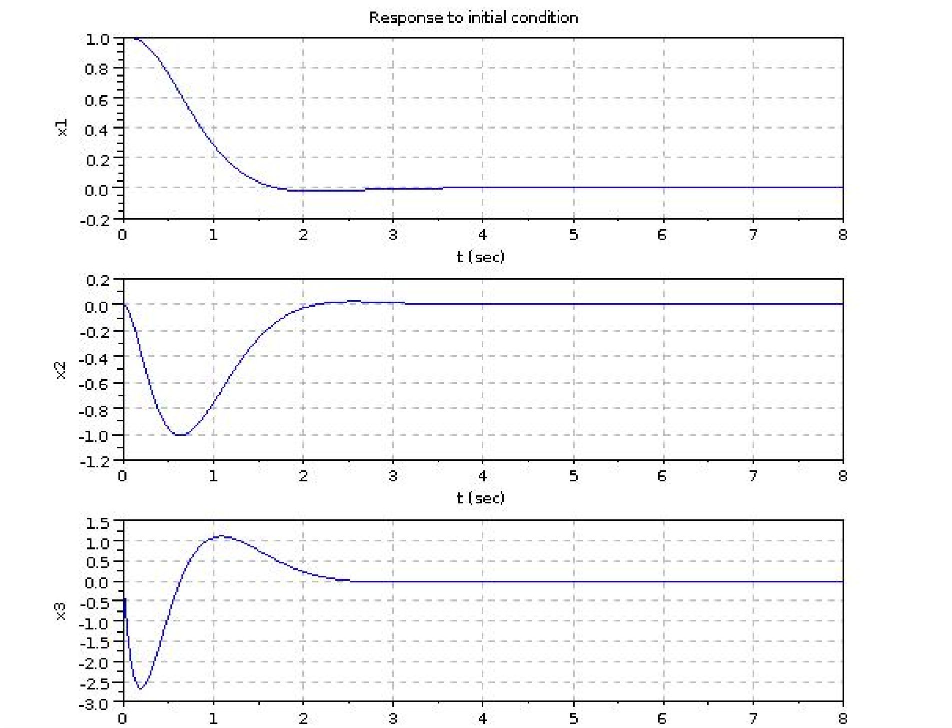

Figure 10.13: Design of quadratic optimal regulator system and finding the response

```
xdel(winsid());   //close all windows

A = [0 1 0; 0 0 1; 0 -2 -3];
B = [0; 0; 1];
C = [1 0 0];
Q = [100 0 0; 0 1 0; 0 0 1];
R = [0.01];

// solve the riccati equation
P = riccati(A, B*inv(R)*B', Q,'c');
K = inv(R)*B'*P;
disp(K,'K = ');
k1 = K(1);

G = syslin('c',A - B*K, B*k1 , C, [0]);
t = 0:0.01:8;
u = ones(1,length(t));
[y,x] = csim(u,t,G);
plot(t,x);
xgrid(color('gray'));
xtitle('Step-Response','t (sec)','state variables');
legend('x1 (= y)','x2','x3');
```

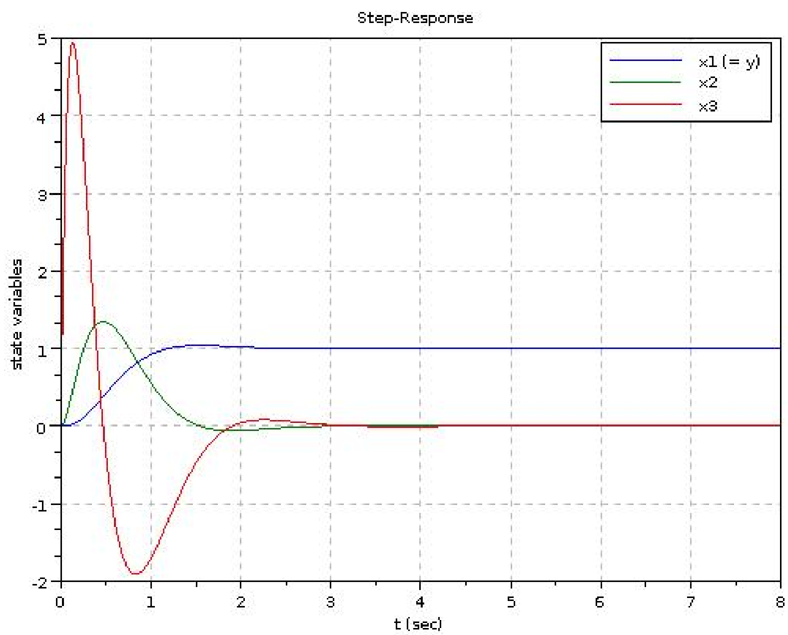

Figure 10.14: Design of quadratic optimal regulator system and finding the response

