Scilab Textbook Companion for Design With Operational Amplifiers And Analog Integrated Circuits by S. Franco[1]

Created by
Prashant Gurumukhdas Goklani
B.E
Others
V.E.S.I.T, Mumbai University
College Teacher
Nandini Ammagi
Cross-Checked by
Ganesh R

May 23, 2016

[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: Design With Operational Amplifiers And Analog Integrated Circuits

Author: S. Franco

Publisher: Tata McGraw - Hill Education, New Delhi

Edition: 3

Year: 2011

ISBN: 9780070530447

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

1 Operational Amplifier Fundamentals 6

2 Circuits with Resistive Feedback 43

3 Active Filters Part I 74

4 Active Filters Part II 108

5 Static Op Amp Limitations 142

6 Dynamic Op Amp Limitations 162

7 Noise 186

8 Stability 203

9 Non Linear Circuits 227

10 Signal Generators 236

11 Voltage Referencres and Regulators 248

12 D to A and A to D Converters 277

13 Non Linear Amplifiers and Phase Locked Loops 285

List of Scilab Codes

Exa 1.1.a Amplifier Fundamentals . . . 6
Exa 1.1.b Amplifier Fundamentals . . . 7
Exa 1.2.a Gain of a Noninverting OP AMP . . . 8
Exa 1.2.b Gain of a Noninverting OP AMP . . . 8
Exa 1.2.c Gain of a Noninverting OP AMP . . . 9
Exa 1.3 Inverting Amplifier Ideal Closed Loop Characterstics . 10
Exa 1.4 Designing Summing Amplifiers . . . 10
Exa 1.5 Designing Function Generators . . . 11
Exa 1.6 Designing Difference Amplifier . . . 12
Exa 1.7.a Application of Negative Feedback . . . 13
Exa 1.7.b Application of Negative Feedback . . . 14
Exa 1.7.c Application of Negative Feedback . . . 14
Exa 1.8.a Calculating Gain Desensitivity . . . 15
Exa 1.8.b Calculating Gain Desensitivity . . . 16
Exa 1.9.a Noninverting Configuration Characterstics . . . 17
Exa 1.9.b Noninverting Configuration Characterstics . . . 19
Exa 1.10.a Inverting Configuration Characterstics . . . 21
Exa 1.10.b Inverting Configuration Characterstics . . . 22
Exa 1.11.a Finding the Loop Gain . . . 23
Exa 1.11.b Finding the Loop Gain . . . 24
Exa 1.12.a Feedback Factor for Negative Feedback . . . 26
Exa 1.12.b Feedback Factor for Negative Feedback . . . 27
Exa 1.13 Feedback Factor for Combination of Negative and Positive Feedback . . . 30
Exa 1.14.a Current Flow and Power Dissipation . . . 31
Exa 1.14.b Current Flow and Power Dissipation . . . 32
Exa 1.15.a Designing variable dc source . . . 33
Exa 1.15.b Designing variable dc source . . . 34

Exa 1.16 Inverting Amplifier driven into saturation 36
Exa 2.1 Closed Loop Parameters of Basic I V Conveter 43
Exa 2.2 Designing High Sensitivity I V Conveter 44
Exa 2.3.a Characterstics of Floating Load V I Converters 45
Exa 2.3.b Characterstics of Floating Load V I Converters 46
Exa 2.3.c Characterstics of Floating Load V I Converters 47
Exa 2.4 Designing Current Source using Grounded Load Converter 48
Exa 2.5.a Effect of Resistance Mismatches in Grounded Load Converters 49
Exa 2.5.b Effect of Resistance Mismatches in Grounded Load Converters 50
Exa 2.5.c Effect of Resistance Mismatches in Grounded Load Converters 51
Exa 2.6 Howland Circuit Calibration 51
Exa 2.7 Effect of finite loop gain on Howland Circuit 52
Exa 2.8.a Output Voltage of a Difference Amplifier 53
Exa 2.8.b Output Voltage of a Difference Amplifier 55
Exa 2.9 Common Mode Rejection Ratio for op amp 57
Exa 2.10.a Designing Triple Op Amp Instrumentation Amplifier . 58
Exa 2.10.b Designing Triple Op Amp Instrumentation Amplifier . 60
Exa 2.10.c Designing Triple Op Amp Instrumentation Amplifier . 62
Exa 2.11.a Study of Resistance Temperarure Detector 63
Exa 2.11.b Study of Resistance Temperarure Detector 64
Exa 2.11.c Study of Resistance Temperarure Detector 65
Exa 2.12.a Designing a Transducer Bridge with Instrumentation Amplifier 65
Exa 2.12.b Designing a Transducer Bridge with Instrumentation Amplifier 66
Exa 2.13.a Transducer Bridge Calibration 68
Exa 2.13.b Transducer Bridge Calibration 70
Exa 2.14 Designing Strain Gauge Bridge with Instrumentation Amplifier 71
Exa 3.1 Pole Zero Response of Transfer Function 74
Exa 3.2 Finding Impulse Response of a given circuit 75
Exa 3.3 Steady State Response of a Circuit 76
Exa 3.4.a Low pass filter with Gain 77
Exa 3.4.b Low pass filter with Gain 78

Exa 3.5 Designing Wideband Band Pass Filter 80
Exa 3.6 Designing Phono Amplifier 81
Exa 3.7 Designing a bass or treble control 83
Exa 3.8 Designing Equal Component Second Order Low Pass Filter 84
Exa 3.9 Designing Second Order Low Pass Filter for 0dB dc gain 85
Exa 3.10 Designing a Unity Gain Low Pass Filter 87
Exa 3.11.a Designing Butterworth Low Pass Filter 88
Exa 3.11.b Designing Butterworth Low Pass Filter 89
Exa 3.12 Designing High Pass KRC Filters 90
Exa 3.13.a Designing KRC Bandpass Filter 92
Exa 3.13.b Designing KRC Bandpass Filter 93
Exa 3.14 Designing Band Reject KRC Filter 95
Exa 3.15 Designing Multiple Feedback Band Pass Filter 96
Exa 3.16 Designing Multiple Feedback Low Pass Filter 97
Exa 3.17 Designing Multiple Feedback Notch Filters 99
Exa 3.18 Designing State Variable Filter for Bandpass Response 100
Exa 3.19 Designing a Biquad Filter 101
Exa 3.20 Designing Biquad Filter for a low pass notch response 103
Exa 3.21.a KRC Filter Sensitivities 105
Exa 3.21.b KRC Filter Sensitivities 106
Exa 4.1 Butterworth Filter Approximations 108
Exa 4.2 Cascade Designing of Chebyshev Low Pass Filter . . . 109
Exa 4.3 Cascade Designing of Cauer Low Pass Filter 112
Exa 4.4 Designing a Chebyshev High Pass Filter 116
Exa 4.5 Cascade Designing of Butterworth Band Pass Filter . 118
Exa 4.6 Cascade Designing of Elliptic Band Pass Filter 121
Exa 4.7 Cascade Designing of Chebyshev Band Reject Filter . 125
Exa 4.8 Designing a Dual Amplifier Band Pass Filter 129
Exa 4.9 Designing a General Impedance Converter Low Pass Filter 130
Exa 4.10 Direct Designing of Low Pass Filter 132
Exa 4.11 Direct Designing of High Pass Filter 135
Exa 4.12 Designing a Switched Capacitor Biquad Filter 137
Exa 4.13 Direct Synthesis of Switched Capacitor Low Pass Filter 138
Exa 4.14 Direct Synthesis of Switched Capacitor Band Pass Filter 139
Exa 5.1.a Errors caused by Input Bias and Offset Current 142
Exa 5.1.b Errors caused by Input Bias and Offset Current 143

Exa 5.1.c Errors caused by Input Bias and Offset Current 144
Exa 5.1.d Errors caused by Input Bias and Offset Current 144
Exa 5.2.a Errors caused by Input Bias and Offset Current II . . 145
Exa 5.2.b Errors caused by Input Bias and Offset Current II . . 146
Exa 5.3 Input Bias Current Drift 147
Exa 5.4.a Error in Input Offset due to CMRR 148
Exa 5.4.b Error in Input Offset due to CMRR 149
Exa 5.5 Error in Input Offset due to PSRR 149
Exa 5.6 Change of offset voltage with the Output Swing 150
Exa 5.7 Input Offset Error Compensation using Internal Offset Nulling 152
Exa 5.8 Input Offset Error Compensation using External Offset Nulling I 153
Exa 5.9.a Input Offset Error Compensation using External Offset Nulling II 155
Exa 5.9.b Input Offset Error Compensation using External Offset Nulling II 156
Exa 5.10 Input Offset Error Compensation in Multiple Op Amp Circuits 158
Exa 5.11 Absolute Maximum Ratings 160
Exa 5.12 Overload Protection Maximum Ratings 161
Exa 6.1.a Closed Loop Response of Non Inverting Amplifier . . . 162
Exa 6.1.b Closed Loop Response of Non Inverting Amplifier . . . 163
Exa 6.2.a Gain Bandwidth Tradeoff 163
Exa 6.2.b Gain Bandwidth Tradeoff 164
Exa 6.2.c Gain Bandwidth Tradeoff 167
Exa 6.4 Input Impedance of Series Topology 168
Exa 6.5 Output Impedance of Shunt Topology 169
Exa 6.6.a Finding Gain Zi and Zo for High Sensitivty I V Converter 170
Exa 6.7.a Effect of Slew Rate Limiting 172
Exa 6.8.a Full Power Bandwidth 174
Exa 6.8.b Full Power Bandwidth 175
Exa 6.8.c Full Power Bandwidth 175
Exa 6.8.d Full Power Bandwidth 176
Exa 6.9 Effect of finite GBP on Integrator Circuits 177
Exa 6.10.b Biquad Filter with Phase Compensation 178
Exa 6.11 Effect of finite GBP on first order filter 180
Exa 6.12 Effect of finite GBP on second order filter 181

Exa 6.14	Parameters for Current Feedback Amplifier	182
Exa 6.15	Current Feedback Amplifier Dynamics	183
Exa 6.16	Compensation of B W Reduction in Current Feedback Amplifier	184
Exa 7.1.a	Noise Properties	186
Exa 7.1.b	Noise Properties	187
Exa 7.1.c	Noise Properties	187
Exa 7.3	Graphical Representation of Noise Dynamics	188
Exa 7.4	Calculation of Thermal Noise	189
Exa 7.5.a	Calculation of Shot Noise	190
Exa 7.5.b	Calculation of Shot Noise	190
Exa 7.7.a	Total Output Noise in an Op Amp	191
Exa 7.8	Improvement in the Circuit to find the Total Output Noise	192
Exa 7.9	Calculation of Signal to Noise Ratio	195
Exa 7.10	Calculation of Noise in Current Feedback Amplifier	195
Exa 7.11	Noise in Photodiode Amplifiers	197
Exa 7.12	Photodiode amplifier with Noise Filtering	199
Exa 7.13	Designing T Feedback Photodiode Amplifiers	200
Exa 8.1	Gain Margin and Phase Margin of an op amp system	203
Exa 8.2	Stability in Differentiator Circuits	204
Exa 8.3	Stray Input Capacitance Compensation for inverting configuration	205
Exa 8.4	Stray Input Capacitance Compensation for non inverting configuration	206
Exa 8.5	Stabalizing a capacitively loaded op amp circuit	208
Exa 8.6	Internal Frequency Compensation	209
Exa 8.7	Dominant Pole Compensation	210
Exa 8.8	Shunt Capacitance Compensation	211
Exa 8.9	Miller Compensation	212
Exa 8.10	Pole Zero Compensation	214
Exa 8.11	Frequency Compensation via Loop Gain Reduction	216
Exa 8.12	Input Lag Compensation	217
Exa 8.13	Feedback Lead Compensation	219
Exa 8.14	Configuring a Decompensated op amp as a Unity Gain Voltage Follower	220
Exa 8.15	Input Stray Capacitance Compensation in CFA Circuits	221
Exa 8.16	Feedback Lead Compensation for Composite Amplifier	222

Exa 8.17 Composite Amplifier with Compensation provided by op amp 2 . . . 224
Exa 9.1 Comparator as a Level Detector I . . . 227
Exa 9.2 Comparator as a Level Detector II . . . 228
Exa 9.3 Designing On Off Temperature Controller . . . 229
Exa 9.4 Comparator as a Window Detector . . . 231
Exa 9.5 Designing Single Supply Inverting Schmitt trigger . . . 232
Exa 9.6 Hysteresis in On Off Controllers . . . 233
Exa 10.1 Designing a Square Wave Generator using Multivibrator 236
Exa 10.3 The 555 timer as an astable multivibrator . . . 238
Exa 10.4 Voltage Control for 555 timer . . . 239
Exa 10.5 Designing Basic Triangular or Square Wave Generator 241
Exa 10.6 Basic ICL8038 connection for 50 percent duty cycle operation . . . 243
Exa 10.7 AD537 application as a temperature to frequency converter . . . 244
Exa 10.8 Designing a Voltage to Frequency Converter . . . 245
Exa 11.1 Line and Load Regulation . . . 248
Exa 11.2 Thermal Coeffecient . . . 249
Exa 11.3 Application of Line and Load Regulation . . . 251
Exa 11.4 Line and Load Regulation of an op amp . . . 252
Exa 11.5 Bandgap Voltage Reference . . . 253
Exa 11.6 Turning a Voltage Reference into a current source . . . 254
Exa 11.7 Current Sources with Current Boosting Transistors . . 256
Exa 11.8 Thermal cold junction compensation using AD590 . . 257
Exa 11.9 Basic Series Regulator . . . 258
Exa 11.10 Overload Protections for Linear Regulators . . . 260
Exa 11.11 Positive Regulator with overload SOA and thermal protection . . . 261
Exa 11.12 Configuring a regulator as a power voltage source . . . 262
Exa 11.13 Configuring a regulator as an adjustable Power Current Source . . . 263
Exa 11.14 Thermal Considerations for Linear Regulator . . . 265
Exa 11.15 Selection of Heat Sink on the basis of Thermal Resistance 266
Exa 11.16 Overvoltage Protection and Under Voltage Sensing . . 267
Exa 11.17 Duty Cycle of a Buck Regulator . . . 268
Exa 11.18 Coil Selection for a Boost Regulator . . . 270
Exa 11.19 Capacitor Selection for a Boost Regulator . . . 271

Exa 11.20 Efficiency of Buck regulator 272
Exa 11.21 Designing Error Amplifier for Buck Regulator 274
Exa 12.1 Specifications of DAC 277
Exa 12.2 Specifications of ADC 278
Exa 12.3 DAC using a current mode R 2R ladder 279
Exa 12.4 Designing Digitally Programmable filter 280
Exa 12.5 Designing Digitally programmable triangular or square wave oscillator 281
Exa 12.6 Concept of Oversampling 282
Exa 12.7 Noise Shaping and Integrate Difference Converters . . 283
Exa 13.1 Stabilty Considerations for Log and Antilog Amplifiers 285
Exa 13.2 Operational Transconducatance Amplifiers 286
Exa 13.3 Response of a first order Phase Locked Loop 288
Exa 13.4 Response of a second order Phase Locked Loop 290
Exa 13.5 Damping Characterstics of Phase Locked Loop 291
Exa 13.6 Filter Design Criteria 293
Exa 13.7 Designing with PLLs 294
Exa 13.8 Designing Frequency Synthesizer using PLL 296

List of Figures

1.1 Inverting Amplifier driven into saturation 37

3.1 Pole Zero Response of Transfer Function 75

6.1 Gain Bandwidth Tradeoff 165
6.2 Effect of Slew Rate Limiting 172

Chapter 1

Operational Amplifier Fundamentals

Scilab code Exa 1.1.a Amplifier Fundamentals

```
1 //Example 1.1(a)
2
3 clear;
4
5 clc;
6
7 Ri=100*10^3;//Input Resistance
8
9 Aoc=100;//Open Circuit Gain
10
11 Ro=1;//Output Resistance
12
13 Rs=25*10^3;//Source Resistance
14
15 RL=3;//Load Resistance
16
17 Av=(Ri/(Rs+Ri))*Aoc*(RL/(Ro+RL));//Overall Gain
18
19 Vredin=(Ri/(Ri+Rs))*100;//Percentage Reduction in
```

```
    Source Voltage due to Input Loading
20
21 Vredo=(RL/(Ro+RL))*100;//Percentage Reduction in
    Output Voltage due to output loading
22
23 printf("Overall Gain (Av)=%.2f V/V",Av);
24
25 printf("\nPercentage Input Loading=%.2f",Vredin);
26
27 printf("\nPercentage Output Loading=%.2f",Vredo);
```

Scilab code Exa 1.1.b Amplifier Fundamentals

```
1 //Example 1.1(b)
2
3 clear;
4
5 clc;
6
7 Ri=100*10^3;//Input Resistance
8
9 Aoc=100;//Open Circuit Gain
10
11 Ro=1;//Output Resistance
12
13 Rs=50*10^3;//Source Resistance
14
15 RL=4;//Load Resistance
16
17 Av=(Ri/(Rs+Ri))*Aoc*(RL/(Ro+RL));//Overall Gain
18
19 Vredin=(Ri/(Ri+Rs))*100;//Percentage Reduction in
    Source Voltage due to Input Loading
20
21 Vredo=(RL/(Ro+RL))*100;//Percentage Reduction in
```

```
    Output Voltage due to output loading
22
23 printf("Overall Gain (Av)=%.2f V/V",Av);
24
25 printf("\nPercentage Input Loading=%.2f (Not
    mentioned in book)",Vredin);
26
27 printf("\nPercentage Output Loading=%.2f (Not
    mentioned in book)",Vredo);
```

Scilab code Exa 1.2.a Gain of a Noninverting OP AMP

```
1 //Example 1.2(a)
2
3 clear;
4
5 clc;
6
7 Vi=1;//Input Voltage
8
9 R1=2*10^3;
10
11 R2=18*10^3;
12
13 a=10^2;//Open Loop Gain
14
15 A=(1+(R2/R1))*(1+(1+(R2/R1))/a)^(-1);//Overall Gain
16
17 Vo=Vi*A;//Output Voltage
18
19 printf("Output Voltage (Vo)=%.3f V",Vo);
```

Scilab code Exa 1.2.b Gain of a Noninverting OP AMP

```
1  //Example 1.2(b)
2
3  clear;
4
5  clc;
6
7  Vi=1;//Input Voltage
8
9  R1=2*10^3;
10
11 R2=18*10^3;
12
13 a=10^4;//Open Loop Gain
14
15 A=(1+(R2/R1))*(1+(1+(R2/R1))/a)^(-1);//Overall Gain
16
17 Vo=Vi*A;//Output Voltage
18
19 printf("Output Voltage (Vo)=%.3f V",Vo);
```

Scilab code Exa 1.2.c Gain of a Noninverting OP AMP

```
1  //Example 1.2(c)
2
3  clear;
4
5  clc;
6
7  Vi=1;//Input Voltage
8
9  R1=2*10^3;
10
11 R2=18*10^3;
12
13 a=10^6;//Open Loop Gain
```

```
14
15 A=(1+(R2/R1))*(1+(1+(R2/R1))/a)^(-1);//Overall Gain
16
17 Vo=Vi*A;//Output Voltage
18
19 printf("Output Voltage (Vo)=%.4f V",Vo);
```

Scilab code Exa 1.3 Inverting Amplifier Ideal Closed Loop Characterstics

```
1 //Example 1.3
2
3 clear;
4
5 clc;
6
7 R1=10*10^3;
8
9 R2=100*10^3;
10
11 Ri=R1;//Input Resistance
12
13 Ro=0;//Output Resistance
14
15 A=-(R2/R1);// Ideal Overall Gain
16
17 printf("Ri=%.2f kohms",(Ri/1000));
18
19 printf("\nRo=%.f ohms",Ro);
20
21 printf("\nA=%.2f V/V",A);
```

Scilab code Exa 1.4 Designing Summing Amplifiers

```
1 //Example 1.4
2
3 clear;
4
5 clc;
6
7 Rf=120*10^3;//Assuming feedback resistance Rf
     =120*10^3
8 //Imposing in equation Vo=-((Rf/R1)V1+(Rf/R2)V2+(RF/
     R3)V3)
9
10 R1=Rf/6;//From coefficient of V1
11
12 R2=Rf/8;//From coefficient of V2
13
14 R3=Rf/4;//From coefficient of V3
15
16 printf("Designed Summing Amplifier :");
17
18 printf("\n R1=%.2f kohms",(R1/1000));
19
20 printf("\n R2=%.2f kohms",(R2/1000));
21
22 printf("\n R3=%.2f kohms",(R3/1000));
23
24 printf("\n Rf=%.2f kohms",(Rf/1000));
```

Scilab code Exa 1.5 Designing Function Generators

```
1 //Example 1.5
2
3 clear;
4
5 clc;
6
```

```
7  Rf=100*10^3;//Assuming Feedback Resistance Rf
8
9  Vee=-15;
10
11 //Imposing Vo=-(Rf/R1)Vi-(Rf/R2)(-15)=-10Vi+5
12
13 R1=Rf/10;
14
15 R2=(Rf*15)/5;
16
17 printf("Designed Function Generator :");
18
19 printf("\n R1=%.2f kohms",(R1/1000));
20
21 printf("\n R2=%.2f kohms",(R2/1000));
22
23 printf("\n Rf=%.2f kohms",(Rf/1000));
```

Scilab code Exa 1.6 Designing Difference Amplifier

```
1  //Example 1.6
2
3  clear;
4
5  clc;
6
7  Ri1=100*10^3;
8
9  Ri2=100*10^3;
10
11 //Using standard equation for difference amplifier
12 //vo=(R2/R1)(((1+(R1/R2))/(1+(R3/R4)))v2-v1)=v2-3v1
13 //Ri1=R1, Ri2=R3+R4, Ro=0
14
15 R1=Ri1;
```

```
16
17 R2=3*R1;
18
19 //Solving the equations R3+R4=Ri2=100kohms and
     3[(1+(1/3))/(1+(R3/R4))]=1
20 //3[4/3((R3+R4)/R4)]=1
21 //As R3+R4=100 -> 4/(100/R4)=1 -> (4R4)/100=1 -> R4
     /25=1 -> R4=25kohms
22
23 R4=25*10^3;//By solving the equations mentioned
     above
24
25 R3=Ri2-R4;//From standard equations
26
27 printf("Designed Difference Amplifier :");
28
29 printf("\nR1=%.2f kohms",(R1/1000));
30
31 printf("\nR2=%.2f kohms",(R2/1000));
32
33 printf("\nR3=%.2f kohms",(R3/1000));
34
35 printf("\nR4=%.2f kohms",(R4/1000));
```

Scilab code Exa 1.7.a Application of Negative Feedback

```
1 //Example 1.7(a)
2
3 clear;
4
5 clc;
6
7 //1.7(a)
8 Gerrormax=0.1;// Maximum Gain Error Percentage
9
```

```
10 //But Gerror100/T ->Gerrormax=100/Tmin -> Tmin=100/
   Gerrormax
11
12 Tmin=100/Gerrormax;
13
14 printf("Loop Gain (T)>=%.2f",Tmin);
```

Scilab code Exa 1.7.b Application of Negative Feedback

```
1 //Example 1.7(b)
2
3 clear;
4
5 clc;
6
7 Gerrormax=0.1;// Maximum Gain Error Percentage
8
9 //But Gerror100/T ->Gerrormax=100/Tmin -> Tmin=100/
   Gerrormax
10
11 Tmin=100/Gerrormax;
12
13 Aideal=100;
14
15 b=1/Aideal;//Feedback Factor
16
17 amin=Tmin/b;//Minimum Open Loop Gain
18
19 printf("\nOpen Loop Gain (a)>=%.f",amin);
```

Scilab code Exa 1.7.c Application of Negative Feedback

```
1 //Example 1.7(c)
```

```
2
3 clear;
4
5 clc;
6
7 Gerrormax=0.1;// Maximum Gain Error Percentage
8
9 //But Gerror100/T ->Gerrormax=100/Tmin -> Tmin=100/
    Gerrormax
10
11 Tmin=100/Gerrormax;
12
13 Aideal=100;
14
15 b=1/Aideal;//Feedback Factor
16
17 amin=Tmin/b;//Minimum Open Loop Gain
18
19 //Imposing A=a/(1+ab).We have a=10^5 and Aideal=100
    ->100=10^5/(1+10^5b)
20
21 y=poly(0,'x');
22
23 z=(100*amin)*y+(100-amin);//Solving the equation
    mentioned in above comment.
24
25 b=roots(z);
26
27 printf("Feedback Factor (b) for A=100 is =%.5f",b);
```

Scilab code Exa 1.8.a Calculating Gain Desensitivity

```
1 //Example 1.8(a)
2
3 clear;
```

```
4
5 clc;
6
7 a=10^5;//Open Loop Gain
8
9 b=10^(-3);//Feedback Factor
10
11 T=a*b;//return ratio or loop gain
12
13 d=1+T;//Desensitivity Factor
14
15 A=a/d;//Overall Gain
16
17 anew=a+(10/100)*a;//Increasing gain by 10%
18
19 Tnew=anew*b;//New return ratio or loop gain
20
21 dnew=1+Tnew;//New Desensitivity Factor
22
23 Anew=anew/dnew;//Overall Gain
24
25 Achange=((Anew-A)/A)*100;//Percentage Change in
     Overall Gain
26
27 printf("Percentage change in A =%.1f percent",
     Achange);
```

Scilab code Exa 1.8.b Calculating Gain Desensitivity

```
1 //Example 1.8(b)
2
3 clear;
4
5 clc;
6
```

```
7  a=10^5;//Open Loop Gain
8
9  b=1;//Feedback Factor
10
11 T=a*b;//return ratio or loop gain
12
13 d=1+T;//Desensitivity Factor
14
15 aperchange=10;//Percentage Change in a
16
17 Achange=(1/(1+T))*aperchange;//Percentage Change in
       Overall Gain
18
19 printf("Percentage change in A =%.4f",Achange);
```

Scilab code Exa 1.9.a Noninverting Configuration Characterstics

```
1  //Example 1.9(a)
2
3  clear;
4
5  clc;
6
7  rd=2*10^6;//Input Resistance
8
9  ro=75;//Output Resistance
10
11 a=200*10^3;//Open loop Gain
12
13 R1=1*10^3;
14
15 R2=999*10^3;
16
17 b=R1/(R1+R2);//Feedback Factor
18
```

```
19 T=a*b;//reteurn ratio or loop gain
20
21 Aapprox=(1+(R2/R1))*(1/(1+(1/T)))//Approximate Gain
22
23 Riapprox=rd*(1+T);//Approximate Input Resistance
24
25 Roapprox=ro/(1+T);
26
27 Anum=((1+(R2/R1))*a)+(ro/rd)//Numerator of exact
     Gain
28
29 Aden=1+a+(R2/R1)+((R2+ro)/rd)+(ro/R1);//Denominator
     of exact Gain
30
31 Aexact=Anum/Aden;//exact Gain
32
33 Ri1=rd*(1+(a/(1+((R2+ro)/R1))));
34
35 Ri2=(R1*(R2+ro))/(R1+R2+ro);
36
37 Riexact=Ri1+Ri2;//Exact Input Resistance
38
39 Ronum=ro;
40
41 Roden=1+((a+(ro/R1)+(ro/rd))/(1+(R2/R1)+(R2/rd)))
42
43 Roexact=Ronum/Roden;//Exact Output Resistance
44
45 //Ideal Value of input resistance Ri1 is infinity
     and ideal value of output resistance Ro1 is 0.
46
47 printf("Exact Value of A is =%.2f V/V",Aexact);
48
49 printf("\nApproximate Value of A is =%.3f V/V",
     Aapprox);
50
51 printf("\nIdeal Value of A is =%.3f V/V",1000);
52
```

```
53 printf("\nExact Value of Ri is =%.3f Mohms",Riexact
   /10^6);
54
55 printf("\nApproximate Value of Ri is =%.3f Mohms",
   Riapprox/10^6);
56
57 printf("\nIdeal Value of Ri is infinity");
58
59 printf("\nExact Value of Ro is =%.3f mohms",Roexact
   *10^3);
60
61 printf("\nApproximate Value of Ro is =%.3f mohms",
   Roapprox*10^3);
62
63 printf("\nApproximate Value of Ro is =%.3f ohms",0);
```

Scilab code Exa 1.9.b Noninverting Configuration Characterstics

```
1 //Example 1.9(b)
2
3 clear;
4
5 clc;
6
7 rd=2*10^6;//Input Resistance
8
9 ro=75;//Output Resistance
10
11 a=200*10^3;//Open loop Gain
12
13 printf("Note (as mentioned in the book): Because of
   much larger value, we simply ignore the exact
   calculations and use only the approximations.");
14
15 //R12=infinity
```

```
16
17 R2=0;
18
19 //b2=R12/(R12+R22) (Feedback Factor)will be equal to
       1 as R12 tends to infinity and R22 is 0
20
21 b=1;//Feedback Factor
22
23 T=a*b;//reteurn ratio or loop gain
24
25 //Aapprox=(1+(R22/R12))*(1/(1+(1/T2)))(Approximate
     Gain) but R22/R12=0
26
27 Trec=1/T;
28
29 Aden=(1+Trec);
30
31 Anum=1;
32
33 Aapprox=Anum/Aden;//Approximate Gain
34
35 Riapprox=rd*(1+T);//Approximate Input Resistance
36
37 Roapprox=ro/(1+T);//Approximate Output Resistance
38
39 //Ideal Value of input resistance Ri2 is infinity
     and ideal value of output resistance Ro2 is 0.
40
41 printf("\nApproximate Value of A is =%.f V/V",
     Aapprox);
42
43 printf("\nIdeal Value of A is =%.2f V/V",1);
44
45 printf("\nApproximate Value of Ri is =%.3f Gohms",
     Riapprox/10^9);
46
47 printf("\nIdeal Value of Ri is infinity");
48
```

```
49 printf("\nApproximate Value of Ro is =%.3f uohms",
      Roapprox*10^6);
50
51 printf("\nApproximate Value of Ro is =%.f ohms",0);
```

Scilab code Exa 1.10.a Inverting Configuration Characterstics

```
1  //Example 1.10(a)
2
3  clear;
4
5  clc;
6
7  R1=100*10^3;
8
9  R2=100*10^3;
10
11 ro=75;//Output Resistance
12
13 a=200*10^3;//Open Loop Gain for ic741
14
15 b=R1/(R1+R2);//Feedback Factor
16
17 T=a*b;//Loop Gain
18
19 Trec=1/T;
20
21 Atemp=1+Trec;
22
23 Atempr=1/Atemp;
24
25 A=(-R2/R1)*Atempr;//Gain
26
27 Rnnum=R2+ro;
28
```

```
29 Rnden=1+a;
30
31 Rn=Rnnum/Rnden;//Equivalent Resistance of the
      inverting input(Calculation Mistake in the book
      as a is taken as 10^5 rather than 2*10^5)
32
33 Ri=R1+Rn;//Equivalent Input Resistance
34
35 Ro=ro/(1+T);//Equivalent Output Resistance
36
37 printf("A=%.5f V/V",A);
38
39 printf("\nRn=%.2f ohms ",Rn);//answer in textbook is
       wrong
40
41 printf("\nRi=%.2f kohms",(Ri/1000));
42
43 printf("\nRo=%.2f mohms",(Ro*1000));
```

Scilab code Exa 1.10.b Inverting Configuration Characterstics

```
1 //Example 1.10(b)
2
3 clear;
4
5 clc;
6
7 R1=1*10^3;
8
9 R2=1*10^6;
10
11 ro=75;//Output Resistance
12
13 a=200*10^3;//Open Loop Gain for ic741
14
```

```
15 b=R1/(R1+R2);//Feedback Factor
16
17 T=a*b;//Loop Gain
18
19 Trec=1/T;
20
21 Atemp=1+Trec;
22
23 Atempr=1/Atemp;
24
25 A=(-R2/R1)*Atempr;//Gain
26
27 Rnnum=R2+ro;
28
29 Rnden=1+a;
30
31 Rn=Rnnum/Rnden;//Equivalent Resistance of the
      inverting input(Calculation Mistake in the book
      as a is taken as 10^5 rather than 2*10^5)
32
33 Ri=R1+Rn;//Equivalent Input Resistance
34
35 Ro=ro/(1+T);//Equivalent Output Resistance
36
37 printf("A=%.5f V/V",A);
38
39 printf("\nRn=%.2f ohms",Rn);
40
41 printf("\nRi=%.3f kohms",(Ri/1000));
42
43 printf("\nRo=%.3f ohms",Ro);
```

Scilab code Exa 1.11.a Finding the Loop Gain

```
1 //Example 1.11(a)
```

```
2
3 clear;
4
5 clc;
6
7 R1=1*10^6;
8
9 R2=1*10^6;
10
11 R3=100*10^3;
12
13 R4=1*10^3;
14
15 RL=2*10^3;//Load Resistance
16
17 A=-(R2/R1)*(1+(R3/R2)+(R3/R4));//Ideal Gain
18
19 printf("Ideal Gain of of the op amp (A)=%.2f V/V",A)
   ;
```

Scilab code Exa 1.11.b Finding the Loop Gain

```
1 //Example 1.11(b)
2
3 clear;
4
5 clc;
6
7 R1=1*10^6;
8
9 R2=1*10^6;
10
11 R3=100*10^3;
12
13 R4=1*10^3;
```

```
14
15 RL=2*10^3;//Load ResistanceR1=1*10^6;
16
17 R2=1*10^6;
18
19 R3=100*10^3;
20
21 R4=1*10^3;
22
23 RL=2*10^3;//Load Resistance
24
25 A=-(R2/R1)*(1+(R3/R2)+(R3/R4));//Ideal Gain
26
27 rd=1*10^6;//Internal input resistance
28
29 a=10^5;//Open Loop Gain
30
31 ro=100;
32
33 RA=(R1*rd)/(R1+rd);
34
35 RB=RA+R2;
36
37 RC=(RB*R4)/(RB+R4);
38
39 RD=RC+R3;
40
41 RE=(RD*RL)/(RD+RL);
42
43 RF=RE+ro;
44
45 c1=-(RA/RB);//vD=c1*v1
46
47 c2=(RC/RD);//v1=c2*vo
48
49 c3=(RE/RF);//vo=c3*vT
50
51 c4=a*(c1*c2*c3);//vR=a*vD=a*(c1*v1)=a*(c1*c2*vo)=a*(
```

```
    c1*c2*c3)vT=c4*vT -> vR=c4*vT
52
53 T=-c4;//T=(-vR/vT)=-c4 (Loop Gain)
54
55 Trec=1/T;
56
57 Atemp=1+Trec;
58
59 Aactual=A/Atemp;//Actual Gain
60
61 Adev=((Aactual-A)/A)*100;//Deviation in Gain
62
63 printf("Actual Gain of op amp=%.1f V/V",Aactual);
64
65 printf("\nPercentage Departure of Actual Gain from
    Ideal gain=%.2f",Adev);
```

Scilab code Exa 1.12.a Feedback Factor for Negative Feedback

```
1 //Example 1.12(a)
2
3 clear;
4
5 clc;
6
7 rd=1*10^6;//Internal Input Resistance
8
9 a=10^4;//Open Loop Gain
10
11 ro=100;//Internal Output Resistance
12
13 R1=10*10^3;//shown in Fig. 1.34a
14
15 R2=20*10^3;//shown in Fig. 1.34a
16
```

```
17 R3=30*10^3;//shown in Fig. 1.34a
18
19 R4=300*10^3;//Feedback Resistance (shown in Fig.
     1.34a)
20
21 RL=2*10^3;//Load Resistance
22
23 RArec=((1/R1)+(1/R2)+(1/R3)+(1/rd))//Reciprocal of
     RA(parallel combination of R1, R2, R3 and rd)
24
25 RA=1/RArec;
26
27 RB=RA+R4;
28
29 RC=(RB*RL)/(RB+RL);
30
31 RD=RC+ro;
32
33 c1=(RA/RB);//vN=c1*vo
34
35 c2=(RC/RD);//vo=c2*vT
36
37 b=c1*c2;//Feedback Factor b=vN/vT=c1*c2
38
39 T=a*b;//Loop Gain
40
41 printf("b=%.3f V/V",b);
42
43 printf("\nT=%.2f",T);
```

Scilab code Exa 1.12.b Feedback Factor for Negative Feedback

```
1 //Example 1.12(b)
2
3 clear;
```

```
4
5  clc;
6
7  rd=1*10^6;//Internal Input Resistance
8
9  a=10^4;//Open Loop Gain
10
11 ro=100;//Internal Output Resistance
12
13 R1=10*10^3;//shown in Fig. 1.34a
14
15 R2=20*10^3;//shown in Fig. 1.34a
16
17 R3=30*10^3;//shown in Fig. 1.34a
18
19 R4=300*10^3;//Feedback Resistance (shown in Fig.
     1.34a)
20
21 RL=2*10^3;//Load Resistance
22
23 RArec=((1/R1)+(1/R2)+(1/R3)+(1/rd))//Reciprocal of
     RA(parallel combination of R1, R2, R3 and rd)
24
25 RA=1/RArec;
26
27 RB=RA+R4;
28
29 RC=(RB*RL)/(RB+RL);
30
31 RD=RC+ro;
32
33 c1=(RA/RB);//vN=c1*vo
34
35 c2=(RC/RD);//vo=c2*vT
36
37 b=c1*c2;//Feedback Factor b=vN/vT=c1*c2
38
39 T=a*b;//Loop Gain
```

```
40
41 //1.12(b)
42
43 p1=-(R4/R1);
44
45 p2=-(R4/R2);
46
47 p3=-(R4/R3);
48
49 //vo(ideal)=p1*v1+p2*v2+p3*v3
50
51 Trec=1/T;
52
53 ctempr=1+Trec;
54
55 ctemp=1/ctempr;
56
57 p1act=-(R4/R1)*ctemp;
58
59 p2act=-(R4/R2)*ctemp;
60
61 p3act=-(R4/R3)*ctemp;
62
63 printf("Ideal Transfer Characterstic of the circuit
     vo=-(%.2f*v1",-p1);
64
65 printf("+%.2f*v2",-p2);
66
67 printf("+%.2f*v3)",-p3);
68
69 printf("\nActual Transfer Characterstic of the
     circuit vo=-(%.2f*v1",-p1act);
70
71 printf("+%.2f*v2",-p2act);
72
73 printf("+%.2f*v3)",-p3act);
```