Appendix

Scilab code AP 1 Determine Gains and transfer function for minimal order observer

```

// Determine Gains and transfer function for minimal
    order observer

function G = transferf(A,B,C,D)
  H = syslin('c',A,B,C,D);
  G = clean(ss2tf(H));
endfunction

function [K,Ke,Go,ch,AA,Ahat,Bhat,Chat,Dhat,Fhat] =
   minorder(A,B,P,Q)
  s = %s;
  K = ppol(A,B,P);
  Ka = K(1);
  Kb = K(2:$);

  Aaa = A(1,1);
  Aab = A(1,2:$);
  Aba = A(2:$,1);
  Abb = A(2:$,2:$);
  Ba = B(1,1);
  Bb = B(2:$,1);

  Ke = ppol(Abb',Aab',Q)'

```

```
  n = length(Kb);
  Ahat = Abb - Ke*Aab;
  Bhat = Ahat*Ke + Aba - Ke*Aaa;
  Chat = [zeros(1,n); eye(n,n)];
  Dhat = [1; Ke];
  Fhat = Bb - Ke*Ba;
  Atld = Ahat - Fhat*Kb;
  Btld = Bhat - Fhat*(Ka + Kb*Ke);
  Ctld = -Kb;
  Dtld = -(Ka + Kb*Ke);

  Go = transferf(Atld,Btld,-Ctld,-Dtld);
ch = det(s*eye(n+1,n+1) - A + B*K) * det(s*eye(n,n)
   - Abb + Ke*Aab);
AA = [A - B*K , B*Kb; zeros(n,n+1) , Abb - Ke*Aab];

endfunction
```

Scilab code AP 2 Plot System Response

```

// Plot System Response
// Computes the response and plots the input and
   response together

function y = plotresp(u,t,G,text)
  y = csim(u,t,G);
  plot(t,u,t,y);
  xtitle(text,'t (sec)','Input and Output');
  xgrid(color('gray'));
  legend('input','output');
endfunction
```

Scilab code AP 3 Compute the feedback gain matrix using ackermanns formula

```
// Compute the feedback gain matrix using ackermanns
    formula
```

```

function [K ,phiA] = ackermann(A,B,P)
  // construct charecteristic equation
  phi = poly(P,'x');
  c = coeff(phi);
  phiA = eye(A)*c(1);
  powA = eye(A);
  for i=2:length(c)
        powA = powA * A;
        phiA = phiA + powA * c(i);
  end
  K = [zeros(1,length(B)- 1), 1] * inv(cont_mat(A,B)) * phiA;
endfunction
```

Scilab code AP 4 Transfer function of A,B,C,D.

```

function G = transferf(A,B,C,D)
  H = syslin('c',A,B,C,D);
  G = clean(ss2tf(H));
endfunction
```

Scilab code AP 5 Inverse Laplace transform of a rational polynomial in s

```
// Inverse Laplace transform of a rational polynomial in s
// depends on pf_residu

function s = ilaplace(H)
  if(H ~= 0) then
    [r z p] = pf_residu(H.num,H.den);
    n = length(r);
    s = '';
    for i = 1:(n-1) ;
      s = s + string(r(i)) + '*e^' + string(p(i)) + 't  + ';
    end
```

```
    s = s + string(r(n)) + '*e^' + string(p(n)) + 't
       ';
  else
    s ='0';
  end
endfunction
```

Scilab code AP 6 Partial Fraction Residue

```

// Partial Fraction Residue
// Gives the coefficients of partial fraction
   expansion for the given polynomial

function [r,z,p] = pf_residu(N,D)
  z = roots(N)   //Zeros
  p = roots(D)   //Poles

  q = round(p);
  m = 1;    // to keep a count of the root's
     multiplicity

  for i = 1:length(p)
    if(i < length(p) & q(i + 1) == q(i))
      m = m + 1;
    else
      P1 = N / pdiv(D,( s - p(i)) ^ m );
      r(i) = horner(P1 ,p(i));
      for j = 1:(m-1)
        P1 = derivat(P1);
        r(i - j) = horner(P1 / gamma(j + 1) ,p(i));
      end             // gamma(j + 1) = j! (factorial
        )
      m = 1;
    end
  end
endfunction