Scilab code Exa 1.13 Feedback Factor for Combination of Negative and Positive Feedback

```
1  //Example 1.13
2
3  clear;
4
5  clc;
6
7  R1=30*10^3;//From Fig. 1.13b
8
9  R3=20*10^3;//Feedback Resistance obtained from Fig.
       1.13b
10
11 R2=10*10^3;//Load Resistance obtained from Fig. 1.13
       b
12
13 rd=100*10^3;//Internal Input Resistance
14
15 ro=100;//Internal Output Resistance
16
17 bNnum=((R1*rd)/(R1+rd))+R3;
18
19 bNden=ro+R2+bNnum;
20
21 bN=bNnum/bNden;
22
23 bPnum=R3;
24
25 bPden=bNden;
26
27 bP=bPnum/bPden;
28
29 b=bN-bP;//Feedback Factor
```

```
30
31 printf("b=%.3f V/V",b);
```

Scilab code Exa 1.14.a Current Flow and Power Dissipation

```
1  //Example 1.14(a)
2
3  clear;
4
5  clc;
6
7  R1=10*10^3;
8
9  R2=20*10^3;
10
11 RL=2*10^3;///Load Resistance
12
13 vI=3;//Input Voltage
14
15 IQ=0.5*10^(-3);
16
17 vO=-(R2/R1)*vI;//Output Voltage
18
19 iL=-vO/RL;//Current through RL
20
21 i1=vI/R1;//Cuurent through R1
22
23 i2=i1;//Current through R2 (as current sunk by the
     op amp is 0)
24
25 iO=i2+iL;//Output Current
26
27 iCC=IQ;
28
29 iEE=iCC+iO;
```

```
30
31 printf("iCC=%.2f mA",(iCC*1000));
32
33 printf("\niEE=%.2f mA",(iEE*1000));
34
35 printf("\niO=%.2f mA",(iO*1000));
```

Scilab code Exa 1.14.b Current Flow and Power Dissipation

```
1  //Example 1.14(b)
2
3  clear;
4
5  clc;
6
7  R1=10*10^3;
8
9  R2=20*10^3;
10
11 RL=2*10^3;///Load Resistance
12
13 vI=3;//Input Voltage
14
15 IQ=0.5*10^(-3);
16
17 vO=-(R2/R1)*vI;//Output Voltage
18
19 iL=-vO/RL;//Current through RL
20
21 i1=vI/R1;//Cuurent through R1
22
23 i2=i1;//Current through R2 (as current sunk by the
     op amp is 0)
24
25 iO=i2+iL;//Output Current
```

```
26
27 iCC=IQ;
28
29 iEE=iCC+iO;
30
31 VCC=15;
32
33 VEE=-15;
34
35 IQ=0.5*10^(-3);
36
37 pOA=(VCC-VEE)*IQ+(vO-VEE)*iO;//Power Dissipated in
      the Op Amp
38
39 printf("Power Dissipated inside the op amp=%.2f mW"
      ,(pOA*1000));
```

Scilab code Exa 1.15.a Designing variable dc source

```
1 //Example 1.15(a)
2
3 clear;
4
5 clc;
6
7 Rp=100*10^3;//Potentiometer Resistance
8
9 VCC=15;
10
11 VEE=-15;
12
13 //We have to choose the resistances in such a way
      that we get VA=10V and VB=-10V, so that if we
      want the source to be in the range -10V<=vW<=10V,
       we need to only turn the wiper. Let RA and RB be
```

```
    the resistances corresponding to nodes A and B
    respectively. If RA=RB=25kohm then there would be
    a drop of 5V accross each component(RA,RB and
    potentiometer) which will make VA=10V and VB=-10V
    . Hence RA and RB are selected as 25kohms.(Refer
    Fig. 1.38)
14
15 //vRA(voltage accross RA)=5=(15*RA)/(50+RA) (Using
    Voltade Divider Rule)where 50kohm is the
    potentiometer  resistance on node A side and RA
    is in kohms. Hence by solving the equation RA=25
    kohm. Similarly solve for RB.
16
17 y=poly(0,'x');
18
19 p=5*(y+50*(10^3))-(15*y);
20
21 RA=roots(p);
22
23 RB=RA;
24
25 printf("Designed Source :");
26
27 printf("\nRA=%.2f kohms",(RA/1000));//mentioned in
    the diagram
28
29 printf("\nRB=%.2f kohms",(RB/1000));//mentioned in
    the diagram
30
31 printf("\nRpot=%.2f kohms",(Rp/1000));//mentioned in
     the diagram
```

Scilab code Exa 1.15.b Designing variable dc source

```
1 //Example 1.15(b)
```

```
2
3  clear;
4
5  clc;
6
7  Rp=100*10^3;//Potentiometer Resistance
8
9  VCC=15;
10
11 VEE=-15;
12
13 //We have to choose the resistances in such a way
      that we get VA=10V and VB=-10V, so that if we
      want the source to be in the range -10V<=vW<=10V,
       we need to only turn the wiper. Let RA and RB be
       the resistances corresponding to nodes A and B
      respectively. If RA=RB=25kohm then there would be
      a drop of 5V accross each component(RA,RB and
      potentiometer) which will make VA=10V and VB=-10V
      . Hence RA and RB are selected as 25kohms.(Refer
      Fig. 1.38)
14
15 //vRA(voltage accross RA)=5=(15*RA)/(50+RA) (Using
      Voltade Divider Rule)where 50kohm is the
      potentiometer  resistance on node A side and RA
      is in kohms. Hence by solving the equation RA=25
      kohm. Similarly solve for RB.
16
17 y=poly(0,'x');
18
19 p=5*(y+50*(10^3))-(15*y);
20
21 RA=roots(p);
22
23 RB=RA;
24
25 RL=1*10^3;//Load Resistance
26
```

```
27 vS=10;//Source voltage
28
29 iL=vS/RL;//Current drawn by the load
30
31 a=200*10^3;//Open Loop Gain (defined for 741)
32
33 b=1;//Feedback Factor (Refer Fig. 1.38)
34
35 T=a*b;//Loop Gain
36
37 ro=75;//Internal Output Resistance (defined for 741)
38
39 Ro=ro/(1+T);//Ouput Resistance
40
41 vSchange=Ro*iL;//Change in Voltage
42
43 printf("Change in Vs=%.3f uV",(vSchange*10^6));
```

Scilab code Exa 1.16 Inverting Amplifier driven into saturation

```
1 //Example 1.16
2
3 clear;
4
5 clc;
6
7 //Part(I)
8 //This part of the program includes the plotting of
    the input wave (triangular wave). To plot the
    wave we have divided the time period(assuming T
    =2) into 3 time intervals t1, t2, t3 and then
    create voltage equation for each using the given
    conditions.
```

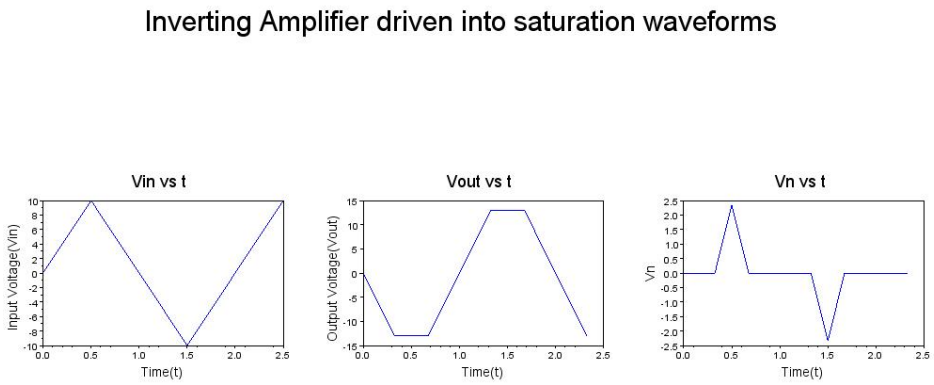

Figure 1.1: Inverting Amplifier driven into saturation

```
9
10 VCC=13;
11
12 VEE=-13;
13
14 A=-2;//Gain
15
16 t1=[0:10^(-4):0.5];
17
18 t2=[0.5:10^(-4):1.5];
19
20 t3=[1.5:10^(-4):2.5];
21
22 vt1=20*t1;
23
24 vt2=20*(1-t2);
25
26 vt3=20*(t3-2);
27
```

```
28 subplot(131);
29
30 title("
      Inverting Amplifier driven into saturation
      waveforms ","fontsize",6);
31
32 subplot(334);
33
34 plot(t1,vt1);
35
36 plot(t2,vt2);
37
38 plot(t3,vt3);
39
40 xlabel("Time(t)","fontsize",3);
41
42 ylabel("Input Voltage(Vin)","fontsize",3);
43
44 title("Vin vs t","fontsize",4);
45
46 //Part(II)
47 //In this part we have plotted vo by using the
      conditions vo=-2vI for -6.5V<vI<6.5V, otherwise
      vo=-13.Again we have divided the time period into
       5 parts to1, to2, to3, to1i, to2i depending upon
       the response in each interval.
48
49 vIbor=VCC/2;
50
51 to1min=0;
52
53 to1max=6.5/20;
54
55 to2min=1-(6.5/20);
56
57 to2max=1+(6.5/20);
58
```

```
59 to3min=2-(6.5/20);
60
61 to3max=2+(6.5/20);
62
63 to1=[to1min:10^(-4):to1max];
64
65 to2=[to2min:10^(-4):to2max];
66
67 to3=[to3min:10^(-4):to3max];
68
69 to1imin=to1max;
70
71 to1imax=to2min;
72
73 to2imin=to2max;
74
75 to2imax=to3min;
76
77 to1i=[to1imin:10^(-4):to1imax];
78
79 to2i=[to2imin:10^(-4):to2imax];
80
81 vo1=-13*(to1-to1min)/(to1max-to1min);
82
83 vo1i=-13;
84
85 vo2=(((13+13)/(to2max-to2min))*(to2-to2min))-13;
86
87 vo2i=13;
88
89 vo3=(((13+13)/(to3min-to3max))*(to3-to3max))-13;
90
91 subplot(335);
92
93 plot(to1,vo1);
94
95 plot(to1i,vo1i);
96
```

```
97  plot(to2,vo2);
98
99  plot(to2i,vo2i);
100
101 plot(to3,vo3);
102
103 ylabel("Output Voltage(Vout)","fontsize",3);
104
105 xlabel("Time(t)","fontsize",3);
106
107 title("Vout vs t","fontsize",4);
108
109 //Part(III)
110 //In this part we will plot vN for which we have
      divided the time period into 7 time intervals
      tNi1, tNi2, tNi3, tN11, tN12, tN21, tN22
      depending upon the response in each cycle voltage
       equation is obtained and plotted.For -6.5<vI<6.5
       vN=0 and when vI will peak at 10V vN will peak
      at 2.33 and for vI<6-.5 and vI>6.5, circuit
      behaviour is symmetric.
111
112 vIbor=VCC/2;
113
114 tNi1min=0;
115
116 tNi1max=6.5/20;
117
118 tNi2min=1-(6.5/20);
119
120 tNi2max=1+(6.5/20);
121
122 tNi3min=2-(6.5/20);
123
124 tNi3max=2+(6.5/20);
125
126 tNi1=[tNi1min:10^(-4):tNi1max];
127
```

```
128 tNi2=[tNi2min:10^(-4):tNi2max];
129
130 tNi3=[tNi3min:10^(-4):tNi3max];
131
132 tN11min=tNi1max;
133
134 tN11max=(tNi2min+tNi1max)/2;
135
136 tN12min=tN11max;
137
138 tN12max=tNi2min;
139
140 tN21min=tNi2max;
141
142 tN21max=(tNi2max+tNi3min)/2;
143
144 tN22min=tN21max;
145
146 tN22max=tNi3min;
147
148 tN11=[tN11min:10^(-4):tN11max];
149
150 tN12=[tN12min:10^(-4):tN12max];
151
152 tN21=[tN21min:10^(-4):tN21max];
153
154 tN22=[tN22min:10^(-4):tN22max];
155
156 vNi1=0;
157
158 vN11=(2.33/(tN11max-tN11min))*(tN11-tN11min);
159
160 vN12=-(2.33/(tN12max-tN12min))*(tN12-tN12max);
161
162 vNi2=0;
163
164 vN21=-(2.33/(tN21max-tN21min))*(tN21-tN21min);
165
```

```
166 vN22=(2.33/(tN22max-tN22min))*(tN22-tN22max);
167
168 vNi3=0;
169
170 subplot(336);
171
172 plot(tNi1,vNi1);
173
174 plot(tN11,vN11);
175
176 plot(tN12,vN12);
177
178 plot(tNi2,vNi2);
179
180 plot(tN21,vN21);
181
182 plot(tN22,vN22);
183
184 plot(tNi3,vNi3);
185
186 xlabel("Time(t)","fontsize",3);
187
188 ylabel("Vn","fontsize",3);
189
190 title("Vn vs t","fontsize",4);
```

Chapter 2

Circuits with Resistive Feedback

Scilab code Exa 2.1 Closed Loop Parameters of Basic I V Conveter

```
1  //Example 2.1
2
3  clear;
4
5  clc;
6
7  R=1*10^6;
8
9  a=200*10^3;//Open Loop Gain for ic741
10
11 rd=2*10^6;//defined for 741
12
13 ro=75;//internal output resistance defined for 741
14
15 Tnum=a*rd;
16
17 Tden=rd+R+ro;
18
19 T=Tnum/Tden;//Loop Gain
```

```
20
21 Anum=-R;
22
23 Aden=1+(1/T);
24
25 A=Anum/Aden;//Oveall Gain
26
27 Rinumn=rd*(R+ro);
28
29 Rinumd=rd+R+ro;
30
31 Rinum=Rinumn/Rinumd;
32
33 Riden=1+T;
34
35 Ri=Rinum/Riden;//Input resistance
36
37 Ronum=ro;
38
39 Roden=1+T;
40
41 Ro=Ronum/Roden;//Ouput Resistance (Value obtained
      for Ro in the book is wrong)
42
43 printf("T=%.f",T);
44
45 printf("\nA=%.6f V/uA",A*10^(-6));
46
47 printf("\nRi=%.1f ohms",Ri);
48
49 printf("\nRo=%.3f mohms",(Ro*(10^3)));//answer in
      textbook is wrong
```

Scilab code Exa 2.2 Designing High Sensitivity I V Conveter

```
1  //Example 2.2
2
3  clear;
4
5  clc;
6
7  sen=0.1*10^9;//sensitivity in V/A
8
9  R=1*10^6;//Assumption
10
11 //sen=k*R ->k=sen/R
12
13 k=sen/R;
14
15 R1=1*10^3;//Assumption
16
17 //k=1+(R2/R1)+(R2/R) ->R2=(k-1)/((1/R1)+(1/R))
18
19 R2num=k-1;
20
21 R2den=((1/R1)+(1/R));
22
23 R2=R2num/R2den;
24
25 printf("Designed High Sensitivity I-V Converter :");
26
27 printf("\nR=%.f Mohms",(R*10^(-6)));
28
29 printf("\nR1=%.f kohms",R1*10^(-3));
30
31 printf("\nR2=%.f kohms",R2*10^(-3))
```

Scilab code Exa 2.3.a Characterstics of Floating Load V I Converters

```
1  //Example 2.3(a)
```

```
2
3 clear;
4
5 clc;
6
7 vI=5;//Input Voltage
8
9 R=10*10^3;
10
11 Vsat=13;//Saturation Voltage
12
13 iO=vI/R;//(from right to left for Fig.2.4(a) and
     from left to right for Fig.2.4(b))
14
15 printf("iO=%.1f mA",iO*10^3);
```

Scilab code Exa 2.3.b Characterstics of Floating Load V I Converters

```
1 //Example 2.3(b)
2
3 clear;
4
5 clc;
6
7 vI=5;//Input Voltage
8
9 R=10*10^3;
10
11 Vsat=13;//Saturation Voltage
12
13 iO=vI/R;//iO for Circuit shown in Fig.2.4(a) (from
     right to left)
14
15 //For Circuit shown in Fig.2.4(a) VoL1<vL1<VoH1
16
```

```
17 VoL1=-Vsat-vI;
18
19 VoH1=Vsat-vI;
20
21 printf("For Circuit shown in Fig.2.4(a) %.1f V< vL <
     ",VoL1);
22
23 printf("%.1f V",VoH1);
24
25 //For Circuit shown in Fig.2.4(b) VoL2<vL2<VoH2
26
27 VoL2=-Vsat;
28
29 VoH2=Vsat;
30
31 printf("\nFor Circuit shown in Fig.2.4(b) %.1f V< vL
     <",VoL2);
32
33 printf("%.1f V",VoH2);
```

Scilab code Exa 2.3.c Characterstics of Floating Load V I Converters

```
1 //Example 2.3(c)
2
3 clear;
4
5 clc;
6
7 vI=5;//Input Voltage
8
9 R=10*10^3;
10
11 Vsat=13;//Saturation Voltage
12
13 iO=vI/R;//iO for Circuit shown in Fig.2.4(a) (from
```

```
    right to left)
14
15 //For Circuit shown in Fig.2.4(a)
16
17 VoL1=-Vsat-vI;
18
19 VoH1=Vsat-vI;
20
21 //For Circuit shown in Fig.2.4(b) VoL2<vL2<VoH2
22
23 VoL2=-Vsat;
24
25 VoH2=Vsat;
26
27 RLmax1=VoH1/iO;//Maximum Possible value of RL
28
29 //For Circuit shown in Fig.2.4(b)
30
31 RLmax2=VoH2/iO;//Maximum Possible Value of Rl
32
33 printf("Max Value of RL for Circuit shown in Fig
    .2.4(a)= %.f kohms",RLmax1*10^(-3));
34
35 printf("\nMax Value of RL for Circuit shown in Fig
    .2.4(b)= %.f kohms",RLmax2*10^(-3));
```

Scilab code Exa 2.4 Designing Current Source using Grounded Load Converter

```
1 //Example 2.4
2
3 clear;
4
5 clc;
6
```

```
7  Vsat=15;//Saturation Voltage
8
9  vL=10;
10
11 iL=10^(-3);//Load Current
12
13 vI=Vsat;//Assuming Vsat as the input Voltage
14
15 R1=vI/iL;//(Tolerance -1%)
16
17 //vL<=(R1/(R1+R2))*Vsat, Vsat=15V ->10<=(R1/(R1+R2))
      *15 or 10=(R1/(R1+R2))*13 (approx)
18 //->R2=((13*R1)/vL)-R1
19
20 R2=((13*R1)/vL)-R1;//(Tolerance -1%)
21
22 R3=R1;//(Tolerance -1%)
23
24 R4=R2;//(Tolerance -1%)
25
26 printf("Designed Current Source using Grounded Load
      Converter :");
27
28 printf("\nR1=%.f kohms",R1*10^(-3));
29
30 printf("\nR2=%.1f kohms",R2*10^(-3));
31
32 printf("\nR3=%.f kohms",R3*10^(-3));
33
34 printf("\nR4=%.1f kohms",R4*10^(-3));
```

Scilab code Exa 2.5.a Effect of Resistance Mismatches in Grounded Load Converters

```
1 //Example 2.5(a)
```

```
2
3 clear;
4
5 clc;
6
7 R1=15*10^3;//From the result of Example 2.4
8
9 p=0.01;//For 1% tolerance p=t/100=1/100=0.01
10
11 emax=4*p;//imbalace factor
12
13 Romin=R1/emax;
14
15 printf("Ro can be anywhere in the range Ro>=%.2f
   kohms",Romin*10^(-3));
```

Scilab code Exa 2.5.b Effect of Resistance Mismatches in Grounded Load Converters

```
1 //Example 2.5(b)
2
3 clear;
4
5 clc;
6
7 R1=15*10^3;//From the result of Example 2.4
8
9 p=0.001;//For 1% tolerance p=t/100=1/100=0.01
10
11 emax=4*p;//imbalace factor
12
13 Romin=R1/emax;
14
15 printf("Ro can be anywhere in the range Ro>=%.2f
   Mohms",Romin*10^(-6));
```

Scilab code Exa 2.5.c Effect of Resistance Mismatches in Grounded Load Converters

```
1 //Example 2.5(c)
2
3 clear;
4
5 clc;
6
7 R1=15*10^3;//From the result of Example 2.4
8
9 Romin=50*10^6;
10
11 emax=R1/Romin;
12
13 p=emax/4;
14
15 pper=p*100;
16
17 printf("Resistance tolerance Required=%.5f percent",
     pper);
```

Scilab code Exa 2.6 Howland Circuit Calibration

```
1 //Example 2.6
2
3 clear;
4
5 clc;
6
7 //From result of Example 2.4
```

```
R1=15*10^3;

R2=4.5*10^3;

R3=R1;

R4=R2;

p=0.01;

e=4*p*R1;//Resistance to be trimmed

R3redmax=R3-e;//R3red<=R3redmax

R3red=R3redmax-400;//Tolerance 1%

Rpot=2*(R3-R3red);

printf("Designed Current Source using Grounded Load
   Converter with trimmed R3 :");

printf("\nR1=%.2f kohms",R1*10^(-3));

printf("\nR2=%.2f kohms",R2*10^(-3));

printf("\nRs=%.2f kohms",R3red*10^(-3));

printf("\nRpot=%.2f kohms",Rpot*10^(-3));

printf("\nR4=%.2f kohms",R4*10^(-3));
```

Scilab code Exa 2.7 Effect of finite loop gain on Howland Circuit

```
//Example 2.7

```

```
3  clear;
4
5  clc;
6
7  R1=15*10^3;
8
9  R2=3*10^3;
10
11 R3=R1;
12
13 R4=R2;
14
15 a=200*10^3;
16
17 Ro1num=R1*R2;
18
19 Ro1den=R1+R2;
20
21 Ro1=Ro1num/Ro1den;
22
23 Ro2num=a;
24
25 Ro2den=(1+(R2/R1));
26
27 Ro2=Ro2num/Ro2den;
28
29 Ro=Ro1*(1+Ro2);//Output resistance
30
31 printf("Output Resistance (Ro)=%.f Mohms",Ro*10^(-6)
   );
```

Scilab code Exa 2.8.a Output Voltage of a Difference Amplifier

```
1  //Example 2.8(a)
2
```

```
3  clear;
4
5  clc;
6
7  R1=10*10^3;
8
9  R3=R1;
10
11 R2=100*10^3;
12
13 R4=R2;
14
15 //For first pair of inputs (v1, v2)=(-0.1 V, +0.1V)
16 v11=-0.1;
17
18 v21=0.1;
19
20 vo1=(R2/R1)*(v21-v11);
21
22 vcm1=(v11+v21)/2;
23
24 //For Second pair of inputs (v1, v2)=(4.9 V, 5.1V)
25
26 v12=4.9;
27
28 v22=5.1;
29
30 vo2=(R2/R1)*(v22-v12);
31
32 vcm2=(v12+v22)/2;
33
34 //For Third pair of inputs (v1, v2)=(9.9 V, 10.1 V)
35
36 v13=9.9;
37
38 v23=10.1;
39
40 vo3=(R2/R1)*(v23-v13);
```

```
41
42 vcm3=(v13+v23)/2;
43
44 printf("vo for (-0.1 V,+0.1 V)=%.2f V",vo1);
45
46 printf("\nvo for (4.9 V,5.1 V)=%.2f V",vo2);
47
48 printf("\nvo for (9.9 V,10.1 V)=%.2f V",vo3);
```

Scilab code Exa 2.8.b Output Voltage of a Difference Amplifier

```
1 //Example 2.8(b)
2
3 clear;
4
5 clc;
6
7 R1=10*10^3;
8
9 R2=98*10^3;
10
11 R3=9.9*10^3;
12
13 R4=103*10^3;
14
15 //For first pair of inputs (v1, v2)=(-0.1 V, +0.1V)
16 v11=-0.1;
17
18 v21=0.1;
19
20 vo1=(R2/R1)*(v21-v11);
21
22 vcm1=(v11+v21)/2;
23
24 //For Second pair of inputs (v1, v2)=(4.9 V, 5.1V)
```

```
25
26 v12=4.9;
27
28 v22=5.1;
29
30 vo2=(R2/R1)*(v22-v12);
31
32 vcm2=(v12+v22)/2;
33
34 //For Third pair of inputs (v1, v2)=(9.9 V, 10.1 V)
35
36 v13=9.9;
37
38 v23=10.1;
39
40 vo3=(R2/R1)*(v23-v13);
41
42 vcm3=(v13+v23)/2;
43
44 //vO=A2*v2-A1*v1
45
46 A2num=(1+(R2/R1));
47
48 A2den=(1+(R3/R4));
49
50 A2=A2num/A2den;
51
52 A1=R2/R1;
53
54 //For first pair of inputs (v1, v2)=(-0.1 V, +0.1V)
55
56 vo1m=A2*v21-A1*v11;
57
58 //For Second pair of inputs (v1, v2)=(4.9 V, 5.1V)
59
60 vo2m=A2*v22-A1*v12;
61
62 //For Third pair of inputs (v1, v2)=(9.9 V, 10.1 V)
```

```
63
64 vo3m=A2*v23-A1*v13;
65
66 printf("vo for (-0.1 V,+0.1 V)=%.3f V",vo1m);
67
68 printf("\nvo for (4.9 V,5.1 V)=%.3f V",vo2m);
69
70 printf("\nvo for (9.9 V,10.1 V)=%.3f V",vo3m);
```

Scilab code Exa 2.9 Common Mode Rejection Ratio for op amp

```
1 //Example 2.9
2
3 clear;
4
5 clc;
6
7 R1=10*10^3;
8
9 R3=R1;
10
11 R2=100*10^3;
12
13 R4=R2;
14
15 p=0.01;
16
17 emax=4*p;
18
19 Adm1=R2/R1;
20
21 Adm2n=emax*(R1+2*R2);
22
23 Adm2d=2*(R1+R2);
24
```

```
25 Adm2=1-(Adm2n/Adm2d);
26
27 Admin=Adm1*Adm2;
28
29 Acmax=(R2/(R1+R2))*emax;
30
31 cmrrm=20*log10(Admin/Acmax);
32
33 printf("(a) CMRR min=%.1f dB",cmrrm);
34
35 //2.9(b)
36
37 vdm=0;
38
39 vcm=10;
40
41 vO=vcm*Acmax+vdm*Admin;
42
43 printf("\n(b) Output Error vO=%.3f V",vO);
44
45 //2.9(c)
46
47 //CMRR=20*log((1+(R2/R1))/emax) -> 80=20*log((1+(R2/
   R1))/emax) -> 4=log((1+(R2/R1))/emax) ->10^4=(1+(
   R2/R1))/emax -> emax=10^4/(1+(R2/R1))
48
49 emax1=(1+(R2/R1))/(10^(4));
50
51 p=emax1/4;
52
53 pper=p*100;
54
55 printf("\n(c) Required Resistance Tolerance=%.4f
   percent",pper);
```

Scilab code Exa 2.10.a Designing Triple Op Amp Instrumentation Amplifier

```
1  //Example 2.10(a)
2
3  clear;
4
5  clc;
6
7  Amin=1;
8
9  Amax=10^3;
10
11 AI=0.5;
12
13 R1=100*10^3;//Tolerance (1%)
14
15 R2=AI*R1;//Tolerance (1%)
16
17 AImin=Amin/AI;
18
19 AImax=Amax/AI;
20
21 //AImin<=AI<=AImax
22 //AImin=1+((2*R3)/(R4+R1)) -> 1+((2*R3)/(R4+R1))-
     Amin=0 -> (1-AImin)*R4+2*R3+(1-AImin)*R1=0...(i)
     and AImax=1+((2*R3)/(R4+0)) ->(1-AImax)*R4+2*R3
     =0....(ii)
23 //Solving these two equations will give R3 and R4
24
25 A=[2 (1-AImin);2 (1-AImax)];
26
27 B=[(1-AImin)*R1;0];
28
29 R=linsolve(A,B);
30
31 R3=R(1,1);//Tolerance (1%)
32
```

```
33 R4=R(2,1);//Tolerance (1%)
34
35 printf("Designed Instrumentation Amplifier :");
36
37 printf("\nR1=%.2f kohms",R1*10^(-3));
38
39 printf("\nR2=%.2f kohms",R2*10^(-3));
40
41 printf("\nR3=%.f kohms",R3*10^(-3));
42
43 printf("\nR4=%.f ohms",R4);
```

Scilab code Exa 2.10.b Designing Triple Op Amp Instrumentation Amplifier

```
1 //Example 2.10(b)
2
3 clear;
4
5 clc;
6
7 Amin=1;
8
9 Amax=10^3;
10
11 AI=0.5;
12
13 R1=100*10^3;//Tolerance (1%)
14
15 R2=AI*R1;//Tolerance (1%)
16
17 AImin=Amin/AI;
18
19 AImax=Amax/AI;
20
```

```
21 //AImin<=AI<=AImax
22 //AImin=1+((2*R3)/(R4+R1)) -> 1+((2*R3)/(R4+R1))-
     Amin=0 -> (1-AImin)*R4+2*R3+(1-AImin)*R1=0...(i)
     and AImax=1+((2*R3)/(R4+0)) ->(1-AImax)*R4+2*R3
     =0....(ii)
23 //Solving these two equations will give R3 and R4
24
25 A=[2 (1-AImin);2 (1-AImax)];
26
27 B=[(1-AImin)*R1;0];
28
29 R=linsolve(A,B);
30
31 R3=R(1,1);//Tolerance (1%)
32
33 R4=R(2,1);//Tolerance (1%)
34
35 p=0.01;
36
37 e=4*p*R2;
38
39 R5=100*10^3;
40
41 R2red=R2-e-500;//to be on the safer side 0.5 kohms
     more is reduced
42
43 Rpot=2*(R2-R2red);//Potentiometer Resistance
44
45 //Circuit is shown in Fig.2.21 in the book
46
47 printf("Designed Instrumentation Amplifier with
     trimmed resistances :");
48
49 printf("\nR1=%.2f kohms",R1*10^(-3));
50
51 printf("\nR2=%.2f kohms",R2*10^(-3));
52
53 printf("\nR3=%.f kohms",R3*10^(-3));
```

```
54
55 printf("\nR4=%.f  ohms",R4);
56
57 printf("\nR5=%.f  kohms",R5*10^(-3));
58
59 printf("\nR6=%.2f  kohms",R2red*10^(-3));
60
61 printf("\nR7=%.2f  kohms",Rpot*10^(-3));
```

Scilab code Exa 2.10.c Designing Triple Op Amp Instrumentation Amplifier

```
1  //Example 2.10(c)
2
3  clear;
4
5  clc;
6
7  Amin=1;
8
9  Amax=10^3;
10
11 AI=0.5;
12
13 R1=100*10^3;//Tolerance (1%)
14
15 R2=AI*R1;//Tolerance (1%)
16
17 AImin=Amin/AI;
18
19 AImax=Amax/AI;
20
21 //AImin<=AI<=AImax
22 //AImin=1+((2*R3)/(R4+R1)) -> 1+((2*R3)/(R4+R1))-
     Amin=0 -> (1-AImin)*R4+2*R3+(1-AImin)*R1=0...(i)
```

```
    and AImax=1+((2*R3)/(R4+0)) ->(1-AImax)*R4+2*R3
    =0....(ii)
//Solving these two equations will give R3 and R4

A=[2 (1-AImin);2 (1-AImax)];

B=[(1-AImin)*R1;0];

R=linsolve(A,B);

R3=R(1,1);//Tolerance (1%)

R4=R(2,1);//Tolerance (1%)

//2.10(c)

Rpot1=100*10^3;

printf("To calibrate the circuit, tie the inputs
   together and set the Rpot1 pot for the maximum
   gain (wiper all the way up). Then, while
   switching the common inputs back and forth
   between -5V and +5V, adjust the Rpot2 pot for the
    minimum change at the output.");
```

Scilab code Exa 2.11.a Study of Resistance Temperarure Detector

```
//Example 2.11(a)

clear;

clc;

R0=100;

```

```
alpha=0.00392;

//R(T)=R0*(1+alpha*T) -> R(T)=100*(1+0.00392*T)

printf("R(T)=%.2f",R0);

printf("(1+%.5f",alpha);

printf("T) ohms");
```

Scilab code Exa 2.11.b Study of Resistance Temperarure Detector

```
//Example 2.11(b)

clear;

clc;

R0=100;

alpha=0.00392;

T1=25;

R1=R0*(1+alpha*T1);

printf("R(25 deg Celsius)=%.2f ohms",R1);

T2=100;

R2=R0*(1+alpha*T2);

printf("\nR(100 deg Celsius)=%.2f ohms",R2);

T3=-15;
```

```
24
25 R3=R0*(1+alpha*T3);
26
27 printf("\nR(-15 deg Celsius)=%.2f ohms",R3);
```

Scilab code Exa 2.11.c Study of Resistance Temperarure Detector

```
1  //Example 2.11(c)
2
3  clear;
4
5  clc;
6
7  R0=100;
8
9  alpha=0.00392;
10
11 dT=10;
12
13 delta=alpha*dT;
14
15 deltaper=delta*100;
16
17 dR=R0*delta;
18
19 printf("Change in R=%.2f ohms",dR);
20
21 printf("\nPercentage Deviation=%.2f percent",
     deltaper);
```

Scilab code Exa 2.12.a Designing a Transducer Bridge with Instrumentation Amplifier

```
1  //Example 2.12(a)
2
3  clear;
4
5  clc;
6
7  R0=100;//Data taken from Example 2.11
8
9  alpha=0.00392;//Data taken from Example 2.11
10
11 Vref=15;
12
13 Prtd=0.2*10^(-3);
14
15 i=(Prtd/R0)^(0.5)-(0.41*10^(-3));
16
17 R1=(Vref/i);
18
19 delta=alpha*1;//Fractional Deviation for 1 degree
      celsius change in temperature
20
21 s=0.1;//Output Voltage for 1 degree Celsius
      temperature change
22
23 A1=s*(2+(R1/R0)+(R0/R1));
24
25 A2=Vref*delta;
26
27 A=(A1/A2)+1.0555913;
28
29 printf("R1=%.f kohms",R1*10^(-3));
30
31 printf("\nA=%.1f V/V",A);
```

Scilab code Exa 2.12.b Designing a Transducer Bridge with Instrumentation Amplifier

```
1  //Example 2.12(b)
2
3  clear;
4
5  clc;
6
7  R0=100;//Data taken from Example 2.11
8
9  alpha=0.00392;//Data taken from Example 2.11
10
11 Vref=15;
12
13 Prtd=0.2*10^(-3);
14
15 i=(Prtd/R0)^(0.5)-(0.41*10^(-3));
16
17 R1=(Vref/i);
18
19 delta=alpha*1;//Fractional Deviation for 1 degree
      celsius change in temperature
20
21 s=0.1;//Output Voltage for 1 degree Celsius
      temperature change
22
23 A1=s*(2+(R1/R0)+(R0/R1));
24
25 A2=Vref*delta;
26
27 A=A1/A2;
28
29 dT=100;
30
31 d2=alpha*dT;
32
33 vO1num=A*Vref*d2;
```

```
34
35 vO1den=1+(R1/R0)+((1+(R0/R1))*(1+d2));
36
37 vO1=vO1num/vO1den;
38
39 vO2num=A*Vref*d2;
40
41 vO2den=(2+(R1/R0)+(R0/R1));
42
43 vO2=vO2num/vO2den;
44
45 vOchange=vO2-vO1;
46
47 printf("vO(100 deg Celsius)=%.3f V",vO1);
48
49 Tchange=vOchange/s;
50
51 printf("\nEquivalent Temperature Error=%.2f deg
   Celsius",Tchange);
```

Scilab code Exa 2.13.a Transducer Bridge Calibration

```
1 //Example 2.13(a)
2
3 clear;
4
5 clc;
6
7 R0=100;//Data taken from Example 2.11
8
9 alpha=0.00392;//Data taken from Example 2.11
10
11 Vref=15;
12
13 P=0.2*10^(-3);
```

```
14
15 i=(P/R0)^(0.5)-(0.41*10^(-3));
16
17 pV=0.05;
18
19 Vrefc=pV*Vref+0.25;
20
21 Vrefr=Vref-Vrefc;
22
23 R3=2/(2*i);
24
25 //R0+R1+(R2/2)=Vrefr/i;
26
27 Rtot=Vrefr/i;
28
29 p=0.01;
30
31 R2=(2*p*Rtot)+221.1748472;//220 ohms are added to be
       on the safe side
32
33 R1=(Rtot-(R2/2)-R0)+108.15494;//Tolerance 1%
34
35 v0=9.97;//Data from Example 2.12
36
37 R1u=R1+(R2/2);
38
39 dT=1;//obtained from Example 2.12
40
41 d2=alpha*dT;
42
43 v0=0.1;//Sensitivity (Refer Example 2.12)
44
45 Anum=v0*(2+(R1u/R0)+(R0/R1u));
46
47 Aden=Vrefr*d2;
48
49 A=Anum/Aden;//Overall Gain by using Eq.2.47
50
```

```
51 printf("Designed Circuit for Calibration :");
52
53 printf("\nR1=%.1f kohms",R1*10^(-3));
54
55 printf("\nR2=%.f ohms",R2);
56
57 printf("\nR3=%.f kohms",R3*10^(-3));
58
59 printf("\nA=%.1f V/V",A);
```

Scilab code Exa 2.13.b Transducer Bridge Calibration

```
1 //Example 2.13(a)
2
3 clear;
4
5 clc;
6
7 R0=100;//Data taken from Example 2.11
8
9 alpha=0.00392;//Data taken from Example 2.11
10
11 Vref=15;
12
13 P=0.2*10^(-3);
14
15 i=(P/R0)^(0.5)-(0.41*10^(-3));
16
17 pV=0.05;
18
19 Vrefc=pV*Vref+0.25;
20
21 Vrefr=Vref-Vrefc;
22
23 R3=2/(2*i);
```

```
24
25 //R0+R1+(R2/2)=Vrefr/i;
26
27 Rtot=Vrefr/i;
28
29 p=0.01;
30
31 R2=(2*p*Rtot)+220;//220 ohms are added to be on the
      safe side
32
33 R1=Rtot-(R2/2)-R0;//Tolerance 1%
34
35 vO=9.97;//Data from Example 2.12
36
37 R1u=R1+(R2/2);
38
39 dT=1;//obtained from Example 2.12
40
41 d2=alpha*dT;
42
43 vO=0.1;//Sensitivity (Refer Example 2.12)
44
45 Anum=vO*(2+(R1u/R0)+(R0/R1u));
46
47 Aden=Vrefr*d2;
48
49 A=Anum/Aden;//Overall Gain by using Eq.2.47
50
51 printf("To calibrate, first set T=0 degree Celsius
      and adjust R2 for vO=0 V. Then set T=100 degree
      Celsius and adjust R3 for vO=10.0 V.");
```