```

```
// for details on this method please refer
// http://en.wikipedia.org/wiki/Partial_fraction
```

Scilab code AP 7 Plot the root locus in a box

```
// Plot the root locus in a box
// rootl(G,box,text)
// G   : linear system
// box: so ordinates of axis bounds
// text: title of plot window

function rootl(G,box,text)
  evans(G);
  xgrid();
  a = gca();
  if box ~= 0 then
    a.box = "on";
    a.data_bounds = box;
  end
  a.children(1).visible = 'off'; //remove the legend
      block
  xtitle(text);
endfunction
```

Scilab code AP 8 Step response characteristics

```
// Step response characteristics
// Plots the step response and computes Maximum
   Overshoot
// Peak Time,Rise Time and Settling Time

function [Mp,tp,tr,ts] = stepch(G,from,to,step,
   settling_margin)

  t = from:step:to;
  u = ones(1,length(t));
  y = csim(u,t,G);
  plot(t,y);
```

```
  xtitle('Unit Step Response','t (sec)','Output');
  xgrid(color('gray'));

  [m t1] = max(y);
  tp = (t1 - 1) * step;
  Mp = m - 1;

  i = 1;
  if tp == to then
    tr = %nan;
  else
    while(y(i) < 0.1) i = i + 1; end;
    r1 = i;
    while(y(i) < 0.9) i = i + 1; end;
    tr = (i-r1) * step;
  end

  l = 1 - settling_margin;
  h = 1 + settling_margin;
  for i = length(t):-1:1
    if( y(i) < l | y(i) > h) break; end;
  end
  ts = (i - 1) * step;
endfunction
```

Scilab code AP 9 Polar plot of a linear system

```
// polar plot of a linear system
// repf = spolarplot(G,omega)
// G: linear sytem and omega:is frequency  in rad/s
// repf: is the complex frequency response

function repf = spolarplot(G,omega)
  f = omega /2/%pi;
  repf = repfreq(G,f);
  r = abs(repf);
  theta = atan(imag(repf),real(repf));
  polarplot(theta,r,style = 2);
```

```
endfunction
```

Scilab code AP 10 Display gain and phase margins

```
// Display gain and phase margins on a bode plot

function [gm,gcrf,pm,pcrf] = shmargins(G)

  show_margins(G,'bode');
  xtitle('Bode diagram','rad/s');
  a = gcf();set(a.children(2).x_label,'text','rad/s'
     );

  [gm pcrf] = g_margin(G);
  [pm gcrf] = p_margin(G);
  disp(gcrf,'Gain crossover frequency = ',pm,'Phase
     margin (degrees)= ');
  disp(pcrf,'Phase crossover frequency = ',gm,'Gain
     margin (dB) = ');
endfunction
```

Scilab code AP 11 Frequency response characteristics

```
// Frequency response characteristics
function [Mr,wr,bw,repf] = freqch(G,omega)

  repf = repfreq(G,omega);    // frequency response
     (complex numbers)

  [mag phi] = dbphi(repf);  // mag in db
  [Mr k] = max(mag);        // resonant peak
  wr = omega(k);            // resonant freq.
  mag = abs(mag + 3);       // mag = abs( mag - (- 3
     dB) )
  [M j] = min(mag);         // j : is the point
     where mag == -3db
  bw = omega(j);

```

```
  disp(wr,'resonant frequency = ');
  disp(Mr,'resonant peak (dB)= ');
  disp(bw,'bandwidth = ');
endfunction
```

Scilab code AP 12 Gain at a point on a root locus

```
// Gain at a point on a root locus

function [K,p] = gainat(G)
  z = locate(1,1);
  x = z(1);y = z(2);
  p = x + %i*y;
  disp( p , 'p = ');
  K = 1 / abs(horner(G,p))
  disp( K , 'K = ');
  plot(x,y,'.');
  xstring(x,y,'K = ' + string(K));
endfunction
```