Scilab code Exa 2.14 Designing Strain Gauge Bridge with Instrumentation Amplifier

```
1  //Example 2.14
2
3  clear;
4
5  clc;
6
7  //2.14(a)
8
9  Rs=120;
10
11 Vref=15;
12
13 imax=20*10^(-3);
14
15 Vb=2*Rs*imax;
16
17 Vtap=Vb/2;
18
19 Vtapch=0.01*Vtap;
20
21 v1=Vtap+Vtapch;
22
23 v2=Vtap-Vtapch;
24
25 v1ch=v1-v2;
26
27 i=v1ch/((Rs*Rs)/(Rs+Rs));
28
29 R1=(Vtap/i)-630;
30
31 R2=1000;
32
33 i3=2*imax+(4.8/R2);
34
35 R3=(2/i3)+6-0.642857 ;
36
37 R4=((Vref-(R3/2)*i3-Vb)/i3)-3;
38
```

```
39 printf("(a) R1=%.2f kohms",R1*10^(-3));
40
41 printf("\n     R2=%.f kohms",R2*10^(-3));
42
43 printf("\n     R3=%.f ohms",R3);
44
45 printf("\n     R4=%.f ohms",R4);
46
47 //2.14(b)
48
49 printf("\n\n(b) To calibrate, first adjust R2 so
   that with no strain we get vO=o V. Then supply a
   known strain, preferably near the full scale, and
    adjust R3 for the desired value of vO.");
```

Chapter 3

Active Filters Part I

Scilab code Exa 3.1 Pole Zero Response of Transfer Function

```
1  //Example 3.1
2
3  clear;
4
5  clc;
6
7  R=10;
8
9  C=40*10^(-6);
10
11 L=5*10^(-3);
12
13 Hsnum=(R/L)*%s;
14
15 Hsden=((%s^(2))+(R/L)*%s+(1/(L*C)));
16
17 Hs=Hsnum/Hsden;//Transfer Function
18
19 h=syslin('c',Hs);
```

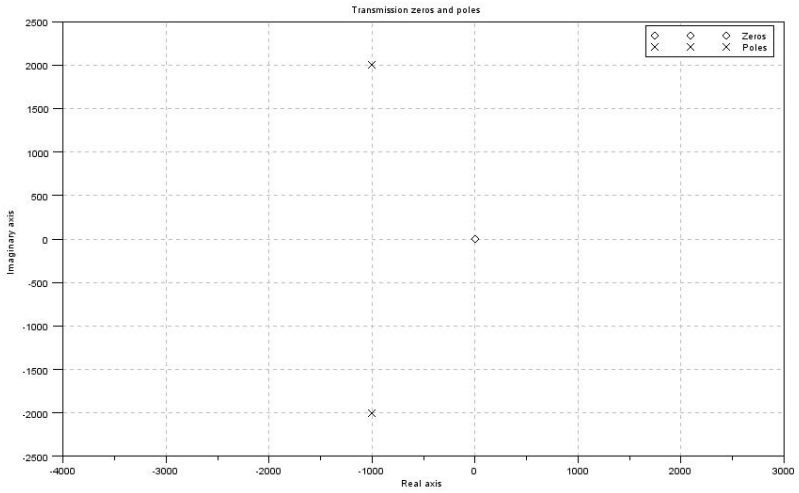

Figure 3.1: Pole Zero Response of Transfer Function

```
20
21 plzr(h);
22
23 zeroes=roots(Hsnum);
24
25 poles=roots(Hsden);
```

Scilab code Exa 3.2 Finding Impulse Response of a given circuit

```
1 //Example 3.2
2
3 clear;
4
5 clc;
6
7 printf("\nThe problem requires to find Laplace
    Transform which is not possible in scilab. Hence
```

```
    standard procedure");
8
9   printf("\nfor finding the Integral Transforms has
    been used")
10
11  syms s t;
12
13  R=10;
14
15  C=40*10^(-6);
16
17  L=5*10^(-3);
18
19  Hsnum=(R/L)*s;
20
21  Hsden=((s^(2))+(R/L)*s+(1/(L*C)));
22
23  Hs=Hsnum/Hsden;//Transfer Function
24
25  vot=ilaplace(Hs);//Impulse Response of Circuit
```

Scilab code Exa 3.3 Steady State Response of a Circuit

```
1   //Example 3.3
2
3   clear;
4
5   clc;
6
7   R=10;
8
9   C=40*10^(-6);
10
```

```
11 L=5*10^(-3);
12
13 s=%i*10^3;
14
15 Hsnum=(R/L)*s;
16
17 Hsden=((s^(2))+(R/L)*s+(1/(L*C)));
18
19 Hs=Hsnum/Hsden;//Transfer Function
20
21 Hsmag=10*abs(Hs);
22
23 Hsphase1=atan(imag(Hs)/real(Hs));
24
25 Hsphase=(Hsphase1*(180/%pi))+45;
26
27 printf("vO(t)=%.3f",Hsmag);
28
29 printf("cos((10^3)t+%.2f",Hsphase);
30
31 printf(") V");
32
33 //vot=Hsmag*cos(10^3*t+Hsphase);
```

Scilab code Exa 3.4.a Low pass filter with Gain

```
1 //Example 3.4(a)
2
3 clear;
4
5 clc;
6
7 dcgaindB=20;//Gain in dB
8
9 dcgain=10^(20/20);
```

```
10
11 f0=10^3;
12
13 //We need R2=dcgain*R1;
14
15 R1approx=20*10^(3);
16
17 R2approx=dcgain*R1approx;
18
19 Capprox=1/(2*%pi*f0*R2approx);
20
21 n=(Capprox*10^9);
22
23 C=Capprox/n;
24
25 R2=(R2approx*n)-1154.9431;
26
27 R1=R2/dcgain;
28
29 printf("Components for achieving the mentioned
     requirements :");
30
31 printf("\nR1=%.1f kohms",R1*10^(-3));
32
33 printf("\nR2=%.f kohms",R2*10^(-3));
34
35 printf("\nC=%.f nF",C*10^9);
```

Scilab code Exa 3.4.b Low pass filter with Gain

```
1 //Example 3.4(b)
2
3 clear;
4
5 clc;
```

```
6
7  dcgaindB=20;//Gain in dB
8
9  dcgain=10^(20/20);
10
11 f0=10^3;
12
13 //We need R2=dcgain*R1;
14
15 R1approx=20*10^(3);
16
17 R2approx=dcgain*R1approx;
18
19 Capprox=1/(2*%pi*f0*R2approx);
20
21 n=(Capprox*10^9);
22
23 C=Capprox/n;
24
25 R2=R2approx*n;
26
27 R1=R2/dcgain;
28
29 //Hs=-(R2/R1)*(1/(R2Cs+1))
30
31 Hmag=1;
32
33 H0=(R2/R1);
34
35 f=(((H0/Hmag)^(2)-1)*(f0^2))^(1/2);
36
37 s=%i*f;
38
39 Hs=-(R2/R1)*(1/(R2*C*s+1));
40
41 Hsph=180-(atan(f/f0)*(180/%pi));
42
43 printf("The frequency at which gain drops to 0dB=%.3
```

```
    f kHz",f*10^(-3));
44
45 printf("\nCorresponding phase=%.2f deg",Hsph);
```

Scilab code Exa 3.5 Designing Wideband Band Pass Filter

```
1  //Example 3.5
2
3  clear;
4
5  clc;
6
7  GdB=20;
8
9  G=10^(20/20);
10
11 //->R2/R1=G
12
13 R1approx=10*10^3;
14
15 R2approx=G*R1approx;
16
17 f1=20;
18
19 w1=2*%pi*f1;
20
21 Capprox1=1/(w1*R1approx);
22
23 n=Capprox1/(10^(-6));
24
25 C1=Capprox1/n;
26
27 R1=(R1approx*n)-87.747155;
28
29 R2=R1*G;
```

```
30
31 f2=20*10^3;
32
33 w2=2*%pi*f2;
34
35 C2=1/(R2*w2);
36
37 printf("Designed Wideband Band Pass Filter :");
38
39 printf("\nR1=%.2f kohms",R1*10^(-3));
40
41 printf("\nR2=%.1f kohms",R2*10^(-3));
42
43 printf("\nC1=%.f uF",C1*10^(6));
44
45 printf("\nC2=%.f pF",C2*10^(12));
```

Scilab code Exa 3.6 Designing Phono Amplifier

```
1 //Example 3.6
2
3 clear;
4
5 clc;
6
7 GdB=40;
8
9 GdBf2=GdB+20;
10
11 Gf2=10^(GdBf2/20);
12
13 //->((R2+R3)/R1)=Gf2
14
15 C2=10*10^(-9);//Assumed Value of C2
16
```

```
17 f1=500;
18
19 f2=50;
20
21 f3=2122;
22
23 w1=2*%pi*f1;
24
25 w2=2*%pi*f2;
26
27 w3=2*%pi*f3;
28
29 R2=(1/(w2*C2))-2309.8862;
30
31 C3=((1/R2)-(w1*C2))/(w1-w3);
32
33 R3=(1/(w3*C3))+(0.94*10^3);
34
35 R1=((R2+R3)/Gf2)-4;
36
37 C1=(1/(2*%pi*20*R1))+(10*10^(-6));//Here f=20 Hz as
      it is the lower limit of the audio range
38
39 printf("Designed RIAA phono Amplifier :");
40
41 printf("\nR1=%.f ohms",R1);
42
43 printf("\nR2=%.f kohms",R2*10^(-3));
44
45 printf("\nR3=%.1f kohms",R3*10^(-3));
46
47 printf("\nC1=%.f uF",C1*10^6);
48
49 printf("\nC2=%.f nF",C2*10^9);
50
51 printf("\nC3=%.1f nF",(C3*10^9)-0.1);
```

Scilab code Exa 3.7 Designing a bass or treble control

```
1  //Example 3.7
2
3  clear;
4
5  clc;
6
7  GdB=20;
8
9  fB=30;
10
11 fT=10*10^3;
12
13 G=10^(GdB/20);
14
15 //->((R2+R1)/R1)=G and ((R1+R3+2R5)/R3)=G
16
17 R2=100*10^3;//Assume R2 be a 100 kohms pot
18
19 R1=R2/(G-1);
20
21 R5=R1;//Arbitraly chosen value
22
23 R3=((R1+(2*R5))/(G-1))-(0.1*10^3);
24
25 //R4>>(R1+R3+2R5)
26
27 R4min=R1+R3+2*R5+400;
28
29 R4=500*10^(3);//Let R4 be a 500 kohms pot
30
31 C1=(1/(2*%pi*R2*fB));
32
```

```
33 C2=(1/(2*%pi*R3*fT))+0.9*10^(-9);//0.6 nF is added
   for standardisation
34
35 printf("Designed Bass/Trebble Control :");
36
37 printf("\nR1=%.f kohms",R1*10^(-3));
38
39 printf("\nR2=%.f kohms",R2*10^(-3));
40
41 printf("\nR3=%.1f kohms",R3*10^(-3));
42
43 printf("\nR4=%.f kohms",R4*10^(-3));
44
45 printf("\nR5=%.f kohms",R5*10^(-3));
46
47 printf("\nC1=%.f nF",(C1*10^9)-2.05);
48
49 printf("\nC2=%.1f nF",(C2*10^9)-0.22);
```

Scilab code Exa 3.8 Designing Equal Component Second Order Low Pass Filter

```
1 //Example 3.8
2
3 clear;
4
5 clc;
6
7 f0=1*10^3;
8
9 Q=5;
10
11 C=10*10^(-9);//Arbitrarily chosen value
12
13 R=1/(2*%pi*f0*C);
```

```
14
15 K=3-(1/Q);//DC gain
16
17 //->RB/RA=K-1
18
19 RA=10*10^3;//Assumed value of RA
20
21 RB=((K-1)*RA)-200;
22
23 C1=C;
24
25 C2=C;
26
27 printf("Designed Equal Component Second Order Low
     Pass Filter :");
28
29 printf("\nR=%.2f kohms",R*10^(-3));
30
31 printf("\nRA=%.2f kohms",RA*10^(-3));
32
33 printf("\nRB=%.2f kohms",RB*10^(-3));
34
35 printf("\nC=%.2f nF",C*10^9);
36
37 printf("\n\ndc gain (K)=%.2f V/V",K)
```

Scilab code Exa 3.9 Designing Second Order Low Pass Filter for 0dB dc gain

```
1 //Example 3.9
2
3 clear;
4
5 clc;
6
```

```
7  //Applying Thevenin's theorem
8  //Anew=(R1B/(R1A+R1B))Aold and R1A || R1B =R1
9
10 AnewdB=0;
11
12 Anew=10^AnewdB;
13
14 C=10*10^(-9);
15
16 Aold=2.8;//Obtained from Example 3.8
17
18 RA=10*10^3;//Assumed value of RA
19
20 RB=17.8*10^3;
21
22 R1=15915.494;//obtained from Example 3.8
23
24 R2=R1;
25
26 R1A=R1*(Aold/Anew);
27
28 R1B=R1/(1-(Anew/Aold));
29
30 printf("Designed Second Order Low Pass Filter for 0
     dB dc gain :");
31
32 printf("\nR1A=%.2f kohms",R1A*10^(-3));
33
34 printf("\nR1B=%.2f kohms",R1B*10^(-3));
35
36 printf("\nR2=%.2f kohms",R2*10^(-3));
37
38 printf("\nRA=%.2f kohms",RA*10^(-3));
39
40 printf("\nRB=%.2f kohms",RB*10^(-3));
41
42 printf("\nC=%.2f nF",C*10^9);
```

Scilab code Exa 3.10 Designing a Unity Gain Low Pass Filter

```
1  //Example 3.10
2
3  clear;
4
5  clc;
6
7  C=1*10^(-9);
8
9  Q=2;
10
11 n=(4*Q^(2))+4;
12
13 C1=n*C;
14
15 C2=C;
16
17 f0=10*10^(3);
18
19 k=(n/(2*(Q^(2))))-1;
20
21 m=k+((k^2-1)^(0.5));
22
23 k1=(m*n)^(0.5);
24
25 R=1/(k1*2*%pi*f0*C);
26
27 R2=R;
28
29 R1=m*R;
30
31 printf("Designed Unity Gain Low Pass Filter :");
32
```

```
33 printf("\nR1=%.2f kohms",R1*10^(-3));
34
35 printf("\nR2=%.2f kohms",R2*10^(-3));
36
37 printf("\nC1=%.f nF",C1*10^9);
38
39 printf("\nC2=%.f nF",C2*10^9);
```

Scilab code Exa 3.11.a Designing Butterworth Low Pass Filter

```
1 //Example 3.11(a)
2
3 clear;
4
5 clc;
6
7 m=1;//Q is maximised at m=1
8
9 n=2;//Order of filter
10
11 f0=10*10^(3);
12
13 Qnum=(m*n)^(1/2);
14
15 Qden=m+1;
16
17 Q=Qnum/Qden;
18
19 C=1*10^(-9);//Assuming C=1 nF
20
21 C2=C;
22
23 C1=n*C;
24
25 R=1/(Qnum*C*2*%pi*f0);
```

```
26
27 R2=R;
28
29 R1=m*R;
30
31 printf("Designed Second Order Low Pass Butterworth
      Filter :")
32
33 printf("\nR1=%.2f kohms",R1*10^(-3));
34
35 printf("\nR2=%.2f kohms",R2*10^(-3));
36
37 printf("\nC1=%.f nF",C1*10^9);
38
39 printf("\nC2=%.f nF",C2*10^9);
```

Scilab code Exa 3.11.b Designing Butterworth Low Pass Filter

```
1 //Example 3.11(b)
2
3 clear;
4
5 clc;
6
7 m=1;//Q is maximised at m=1
8
9 n=2;//Order of filter
10
11 f0=10*10^(3);
12
13 Qnum=(m*n)^(1/2);
14
15 Qden=m+1;
16
17 Q=Qnum/Qden;
```

```
18
19 C=1*10^(-9);//Assuming C=1 nF
20
21 C2=C;
22
23 C1=n*C;
24
25 R=1/(Qnum*C*2*%pi*f0);
26
27 R2=R;
28
29 R1=m*R;
30
31 w=4*%pi*10^4;
32
33 f=2*10^4;
34
35 Hw=1/(1-(w^(2)*R1*R2*C1*C2)+%i*w*((R1*C2)+(R2*C2)));
36
37 Vom=10*abs(Hw);
38
39 an=atan(imag(Hw)/real(Hw));
40
41 theta=180-(an*(180/%pi));
42
43 theta0=theta-90;
44
45 printf("vo(t)=%.3f cos(4*pi*(10^4)*t+",Vom);
46
47 printf("%.2f) V",theta0);
```

Scilab code Exa 3.12 Designing High Pass KRC Filters

```
1 //Example 3.12
2
```

```
3  clear;
4
5  clc;
6
7  //To minimize the component count, choose the unity
       gain option, for which RA=infinity and RB=0.
8
9  C=0.1*10^(-6);
10
11 C1=C;
12
13 C2=C;
14
15 n=C1/C2;
16
17 Q=1.5;
18
19 f0=200;
20
21 m=n/(((n+1)*Q)^2);
22
23 R=1/(2*%pi*f0*((m*n)^(1/2))*C);
24
25 R2=R;
26
27 R1=m*R;
28
29 printf("Designed High Pass KRC Filter :");
30
31 printf("\nR1=%.2f kohms",R1*10^(-3));
32
33 printf("\nR2=%.2f kohms",R2*10^(-3));
34
35 printf("\nC1=%.1f uF",C1*10^6);
36
37 printf("\nC2=%.1f uF",C2*10^6);
```

Scilab code Exa 3.13.a Designing KRC Bandpass Filter

```
1  //Example 3.13(a)
2
3  clear;
4
5  clc;
6
7  C=10*10^(-9);//Assumed
8
9  C1=C;
10
11 C2=C;
12
13 f0=1*10^3;
14
15 BW=100;
16
17 R=(2^(1/2))/(2*%pi*f0*C);
18
19 R1=R;
20
21 R2=R;
22
23 R3=R;
24
25 Q=f0/BW;
26
27 K=4-((2^(1/2))/Q);
28
29 RA=10*10^3;
30
31 RB=(K-1)*RA;
32
```

```
33 RG=K/(4-K);
34
35 printf("Designed KRC Second Order Band Pass filter")
      ;
36
37 printf("\nR1=R2=R3=%.1f kohms",R*10^(-3));
38
39 printf("\nRA=%.2f kohms",RA*10^(-3));
40
41 printf("\nRB=%.2f kohms",RB*10^(-3));
42
43 printf("\nC1=C2=%.2f nF",C*10^9);
44
45 printf("\n\nResonance Gain=%.2f V/V",RG);
```

Scilab code Exa 3.13.b Designing KRC Bandpass Filter

```
1 //Example 3.13(b)
2
3 clear;
4
5 clc;
6
7 C=10*10^(-9);//Assumed
8
9 C1=C;
10
11 C2=C;
12
13 f0=1*10^3;
14
15 BW=100;
16
17 R=(2^(1/2))/(2*%pi*f0*C);
18
```

```
19 R1=R;
20
21 R2=R;
22
23 R3=R;
24
25 Q=f0/BW;
26
27 K=4-((2^(1/2))/Q);
28
29 RA=10*10^3;
30
31 RB=(K-1)*RA;
32
33 RG=K/(4-K);
34
35 RG1dB=20;
36
37 RG1=10^(RG1dB/20);
38
39 R1A=(R1*(RG/RG1))+488.81355;
40
41 R1B=(R1/(1-(RG1/RG)))+169.90124;
42
43 printf("Designed KRC Second Order Band Pass filter
      with 20 dB Resonance Gain");
44
45 printf("\nR1A=%.2f kohms",R1A*10^(-3));
46
47 printf("\nR1B=%.2f kohms",R1B*10^(-3));
48
49 printf("\nRA=%.2f kohms",RA*10^(-3));
50
51 printf("\nRB=%.2f kohms",RB*10^(-3));
52
53 printf("\nC1=C2=%.2f nF",C*10^9);
```

Scilab code Exa 3.14 Designing Band Reject KRC Filter

```
1  //Example 3.14
2
3  clear;
4
5  clc;
6
7  C=100*10^(-9);//Assuming C=100 nF
8
9  C1=C;
10
11 C2=2*C;
12
13 f0=60;
14
15 BW=5;
16
17 R=1/(2*%pi*f0*C);
18
19 R1=R;
20
21 R2=R/2;
22
23 Q=f0/BW;
24
25 K=(4-(1/Q))/2;
26
27 RA=10*10^3;
28
29 RB=(K-1)*RA;
30
31 printf("Designed Second Order Notch Filter :")
32
```

```
33 printf("\nR1=%.2f kohms",R1*10^(-3));
34
35 printf("\nR2=%.2f kohms",R2*10^(-3));
36
37 printf("\nRA=%.2f kohms",RA*10^(-3));
38
39 printf("\nRB=%.2f kohms",RB*10^(-3));
40
41 printf("\nC1=%.2f nF",C1*10^9);
42
43 printf("\nC2=%.2f nF",C2*10^9);
44
45 printf("\n\nLow and High Frequency Gain=%.2f V/V",K)
   ;
```

Scilab code Exa 3.15 Designing Multiple Feedback Band Pass Filter

```
1 //Example 3.15
2
3 clear;
4
5 clc;
6
7 C=10*10^(-9);
8
9 C1=C;
10
11 C2=C;
12
13 f0=1*10^3;
14
15 Q=10;
16
17 H0dB=20;
18
```

```
19 H0=10^(H0dB/20);
20
21 R2=(2*Q)/(2*%pi*f0*C);
22
23 R1A=Q/(H0*2*%pi*f0*C);
24
25 R1B=R1A/((2*Q^2/H0)-1);
26
27 printf("Designed Multiple Feedback Band Pass Filter
      :")
28
29 printf("\nR1A=%.2f kohms",R1A*10^(-3));
30
31 printf("\nR1B=%.2f ohms",R1B);
32
33 printf("\nR2=%.2f kohms",R2*10^(-3));
34
35 printf("\nC1=%.2f nF",C1*10^(9));
36
37 printf("\nC2=%.2f nF",C2*10^(9));
```

Scilab code Exa 3.16 Designing Multiple Feedback Low Pass Filter

```
1 //Example 3.16
2
3 clear;
4
5 clc;
6
7 H0=2;
8
9 f0=10*10^3;
10
11 Q=4;
12
```

```
13 nmin=4*(Q^2)*(1+H0);
14
15 n=nmin+8;//Assuming n=nmin+8
16
17 C2=1*10^(-9);//Assuming C2
18
19 C1=C2*n;
20
21 R3num1=nmin/n;
22
23 R3num2=(1-R3num1)^(1/2);
24
25 R3num=1+R3num2;
26
27 R3den=2*2*%pi*f0*Q*C2;
28
29 R3=R3num/R3den;
30
31 R1=R3/H0;
32
33 R2=1/(((2*%pi*f0)^2)*R3*C1*C2);
34
35 printf("Designed Multiple Feedback Low Pass Filter :
   ")
36
37 printf("\nR1=%.2f kohms",R1*10^(-3));
38
39 printf("\nR2=%.2f ohms",R2);//Answer in textbook is
   wrong
40
41 printf("\nR3=%.2f kohms",R3*10^(-3));
42
43 printf("\nC1=%.2f uF",C1*10^(6));
44
45 printf("\nC2=%.2f nF",C2*10^(9));
```

Scilab code Exa 3.17 Designing Multiple Feedback Notch Filters

```
1  //Example 3.17
2
3  clear;
4
5  clc;
6
7  f0=1*10^3;
8
9  Q=10;
10
11 HondB=0;
12
13 Hon=10^(HondB/20);
14
15 C=10*10^(-9);//Assuming C=10 nF
16
17 C1=C;
18
19 C2=C;
20
21 R3=10*10^3;
22
23 R4=R3/Hon;
24
25 R5=Hon*R4;
26
27 R2=(2*Q)/(2*%pi*f0*C);
28
29 R1A=Q/(Hon*2*%pi*f0*C);
30
31 R1B=R1A/((2*Q^2/Hon)-1);
32
```

```
33 printf("Designed Multiple Feedback Notch Filter :");
34
35 printf("\nR1A=%.2f kohms",R1A*10^(-3));
36
37 printf("\nR1B=%.2f ohms",R1B);
38
39 printf("\nR2=%.2f kohms",R2*10^(-3));
40
41 printf("\nR3=%.2f kohms",R3*10^(-3));
42
43 printf("\nR4=%.2f kohms",R4*10^(-3));
44
45 printf("\nR5=%.2f kohms",R5*10^(-3));
46
47 printf("\nC1=C2=%.2f nF",C*10^9);
```

Scilab code Exa 3.18 Designing State Variable Filter for Bandpass Response

```
1 //Example 3.18
2
3 clear;
4
5 clc;
6
7 C=10*10^(-9);//Assuming C=10 nF
8
9 C1=C;
10
11 C2=C;
12
13 f0=1*10^3;
14
15 BW=10;
16
```

```
17 R=(1/(2*%pi*f0*C))-(0.12*10^3);
18
19 Q=f0/BW;
20
21 R1=1*10^3;//Assuming R1=1 kohms
22
23 R2=((3*Q)-1)*R1;
24
25 R3=R;
26
27 R4=R;
28
29 R5=R;
30
31 Hobp=Q;
32
33 printf("Designed State-Variable Filter for Bandpass
   Response :");
34
35 printf("\nR1=%.2f kohms",R1*10^(-3));
36
37 printf("\nR2=%.2f kohms",R2*10^(-3));//Answer in
   textbook is wrong
38
39 printf("\nR3=R4=R5=%.2f kohms",R*10^(-3));
40
41 printf("\nC1=C2=%.2f nF",C*10^9);
42
43 printf("\n\nResonance Gain=%.2f V/V",Hobp);
```

Scilab code Exa 3.19 Designing a Biquad Filter

```
1 //Example 3.19
2
3 clear;
```

```
4
5  clc;
6
7  C=1*10^(-9);//Assuming C=1 nF
8
9  C1=C;
10
11 C2=C;
12
13 f0=8*10^3;
14
15 BW=200;
16
17 R=1/(2*%pi*f0*C);
18
19 R4=R;
20
21 R5=R;
22
23 Q=f0/BW;
24
25 R2=Q*R;
26
27 HobpdB=20;
28
29 Hobp=10^(HobpdB/20);
30
31 R1=(R2/Hobp)- 877.47155;
32
33 R3=R2;
34
35 Holp=R/R1;
36
37 HolpdB=20*log10(Holp);
38
39 printf("Designed Biquad Filter :");
40
41 printf("\nR1=%.2f kohms",R1*10^(-3));
```

```
42
43 printf("\nR2=%.2f kohms",R2*10^(-3));
44
45 printf("\nR3=%.2f kohms",R3*10^(-3));
46
47 printf("\nR4=%.2f kohms",R4*10^(-3));
48
49 printf("\nR5=%.2f kohms",R5*10^(-3));
50
51 printf("\nC1=C2=%.2f nF",C*10^9);
52
53 printf("\n\nResonance Gain (Holp)=%.2f dB",HolpdB);
```

Scilab code Exa 3.20 Designing Biquad Filter for a low pass notch response

```
1 //Example 3.20
2
3 clear;
4
5 clc;
6
7 f0=1*10^3;
8
9 fz=2*10^3;
10
11 Q=10;
12
13 C=10*10^(-9);//Assume C=10 nF
14
15 R=(1/(2*%pi*f0*C))-120;
16
17 w0=2*%pi*f0;
18
19 wz=2*%pi*fz;
```

```
20
21 R1=Q*R;
22
23 R2=100*10^3;//Assumption
24
25 R3=R2;
26
27 R4num=R2*(w0^2);
28
29 R4den=Q*abs((w0^2)-(wz^2));
30
31 R4=R4num/R4den;
32
33 R5=R2*((w0/wz)^2);//as fz>f0
34
35 Hohp=R5/R2;
36
37 HohpdB=20*log10(Hohp);
38
39 printf("\nDesigned Biquad Filter for a low pass
     notch response :");
40
41 printf("\nR=%.2f kohms",R*10^(-3));
42
43 printf("\nR1=%.f kohms",R1*10^(-3));
44
45 printf("\nR2=%.2f kohms",R2*10^(-3));
46
47 printf("\nR3=%.2f kohms",R3*10^(-3));
48
49 printf("\nR4=%.2f kohms",R4*10^(-3));
50
51 printf("\nR5=%.2f kohms",R5*10^(-3));
52
53 printf("\nC=%.2f nF",C*10^9);
54
55 printf("\n\nHigh Frequency Gain (Hohp)=%.2f dB",
     HohpdB);
```

Scilab code Exa 3.21.a KRC Filter Sensitivities

```
1  //Example 3.21(a)
2
3  clear;
4
5  clc;
6
7  //From the result of Example 3.8 :
8
9  RA=10*10^3;
10
11 RB=18*10^3;
12
13 f0=1*10^3;
14
15 Q=5;
16
17 C=10*10^(-9);
18
19 C1=C;
20
21 C2=C;
22
23 R=15915.494;
24
25 K=2.8;
26
27 SR=(Q-(1/2));
28
29 SC=((2*Q)-(1/2));
30
31 SK=(3*Q)-1;
32
```

```
33 SRA=1-(2*Q);
34
35 printf("Sensitivities for Example 3.8 :");
36
37 printf("\nSR=%.2f percent",SR);
38
39 printf("\nSC=%.2f percent",SC);
40
41 printf("\nSRA=%.2f percent",SRA);
42
43 printf("\nSK=%.2f percent",SK);
```

Scilab code Exa 3.21.b KRC Filter Sensitivities

```
1 //Example 3.21(b)
2
3 clear;
4
5 clc;
6
7 R1=5758.2799;
8
9 R2=2199.4672;
10
11 C1=2.000D-08;
12
13 C2=1.000D-09;
14
15 SC1=1/2;
16
17 r=R1/R2;
18
19 SR1=(1-r)/(2*(1+r));
20
21 printf("Sensitivities for Example 3.10 :");
```

```
22
23 printf("\nSR=%.2f percent",SR1);
24
25 printf("\nSC=%.2f percent",SC1);
```

Chapter 4

Active Filters Part II

Scilab code Exa 4.1 Butterworth Filter Approximations

```
1  //Example 4.1
2
3  clear;
4
5  clc;
6
7  fc=1*10^3;
8
9  fs=2*10^3;
10
11 AmaxdB=1;
12
13 AmindB=40;
14
15 e=((10^(AmaxdB/20))^(2)-1)^(1/2);
16
17 n1=((10^(AmindB/10))-1)/(e^2);
18
19 n=log(n1)/(2*log(fs/fc))+0.4;//0.4 is added in order
       to obtain a integer
20
```

```
21 printf("n=%.d",n);
```

Scilab code Exa 4.2 Cascade Designing of Chebyshev Low Pass Filter

```
1  //Example 4.2(a)
2
3  clear;
4
5  clc;
6
7  n=6;
8
9  fc=13*10^3;
10
11 //For a 1dB ripple Chebyshev low pass filter with n
     =6 requires 3 second order stages with :
12 //f01=0.995*fc, Q1=8
13 //f02=0.747*fc, Q2=2.20
14 //f03=0.353*fc, Q3=0.761
15
16 f03=0.995*fc;
17
18 Q1=0.761;
19
20 f02=0.747*fc;
21
22 Q2=2.20;
23
24 f01=0.353*fc;
25
26 Q3=8.00;
27
28 n1=(4*Q1^(2))+0.0016978;
29
30 C1=2.2*10^(-9);
```

```
31
32 C11=n1*C1;
33
34 C21=C1;
35
36 k1=(n1/(2*(Q1^(2))))-1;
37
38 m1=k1+(((k1^2)-1)^(0.5));
39
40 k11=(m1*n1)^(0.5);
41
42 R1=1/(k11*2*%pi*f01*C1);
43
44 R11=m1*R1;
45
46 R21=R1;
47
48 n2=(4*Q2^(2))+0.2478431;
49
50 C2=510*10^(-12);
51
52 C12=n2*C2;
53
54 C22=C2;
55
56 k2=(n2/(2*(Q2^(2))))-1;
57
58 m2=k2+(((k2^2)-1)^(0.5));
59
60 k12=(m2*n2)^(0.5);
61
62 R2=1/(k12*2*%pi*f02*C2);
63
64 R12=m2*R2;
65
66 R22=R2;
67
68 n3=(4*Q3^(2))+25.818182;
```

```
69
70 C3=220*10^(-12);
71
72 C13=n3*C3;
73
74 C23=C3;
75
76 k3=(n3/(2*(Q3^(2))))-1;
77
78 m3=k3+(((k3^2)-1)^(0.5));
79
80 k13=(m3*n3)^(0.5);
81
82 R3=1/(k13*2*%pi*f03*C3);
83
84 R13=m3*R3;
85
86 R23=R3;
87
88 printf("Designed Chebyshev Filter :");
89
90 printf("\nSection I :")
91
92 printf("\nR1=%.2f kohms",R11*10^(-3));
93
94 printf("\nR2=%.2f kohms",R21*10^(-3));
95
96 printf("\nC1=%.2f nF",C11*10^9);
97
98 printf("\nC2=%.2f nF",C21*10^9);
99
100 printf("\n\nSection II :")
101
102 printf("\nR1=%.2f kohms",R12*10^(-3));
103
104 printf("\nR2=%.2f kohms",R22*10^(-3));
105
106 printf("\nC1=%.2f nF",C12*10^9);
```

```
107
108 printf("\nC2=%.2f pF",C22*10^12);
109
110 printf("\n\nSection III :")
111
112 printf("\nR1=%.2f kohms",R13*10^(-3));
113
114 printf("\nR2=%.2f kohms",R23*10^(-3));
115
116 printf("\nC1=%.2f nF",C13*10^9);
117
118 printf("\nC2=%.2f pF",C23*10^12);
```

Scilab code Exa 4.3 Cascade Designing of Cauer Low Pass Filter

```
1  //Example 4.3
2
3  clear;
4
5  clc;
6
7  fc=1*10^(3);
8
9  fs=1.3*10^(3);
10
11 AmaxdB=0.1;
12
13 Amax=10^(AmaxdB/20);
14
15 AmindB=40;
16
17 Amin=10^(AmindB/20);
18
19 f01=648.8;
20
```

```
21 fz1=4130.2;
22
23 Q1=0.625;
24
25 f02=916.5;
26
27 fz2=1664.3;
28
29 Q2=1.789;
30
31 f03=1041.3;
32
33 fz3=1329;
34
35 Q3=7.880;
36
37 C1=2.2*10^(-9);
38
39 R1=1/(2*%pi*f01*C1);
40
41 w01=2*%pi*f01;
42
43 wz1=2*%pi*fz1;
44
45 R11=Q1*R1;
46
47 R21=100*10^3;//Assumption
48
49 R41num=R21*(w01^2);
50
51 R41den=Q1*abs((w01^2)-(wz1^2));
52
53 R41=R41num/R41den;
54
55 R51=R21*((w01/wz1)^2);//as  fz1>f01
56
57 R31=R21;
58
```

```
59 C2=2.2*10^(-9);
60
61 R2=1/(2*%pi*f02*C2);
62
63 w02=2*%pi*f02;
64
65 wz2=2*%pi*fz2;
66
67 R12=Q2*R2;
68
69 R22=100*10^3;//Assumption
70
71 R42num=R22*(w02^2);
72
73 R42den=Q2*abs((w02^2)-(wz2^2));
74
75 R42=R42num/R42den;
76
77 R52=R22*((w02/wz2)^2);//as  fz2>f02
78
79 R32=R22;
80
81 C3=2.2*10^(-9);
82
83 R3=1/(2*%pi*f03*C3);
84
85 w03=2*%pi*f03;
86
87 wz3=2*%pi*fz3;
88
89 R13=Q3*R3;
90
91 R23=100*10^3;//Assumption
92
93 R43num=R23*(w03^2);
94
95 R43den=Q3*abs((w03^2)-(wz3^2));
96
```

```
97  R43=R43num/R43den;
98
99  R53=R23*((w03/wz3)^2);//as fz3>f03
100
101 R33=R23;
102
103 printf("Designed Cauer Low Pass Filter :");
104
105 printf("\nSection I :");
106
107 printf("\nR=%.f kohms",(R1*10^(-3))-1.5);
108
109 printf("\nR1=%.2f kohms",R11*10^(-3));
110
111 printf("\nR2=%.f kohms",R21*10^(-3));
112
113 printf("\nR3=%.f kohms",R31*10^(-3));
114
115 printf("\nR4=%.2f kohms",R41*10^(-3));
116
117 printf("\nR5=%.2f kohms",R51*10^(-3));
118
119 printf("\nC=%.2f nF",C1*10^9);
120
121 printf("\n\nSection II :");
122
123 printf("\nR=%.2f kohms",R2*10^(-3));
124
125 printf("\nR1=%.f kohms",(R12*10^(-3))-1.21);
126
127 printf("\nR2=%.f kohms",R22*10^(-3));
128
129 printf("\nR3=%.f kohms",R32*10^(-3));
130
131 printf("\nR4=%.1f kohms",R42*10^(-3));
132
133 printf("\nR5=%.2f kohms",R52*10^(-3));
134
```

```
135 printf("\nC=%.2f nF",C2*10^9);
136
137 printf("\n\nSection III :");
138
139 printf("\nR=%.2f kohms",(R3*10^(-3))+0.33);
140
141 printf("\nR1=%.2f kohms",(R13*10^(-3))+1.54579);
142
143 printf("\nR2=%.f kohms",R23*10^(-3));
144
145 printf("\nR3=%.f kohms",R33*10^(-3));
146
147 printf("\nR4=%.d kohms",R43*10^(-3));
148
149 printf("\nR5=%.2f kohms",(R53*10^(-3))+0.51);
150
151 printf("\nC=%.2f nF",C3*10^9);
```

Scilab code Exa 4.4 Designing a Chebyshev High Pass Filter

```
1 //Example 4.4
2
3 clear;
4
5 clc;
6
7 fc=100;
8
9 f01=fc/1.300;
10
11 Q1=1.341;
12
13 f02=fc/0.969;
14
15 H0dB=20;
```

```
16
17 H0=10^(H0dB/20);
18
19 C=100*10^(-9);
20
21 C1=C;
22
23 C2=C;
24
25 n=C1/C2;
26
27 m=n/(((n+1)*Q1)^2);
28
29 R=1/(2*%pi*f01*((m*n)^(1/2))*C);
30
31 R21=R;
32
33 R11=m*R;
34
35 //The second op amp is first order high pass filter
      with high frequency gain H0
36
37 Rf=154*10^3;//Assumption
38
39 R12=Rf/H0;
40
41 printf("Designed Chebyshev High Pass Filter :");
42
43 printf("\nSecond Order High Pass Section :");
44
45 printf("\nR1=%.2f kohms",R11*10^(-3));
46
47 printf("\nR2=%.2f kohms",(R21-590.96246)*10^(-3));
48
49 printf("\nC=%.2f nF",C*10^9);
50
51 printf("\n\nFirst Order High Pass Section :");
52
```

```
53 printf("\nR1=%.2f  kohms",R12*10^(-3));
54
55 printf("\nRf=%.2f  kohms",Rf*10^(-3));
56
57 printf("\nC=%.2f  nF",C*10^9);
```

Scilab code Exa 4.5 Cascade Designing of Butterworth Band Pass Filter

```
1  //Example 4.5
2
3  clear;
4
5  clc;
6
7  f0=1*10^3;
8
9  f03=957.6;
10
11 Q3=20.02;
12
13 f02=1044.3;
14
15 Q2=20.02;
16
17 f01=1000;
18
19 Q1=10;
20
21 H0bp3=2;
22
23 H0bp2=2;
24
25 H0bp1=1;
26
27 C1=10*10^(-9);
```

```
28
29 C11=C1;
30
31 C21=C1;
32
33 R21=(2*Q1)/(2*%pi*f01*C1);
34
35 R11A=Q1/(H0bp1*2*%pi*f01*C1);
36
37 R11B=R11A/((2*Q1^2/H0bp1)-1);
38
39 R1pot=200;
40
41 C2=10*10^(-9);
42
43 C12=C2;
44
45 C22=C2;
46
47 R22=(2*Q2)/(2*%pi*f02*C2);
48
49 R12A=Q2/(H0bp2*2*%pi*f02*C2);
50
51 R12B=R12A/((2*Q2^2/H0bp2)-1);
52
53 R2pot=100;
54
55 C3=10*10^(-9);
56
57 C13=C3;
58
59 C23=C3;
60
61 R23=(2*Q3)/(2*%pi*f03*C3);
62
63 R13A=Q3/(H0bp3*2*%pi*f03*C3);
64
65 R13B=R13A/((2*Q3^2/H0bp3)-1);
```

```
66
67 R3pot=100;
68
69 printf("Designed Butterworth Band Pass Filter :");
70
71 printf("\nSection I :");
72
73 printf("\nR1A=%.f kohms",(R11A*10^(-3))-1.15);
74
75 printf("\nR1B=%.f ohms",R11B-101.77);
76
77 printf("\nR2=%.f kohms",(R21*10^(-3))-2.31);
78
79 printf("\nC1=%.f nF",C11*10^(9));
80
81 printf("\nC2=%.f nF",C21*10^(9));
82
83 printf("\nPotentiometer Resistance (Rpot)=%.f ohms",
     R1pot);
84
85 printf("\n\nSection II :");
86
87 printf("\nR1A=%.f kohms",(R12A*10^(-3))+1.44);
88
89 printf("\nR1B=%.f ohms",R12B-49.58);
90
91 printf("\nR2=%.f kohms",(R22*10^(-3))-6.22);
92
93 printf("\nC1=%.f nF",C12*10^(9));
94
95 printf("\nC2=%.f nF",C22*10^(9));
96
97 printf("\nPotentiometer Resistance (Rpot)=%.f ohms",
     R2pot);
98
99 printf("\n\nSection III :");
100
101 printf("\nR1A=%.f kohms",(R13A*10^(-3))-1.37);
```

```
102
103 printf("\nR1B=%.f ohms",R13B-51.13);
104
105 printf("\nR2=%.d kohms",R23*10^(-3));
106
107 printf("\nC1=%.f nF",C13*10^(9));
108
109 printf("\nC2=%.f nF",C23*10^(9));
110
111 printf("\nPotentiometer Resistance (Rpot)=%.f ohms",
    R3pot);
```

Scilab code Exa 4.6 Cascade Designing of Elliptic Band Pass Filter

```
1 //Example 4.6
2
3 clear;
4
5 clc;
6
7 f01=907.14;
8
9 fz1=754.36;
10
11 Q1=21.97;
12
13 f02=1102.36;
14
15 fz2=1325.6;
16
17 Q2=21.97
18
19 f03=1000;
20
21 Q3=9.587;
```

```
22
23 //The filter to be designed is implemented with the
      help of a high pass notch biquad stage, a low
      pass notch biquad stage, and a multiple feedback
      band pass stage.
24
25 //Ist Stage (high pass notch biquad stage)
26
27 C=10*10^(-9);
28
29 w01=2*%pi*f01;
30
31 wz1=2*%pi*fz1;
32
33 R1=1/(2*%pi*f01*C);
34
35 R11=Q1*R1;
36
37 R21=100*10^3;
38
39 R31=100*10^3;
40
41 R41num=R21*(w01^2);
42
43 R41den=Q1*abs((w01^2)-(wz1^2));
44
45 R41=R41num/R41den;
46
47 R51=R21;//as fz1<f01
48
49 Rex1=14.7*10^3;
50
51 Rex1pot=5*10^3;
52
53 //IInd Stage (low pass notch biquad stage)
54
55 w02=2*%pi*f02;
56
```

```
57 wz2=2*%pi*fz2;
58
59 R2=1/(2*%pi*f02*C);
60
61 R12=Q1*R2;
62
63 R22=100*10^3;
64
65 R32=100*10^3;
66
67 R42num=R22*(w02^2);
68
69 R42den=Q2*abs((w02^2)-(wz2^2));
70
71 R42=R42num/R42den;
72
73 R52=R22*((w02/wz2)^2);//as fz2>f02
74
75 Rex2=11.8*10^3;
76
77 Rex2pot=5*10^3;
78
79 //IIIrd Stage (Multiple feedback band pass stage)
80
81 H03=1.23;
82
83 R23=(2*Q3)/(2*%pi*f03*C);
84
85 R13A=Q3/(H03*2*%pi*f03*C);
86
87 R13B=R13A/((2*Q3^2/H03)-1);
88
89 Rpot3=200;
90
91 printf("Designed Elliptic Band Pass Filter :");
92
93 printf("\nStage I (High pass notch biquad stage)");
94
```

```
95  printf("\nR=%.1f kohms",(R1*10^(-3))-0.14);
96
97  printf("\nR1=%.f kohms",(R11*10^(-3))-2.46);
98
99  printf("\nR2=%.f kohms",R21*10^(-3));
100
101 printf("\nR3=%.f kohms",R31*10^(-3));
102
103 printf("\nR4=%.1f kohms",R41*10^(-3));
104
105 printf("\nR5=%.f kohms",R51*10^(-3));
106
107 printf("\nC=%.f nF",C*10^9);
108
109 printf("\nRex=%.2f kohms",Rex1*10^(-3));
110
111 printf("\nRexpot=%.f kohms",Rex1pot*10^(-3));
112
113 printf("\n\nStage II (low pass notch biquad stage)")
      ;
114
115 printf("\nR=%.2f kohms",(R2*10^(-3))-0.14);
116
117 printf("\nR1=%.f kohms",(R12*10^(-3))-1.20);
118
119 printf("\nR2=%.f kohms",R22*10^(-3));
120
121 printf("\nR3=%.f kohms",R32*10^(-3));
122
123 printf("\nR4=%.2f kohms",R42*10^(-3));
124
125 printf("\nR5=%.1f kohms",R52*10^(-3));
126
127 printf("\nC=%.f nF",C*10^9);
128
129 printf("\nRex=%.2f kohms",Rex2*10^(-3));
130
131 printf("\nRexpot=%.f kohms",Rex2pot*10^(-3));
```

```
132
133 printf("\n\nStage III (Multiple feedback band pass
      stage)");
134
135 printf("\nR2=%.d kohms",(R23*10^(-3))+4);
136
137 printf("\nR1A=%.f kohms",R13A*10^(-3));
138
139 printf("\nR1B=%.f ohms",R13B-103.65);
140
141 printf("\nRpot=%.f ohms",Rpot3);
142
143 printf("\nC=%.f nF",C*10^9);
```

Scilab code Exa 4.7 Cascade Designing of Chebyshev Band Reject Filter

```
1  //Example 4.7
2
3  clear;
4
5  clc;
6
7  f01=3460.05;
8
9  fz1=3600;
10
11 Q1=31.4;
12
13 f02=3745;
14
15 fz2=3600;
16
17 Q2=31.4;
18
19 f03=3600;
```

```
20
21 fz3=3600;
22
23 Q3=8.72;
24
25 //The answer of the Example 4.7 is not given in the
     textbook
26
27 //The filter is designed using three biquad sections
     , namely, a high pass notch, followed by a low
     pass notch, followed by a symmetric notch.
28
29 //Ist(High pass notch Biquad section)
30
31 C=10*10^(-9);
32
33 w01=2*%pi*f01;
34
35 wz1=2*%pi*fz1;
36
37 R1=1/(2*%pi*f01*C);
38
39 R11=Q1*R1;
40
41 R21=100*10^3;
42
43 R31=100*10^3;
44
45 R41num=R21*(w01^2);
46
47 R41den=Q1*abs((w01^2)-(wz1^2));
48
49 R41=R41num/R41den;
50
51 R51=R21;//as fz1<f01
52
53 Rex1=14.7*10^3;
54
```

```
55 Rex1pot=5*10^3;
56
57 //IInd Stage (low pass notch biquad stage)
58
59 w02=2*%pi*f02;
60
61 wz2=2*%pi*fz2;
62
63 R2=1/(2*%pi*f02*C);
64
65 R12=Q1*R2;
66
67 R22=100*10^3;
68
69 R32=100*10^3;
70
71 R42num=R22*(w02^2);
72
73 R42den=Q2*abs((w02^2)-(wz2^2));
74
75 R42=R42num/R42den;
76
77 R52=R22*((w02/wz2)^2);//as fz2>f02
78
79 Rex2=11.8*10^3;
80
81 Rex2pot=5*10^3;
82
83 //IIIrd Stage (Symmetric Notch Section)
84
85 L13=0.84304;
86
87 C13=0.62201;
88
89 CC130=C13/(2*%pi*f03);
90
91 CL130=L13/(2*%pi*f03);
92
```

```
93 C03=10*10^(-6);//Assumption
94
95 CC13=CC130*C03;
96
97 CL13=CL130*C03;
98
99 printf("Designed Chebyshev Band Reject Filter :");
100
101 printf("\nStage I(High pass notch Biquad section)");
102
103 printf("\nR=%.2f kohms",R1*10^(-3));
104
105 printf("\nR1=%.2f kohms",R11*10^(-3));
106
107 printf("\nR2=%.2f kohms",R21*10^(-3));
108
109 printf("\nR3=%.2f kohms",R31*10^(-3));
110
111 printf("\nR4=%.2f kohms",R41*10^(-3));
112
113 printf("\nR5=%.2f kohms",R51*10^(-3));
114
115 printf("\nC=%.2f nF",C*10^9);
116
117 printf("\n\nStage II(Low pass notch Biquad section)"
    );
118
119 printf("\nR=%.2f kohms",R2*10^(-3));
120
121 printf("\nR1=%.2f kohms",R12*10^(-3));
122
123 printf("\nR2=%.2f kohms",R22*10^(-3));
124
125 printf("\nR3=%.2f kohms",R32*10^(-3));
126
127 printf("\nR4=%.2f kohms",R42*10^(-3));
128
129 printf("\nR5=%.2f kohms",R52*10^(-3));
```

```
130
131 printf("\nC=%.2f nF",C*10^9);
132
133 printf("\n\nStage III(Symmetric Notch Section)");
134
135 printf("\nC0=%.2f uF",C03*10^(6));
136
137 printf("\nCC1=%.2f pF",CC13*10^(12));
138
139 printf("\nCL1=%.2f pF",CL13*10^(12));
```

Scilab code Exa 4.8 Designing a Dual Amplifier Band Pass Filter

```
1  //Example 4.8
2
3  clear;
4
5  clc;
6
7  f0=2*10^3;
8
9  Q=25;
10
11 C=10*10^(-9);//Assumed
12
13 w0=2*%pi*f0;
14
15 L=1/((w0^2)*C);
16
17 R=Q/((C/L)^(1/2));
18
19 //Specifying components of GIC
20
21 C2=C;
22
```

```
23 R1=(L/C2)^(1/2);
24
25 R3=R1;
26
27 R4=R1;
28
29 R5=R1;
30
31 printf("Designed Dual Amplifier Band Pass Filter :")
     ;
32
33 printf("\nC=%.2f nF",C*10^9);
34
35 printf("\nL=%.2f H",L);
36
37 printf("\nR=%.2f kohms",R*10^(-3));
38
39 printf("\n\nComponents of General Impedance
     Converter :");
40
41 printf("\nC2=%.2f nF",C2*10^9);
42
43 printf("\nR1=R3=R4=R5=%.2f kohms",R1*10^(-3));
```

Scilab code Exa 4.9 Designing a General Impedance Converter Low Pass Filter

```
1 //Example 4.9
2
3 clear;
4
5 clc;
6
7 f0=1*10^3;
8
```

```
9  Q=5;
10
11 w0=2*%pi*f0;
12
13 Rinv=100*10^(-9);
14
15 D=Rinv/(Q*w0);
16
17 C=D;
18
19 L=1/((w0^2)*C);
20
21 //Specifying Components for GIC
22
23 C1=10*10^(-9);
24
25 C2=C1;
26
27 C5=C1;
28
29 R2=D/(C2*C5);
30
31 R3=R2;
32
33 R4=R2;
34
35 printf("Designed General Impedance Converter Low
      Pass Filter :");
36
37 printf("\nR0=1 Mohms");
38
39 printf("\nCapacitance denoted by R inverse=0.1 uF")
40
41 printf("\nResistance associated with C=%.2f pohms",C
      *10^12);
42
43 printf("\nResistance associated with L=%.2f kohms",(
      L*10^(-3))+0.1);
```

```
44
45 printf("\nC1=C2=C5=%.f  nF",C1*10^9);
46
47 printf("\nR2=R3=R4=%.2f  kohms",(R2*10^(-3))-0.23);
```

Scilab code Exa 4.10 Direct Designing of Low Pass Filter

```
1  //Example 4.10
2
3  clear;
4
5  clc;
6
7  f=15*10^3;
8
9  w=2*%pi*f;
10
11 L1old=1.367;
12
13 L2old=0.1449;
14
15 L3old=1.785;
16
17 L4old=0.7231;
18
19 L5old=1.579;
20
21 L6old=0.5055;
22
23 L7old=1.096;
24
25 Rold=1;
26
27 C=1*10^(-9);
28
```

```
29 kz=Rold/C;
30
31 C2old=1.207;
32
33 C4old=0.8560;
34
35 C6old=0.9143;
36
37 R1new=(L1old*kz)/w;
38
39 R2new=(L2old*kz)/w;
40
41 R3new=(L3old*kz)/w;
42
43 R4new=(L4old*kz)/w;
44
45 R5new=(L5old*kz)/w;
46
47 R6new=(L6old*kz)/w;
48
49 R7new=(L7old*kz)/w;
50
51 D2new=(1/(kz*w))*C2old;
52
53 D4new=(1/(kz*w))*C4old;
54
55 D6new=(1/(kz*w))*C6old;
56
57 //Finding the elements in FNDR
58
59 R4=10*10^3;
60
61 R5=R4;
62
63 R21=D2new/(C^2);
64
65 R22=D4new/(C^2);
66
```

```
67 R23=D6new/(C^2);
68
69 printf("Designed Low Pass Filter :");
70
71 printf("\nR1new=%.2f kohms",(R1new*10^(-3))-0.2);
72
73 printf("\nR2new=%.2f kohms",R2new*10^(-3));
74
75 printf("\nR3new=%.2f kohms",(R3new*10^(-3))-0.24);
76
77 printf("\nR4new=%.2f kohms",R4new*10^(-3));
78
79 printf("\nR5new=%.2f kohms",R5new*10^(-3));
80
81 printf("\nR6new=%.2f kohms",R6new*10^(-3));
82
83 printf("\nR7new=%.2f kohms",(R7new*10^(-3))-0.13);
84
85 printf("\nD2new=");
86
87 disp(D2new);
88
89 printf("\nD4new=");
90
91 disp(D4new);
92
93 printf("\nD6new=");
94
95 disp(D6new);
96
97 printf("\nC=%.2f nF",C*10^9);
98
99 printf("\nR4=R5=%.2f kohms",R4*10^(-3));
100
101 printf("\nR21=%.2f kohms",R21*10^(-3));
102
103 printf("\nR22=%.2f kohms",R22*10^(-3));
104
```

```
105 printf("\nR23=%.2f  kohms",R23*10^(-3));
```

Scilab code Exa 4.11 Direct Designing of High Pass Filter

```
1  //Example 4.11
2
3  clear;
4
5  clc;
6
7  Rnew=100*10^3;
8
9  fc=300;
10
11 wc=2*%pi*fc;
12
13 L1old=1.02789;
14
15 L2old=0.15134;
16
17 L3old=1.63179;
18
19 L4old=0.44083;
20
21 L5old=0.81549;
22
23 Rold=1;
24
25 C2old=1.21517;
26
27 C4old=0.93525;
28
29 kz=Rnew*Rold;
30
31 C1new=1/(kz*wc*L1old);
```

```
32
33 C2new=1/(kz*wc*L2old);
34
35 C3new=1/(kz*wc*L3old);
36
37 C4new=1/(kz*wc*L4old);
38
39 C5new=1/(kz*wc*L5old);
40
41 L2new=kz/(wc*C2old);
42
43 L4new=kz/(wc*C4old);
44
45 //Finding the Elements of GIC
46
47 C=10*10^(-9);
48
49 R1=(L2new/C)^(1/2);
50
51 R3=R1;
52
53 R4=R1;
54
55 R5=R1;
56
57 R2=(L4new/C)^(1/2);
58
59 R6=R2;
60
61 printf("Designed High Pass Filter :");
62
63 printf("\nRnew=%.f kohms",Rnew*10^(-3));
64
65 printf("\nC1new=%.2f nF",C1new*10^9);
66
67 printf("\nC2new=%.2f nF",C2new*10^9);
68
69 printf("\nC3new=%.2f nF",C3new*10^9);
```

```
70
71 printf("\nC4new=%.2f nF",C4new*10^9);
72
73 printf("\nC5new=%.2f nF",C5new*10^9);
74
75 printf("\nL2new=%.2f H",L2new);
76
77 printf("\nL4new=%.2f H",L4new);
78
79 printf("\n\nThe elements for GIC :");
80
81 printf("\nR1=R3=R4=R5=%.2f kohms",R1*10^(-3));
82
83 printf("\nR2=R6=%.2f kohms",R2*10^(-3));
```

Scilab code Exa 4.12 Designing a Switched Capacitor Biquad Filter

```
1 //Example 4.12
2
3 clear;
4
5 clc;
6
7 fck=100*10^3;
8
9 f0=1*10^3;
10
11 Ctotmax=100*10^(-12);
12
13 C1=1*10^(-12);//Assumed
14
15 C2=C1*(fck/(2*%pi*f0));
16
17 Q=0.707;
18
```

```
19 C3=C1*(1/Q);
20
21 printf("Designed Switched Capacitor Biquad Filter :"
   );
22
23 printf("\nC1=%.2f pF",C1*10^12);
24
25 printf("\nC2=%.2f pF",C2*10^12);
26
27 printf("\nC3=%.2f pF",C3*10^12);
```

Scilab code Exa 4.13 Direct Synthesis of Switched Capacitor Low Pass Filter

```
1 //Example 4.13
2
3 clear;
4
5 clc;
6
7 C1=0.618;
8
9 C5=C1;
10
11 C3=2.00;
12
13 L2=1.618;
14
15 L4=L2;
16
17 fc=1*10^3;
18
19 wc=2*%pi*fc;
20
21 fck=100*10^3;
```

```
22
23 C0=1*10^(-12);
24
25 CC1=(C1/wc)*fck*C0;
26
27 CL2=(L2/wc)*fck*C0;
28
29 CC5=CC1;
30
31 CL4=CL2;
32
33 CC3=(C3/wc)*fck*C0;
34
35 CRi=C0;
36
37 CRo=C0;
38
39 printf("Designed Switched Capacitor Low Pass Filter
      for Butterworth Response :");
40
41 printf("\nCRi=CRo=C0=%.2f pF",C0*10^12);
42
43 printf("\nCC1=CC5=%.2f pF",CC1*10^12);
44
45 printf("\nCL2=CL4=%.2f pF",CL2*10^12);
46
47 printf("\nCC3=%.2f pF",CC3*10^12);
```

Scilab code Exa 4.14 Direct Synthesis of Switched Capacitor Band Pass Filter

```
1 //Example 4.14
2
3 clear;
4
```

```
5  clc;
6
7  f0=1*10^3;
8
9  BW=600;
10
11 fck=100*10^3;
12
13 C1=0.84304;
14
15 L2=0.62201;
16
17 BWnorm=BW/f0;
18
19 C1norm=C1/BWnorm;
20
21 L1norm=BWnorm/C1;
22
23 L2norm=L2/BWnorm;
24
25 C2norm=BWnorm/L2;
26
27 Rs=1;
28
29 Ri=Rs;
30
31 Ro=Rs;
32
33 C0=1*10^(-12);
34
35 CRi=C0;
36
37 CRo=C0;
38
39 CC1=((fck*C1norm)/(2*%pi*f0))*C0;
40
41 CL1=((fck*L1norm)/(2*%pi*f0))*C0;
42
```

```
43 CC2=((fck*C2norm)/(2*%pi*f0))*C0;
44
45 CL2=((fck*L2norm)/(2*%pi*f0))*C0;
46
47 printf("Designed Switched Capacitor Band Pass Filter
       :");
48
49 printf("\nRi=R0=Rs=%.2f ohms",Rs);
50
51 printf("\nCRi=CRo=C0=%.2f pF",C0*10^12);
52
53 printf("\nCC1=%.2f pF",CC1*10^12/C1norm);
54
55 printf("\nC1=%.2f pF",CC1*10^12)
56
57 printf("\nCL1=%.2f pF",CL1*10^12);
58
59 printf("\nCC2=%.2f pF",(CC2*10^12)-0.54);
60
61 printf("\nCL2=%.2f pF",CL2*10^12);
```

Chapter 5

Static Op Amp Limitations

Scilab code Exa 5.1.a Errors caused by Input Bias and Offset Current

```
1  //Example 5.1(a)
2
3  clear;
4
5  clc;
6
7  R1=22*10^3;
8
9  R2=2.2*10^6;
10
11 IB=80*10^(-9);
12
13 IOS=20*10^(-9);
14
15 Rp=0;
16
17 dcgain=(1+(R2/R1));
18
19 R=(R1*R2)/(R1+R2);
20
21 Ip=((2*IB)+IOS)/2;
```

```
22
23 In=((2*IB)-IOS)/2;
24
25 Eo=dcgain*((R*IB));
26
27 printf("Eo=(+-)%.2f  mV",(Eo*10^3)-1);
```

Scilab code Exa 5.1.b Errors caused by Input Bias and Offset Current

```
1  //Example 5.1(b)
2
3  clear;
4
5  clc;
6
7  R1=22*10^3;
8
9  R2=2.2*10^6;
10
11 IB=80*10^(-9);
12
13 IOS=20*10^(-9);
14
15 Rp=(R1*R2)/(R1+R2);
16
17 dcgain=(1+(R2/R1));
18
19 R=(R1*R2)/(R1+R2);
20
21 Ip=((2*IB)+IOS)/2;
22
23 In=((2*IB)-IOS)/2;
24
25 Eo=dcgain*((R*In)-(Rp*Ip));
26
```

```
27 printf("Eo=(+-)%.f mV",-Eo*10^3);
```

Scilab code Exa 5.1.c Errors caused by Input Bias and Offset Current

```
1  //Example 5.1(c)
2
3  clear;
4
5  clc;
6
7  R1=22*10^2;
8
9  R2=2.2*10^5;
10
11 IB=80*10^(-9);
12
13 IOS=20*10^(-9);
14
15 Rp=(R1*R2)/(R1+R2);
16
17 dcgain=(1+(R2/R1));
18
19 R=(R1*R2)/(R1+R2);
20
21 Ip=((2*IB)+IOS)/2;
22
23 In=((2*IB)-IOS)/2;
24
25 Eo=dcgain*((R*In)-(Rp*Ip));
26
27 printf("Eo=(+-)%.2f mV",-Eo*10^3);
```

Scilab code Exa 5.1.d Errors caused by Input Bias and Offset Current

```
1  //Example 5.1(d)
2
3  clear;
4
5  clc;
6
7  R1=22*10^2;
8
9  R2=2.2*10^5;
10
11 IB=80*10^(-9);
12
13 IOS=3*10^(-9);
14
15 Rp=(R1*R2)/(R1+R2);
16
17 dcgain=(1+(R2/R1));
18
19 R=(R1*R2)/(R1+R2);
20
21 Ip=((2*IB)+IOS)/2;
22
23 In=((2*IB)-IOS)/2;
24
25 Eo=dcgain*((R*In)-(Rp*Ip));
26
27 printf("Eo=(+-)%.1f  mV",-Eo*10^3);
```

Scilab code Exa 5.2.a Errors caused by Input Bias and Offset Current II

```
1  //Example 5.2(a)
2
3  clear;
4
5  clc;
```

```
6
7  R=100*10^3;
8
9  C=1*10^(-9);
10
11 vo0=0;
12
13 IB=80*10^(-9);
14
15 IOS=20*10^(-9);
16
17 Vsat=13;
18
19 Rp=0;
20
21 Ip=((2*IB)+IOS)/2;
22
23 In=((2*IB)-IOS)/2;
24
25 vo1=(R*IB)/(R*C);
26
27 t=Vsat/vo1;
28
29 printf("Time taken by the op amp to enter saturation
       (t)=%.4f sec",t);
```

Scilab code Exa 5.2.b Errors caused by Input Bias and Offset Current II

```
1  //Example 5.2(b)
2
3  clear;
4
5  clc;
6
7  R=100*10^3;
```

```
8
9  C=1*10^(-9);
10
11 vo0=0;
12
13 IB=80*10^(-9);
14
15 IOS=20*10^(-9);
16
17 Vsat=13;
18
19 Rp=R;
20
21 Ip=((2*IB)+IOS)/2;
22
23 In=((2*IB)-IOS)/2;
24
25 vo1=(R*IB)/(R*C);
26
27 t1=Vsat/vo1;
28
29 t=t1*(IB/IOS);
30
31 printf("Time taken by the op amp to enter saturation
       (t)=%.2f sec",t);
```

Scilab code Exa 5.3 Input Bias Current Drift

```
1  //Example 5.3
2
3  clear;
4
5  clc;
6
7  T0=25;
```

```
8
9  IBT0=1*10^(-12);
10
11 T=100;
12
13 IBT=IBT0*2^((T-T0)/10);
14
15 printf("IB(100degC)=%.2f nA",IBT*10^9);
```

Scilab code Exa 5.4.a Error in Input Offset due to CMRR

```
1  //Example 5.4(a)
2
3  clear;
4
5  clc;
6
7  R1=10*10^3;
8
9  R2=100*10^3;
10
11 CMRRdB=90;
12
13 CMRRrec=10^(-(CMRRdB/20));//Reciprocal of CMRR
14
15 delvi=10;
16
17 delvp=(R2/(R1+R2))*delvi;
18
19 delVos=CMRRrec*delvp;
20
21 dcgain=1+(R2/R1);
22
23 delvo=dcgain*delVos;
24
```

```
25 printf("Typical change in vo=%.2f mV",delvo*10^3);
```

Scilab code Exa 5.4.b Error in Input Offset due to CMRR

```
1  //Example 5.4(b)
2
3  clear;
4
5  clc;
6
7  R1=10*10^3;
8
9  R2=100*10^3;
10
11 CMRRdB=57;//refer curve of fig.5A.6 at 10 kHz
12
13 CMRRrec=10^(-(CMRRdB/20));//Reciprocal of CMRR
14
15 delvi=10;
16
17 delvp=(R2/(R1+R2))*delvi;
18
19 delVos=CMRRrec*delvp;
20
21 dcgain=1+(R2/R1);
22
23 delvo=dcgain*delVos;
24
25 printf("Typical change in vo=%.3f V",delvo);
```

Scilab code Exa 5.5 Error in Input Offset due to PSRR

```
1  //Example 5.5
```

```
2
3 clear;
4
5 clc;
6
7 R1=100;
8
9 R2=100*10^3;
10
11 delvs=0.1;
12
13 dcgain=1+(R2/R1);
14
15 PSRRremin=30*10^(-6);//Minnimum rating of the
     reciprocal of PSRR
16
17 PSRRremax=150*10^(-6);
18
19 delVosmin=delvs*PSRRremin;
20
21 delVosmax=delvs*PSRRremax;
22
23 delvomin=delVosmin*dcgain;
24
25 delvomax=delVosmax*dcgain;
26
27 printf("The output ripple is=%.2f mV(typical)",
     delvomin*10^3);
28
29 printf(" and %.2f mV(maximum) peak to peak",delvomax
     *10^3);
```

Scilab code Exa 5.6 Change of offset voltage with the Output Swing

```
1 //Example 5.6
```

```
2
3  clear;
4
5  clc;
6
7  atyp=10^5;//typical value of a
8
9  amin=10^4;//minimum value of a
10
11 TCVosavg=3*10^(-6);
12
13 CMRRdBtyp=100;//typical value of CMRR in dB
14
15 CMRRrectyp=10^(-CMRRdBtyp/20);
16
17 PSRRdBtyp=100;//typical value of PSRR in dB
18
19 PSRRrectyp=10^(-PSRRdBtyp/20);
20
21 CMRRdBmin=80;//minimum value of CMRR in dB
22
23 CMRRrecmax=10^(-CMRRdBmin/20);
24
25 PSRRdBmin=80;//minimum value of PSRR in dB
26
27 PSRRrecmax=10^(-PSRRdBmin/20);
28
29 Tmin=0;
30
31 Tmax=70;
32
33 Vs=15;
34
35 vpmin=-1;
36
37 vpmax=1;
38
39 vomin=-5;
```

```
40
41 vomax=5;
42
43 Troom=25;
44
45 delVos1=TCVosavg*(Tmax-Troom);
46
47 delVos2typ=vpmax*CMRRrectyp;
48
49 delVos2max=vpmax*CMRRrecmax;
50
51 delVos3typ=2*(0.05*Vs)*PSRRrectyp;
52
53 delVos3max=2*(0.05*Vs)*PSRRrecmax;
54
55 delVos4typ=vomax/atyp;
56
57 delVos4max=vomax/amin;
58
59 delVoswor=delVos1+delVos2max+delVos3max+delVos4max;
60
61 deVospro=((delVos1^2)+(delVos2typ^2)+(delVos3typ^2)
   +(delVos4typ^2))^(1/2);
62
63 printf("Worst Change in Vos=(+-)%.2f uV",delVoswor
   *10^6);
64
65 printf("\nThe most probable change in Vos=(+-)%.f uV
   ",deVospro*10^6);
```

Scilab code Exa 5.7 Input Offset Error Compensation using Internal Offset Nulling

```
1 //Example 5.7
2
```

```
3  clear;
4
5  clc;
6
7  As=-10;
8
9  Rpot=10*10^3;
10
11 Vpot=15;
12
13 EImax=15*10^(-3);
14
15 Vosmax=6*10^(-3);
16
17 Iosmax=200*10^(-9);
18
19 Rpmax=(EImax-Vosmax)/Iosmax;//Parallel Combination
     of R1 and R2
20
21 R1max=(abs(As)+1)*(Rpmax/abs(As));
22
23 R1=R1max-(2.5*10^3);//Standardising R1
24
25 R2=abs(As)*R1;
26
27 Rp=(R1*R2)/(R1+R2);
28
29 printf("R1=%.2f kohms",R1*10^(-3));
30
31 printf("\nR2=%.2f kohms",R2*10^(-3));
32
33 printf("\nRp=%.f kohms",Rp*10^(-3));
```

Scilab code Exa 5.8 Input Offset Error Compensation using External Offset Nulling I

```
1  //Example 5.8
2
3  clear;
4
5  clc;
6
7  As=-5;
8
9  Ri=30*10^3;
10
11 Vs=15;
12
13 R1=Ri;
14
15 R2=abs(As)*R1;
16
17 Rp=(R1*R2)/(R1+R2);
18
19 Vosmax=6*10^(-3);
20
21 Iosmax=200*10^(-9);
22
23 EImax=Vosmax+(Rp*Iosmax);
24
25 RA=1*10^3;
26
27 Rpc=Rp-RA;
28
29 EImaxs=EImax+(4*10^(-3));
30
31 RB=RA*(Vs/EImaxs);
32
33 RC=100*10^3;///Choosing RC=100 kohms
34
35 printf("R1=%.2f kohms",R1*10^(-3));
36
37 printf("\nR2=%.2f kohms",R2*10^(-3));
38
```

```
39 printf("\nRp=%.2f  kohms",Rpc*10^(-3));
40
41 printf("\nRA=%.2f  kohms",RA*10^(-3));
42
43 printf("\nRB=%.2f  Mohms",RB*10^(-6));
44
45 printf("\nRC=%.2f  kohms",RC*10^(-3));
```

Scilab code Exa 5.9.a Input Offset Error Compensation using External Offset Nulling II

```
1  //Example 5.9(a)
2
3  clear;
4
5  clc;
6
7  As=5;
8
9  Vs=15;
10
11 R1=25.5*10^3;//Assuming R1=25.5 kohms
12
13 R2=(As-1)*R1;
14
15 Rp=(R1*R2)/(R1+R2);
16
17 brec=As;//reciprocal of b
18
19 Vosmax=6*10^(-3);
20
21 Iosmax=200*10^(-9);
22
23 EImax=Vosmax+(Rp*Iosmax);
24
```

```
25 Eomax=brec*EImax;
26
27 Vx=Eomax/(-R2/R1);
28
29 Vxs=Vx-(2.5*10^(-3));
30
31 RA=100;
32
33 RB=RA*abs(Vs/Vxs);
34
35 RC=100*10^3;///Choosing RC=100 kohms
36
37 printf("R1=%.2f kohms",R1*10^(-3));
38
39 printf("\nR2=%.2f kohms",R2*10^(-3));
40
41 printf("\nRp=%.f kohms",Rp*10^(-3));
42
43 printf("\nRA=%.f ohms",RA);
44
45 printf("\nRB=%.f kohms",(RB*10^(-3))+0.66);
46
47 printf("\nRC=%.f kohms",RC*10^(-3));
```

Scilab code Exa 5.9.b Input Offset Error Compensation using External Offset Nulling II

```
1 //Example 5.9(b)
2
3 clear;
4
5 clc;
6
7 As=100;
8
```

```
Vs=15;

R2=100*10^3;//Assuming R1=25.5 kohms

R1o=R2/(As-1);

R1=909;

RA=R1o-R1;

Rp=(R1o*R2)/(R1o+R2);

brec=As;//reciprocal of b

Vosmax=6*10^(-3);

Iosmax=200*10^(-9);

EImax=Vosmax+(Rp*Iosmax);

Eomax=brec*EImax;

Vx=Eomax/(-R2/R1);

Vxs=Vx-(2.5*10^(-3));

RA=100;

RB=RA*abs(Vs/Vxs);

RC=100*10^3;///Choosing RC=100 kohms

printf("R1=%.f ohms",R1o);

printf("\nR2=%.2f kohms",R2*10^(-3));

printf("\nRp=%.f kohms",Rp*10^(-3));

```

```
47 printf("\nRA=%.f  ohms",RA+1);
48
49 printf("\nRB=%.f  kohms",(RB*10^(-3))+15.63);
50
51 printf("\nRC=%.f  kohms",RC*10^(-3));
```

Scilab code Exa 5.10 Input Offset Error Compensation in Multiple Op Amp Circuits

```
1  //Example 5.10
2
3  clear;
4
5  clc;
6
7  T=25;
8
9  Ib=75*10^(-9);
10
11 Ios=80*10^(-9);
12
13 Vos=100*10^(-6);
14
15 Vs=15;
16
17 R1=4.99*10^(3);
18
19 R2=365;
20
21 R3=4.99*10^3;
22
23 R4=499;
24
25 R5=499;
26
```

```
27 R6=20*10^3;
28
29 R7=19.6*10^3;
30
31 R8=100;
32
33 R9=100*10^3;
34
35 R10=1*10^3;
36
37 C=100*10^(-12);
38
39 EI1=Vos+(((R1*(R2+(R8/2)))/(R1+(R2+(R8/2))))*Ib);
40
41 EI2=EI1;
42
43 EI3=Vos+(((R4*R6)/(R4+R6))*Ios);
44
45 A=10^3;
46
47 Eo=(A*(EI1+EI2))+((R6/R4)*EI3);
48
49 Eos=Eo+64*10^(-3);
50
51 Vx=Eos;
52
53 RB=100*10^3;
54
55 RA=RB/abs(Vs/Vx);
56
57 RC=100*10^3;///Choosing RC=100 kohms
58
59 printf("RA=%.f kohms",RA*10^(-3));
60
61 printf("\nRB=%.f kohms",RB*10^(-3));
62
63 printf("\nRC=%.f kohms",RC*10^(-3));
```

Scilab code Exa 5.11 Absolute Maximum Ratings

```
1  //Example 5.11
2
3  clear;
4
5  clc;
6
7  Tmax=70;
8
9  T=100;
10
11 Iqmax=2.8*10^(-3);
12
13 VCC=15;
14
15 VEE=-15;
16
17 P1=(VCC-VEE)*Iqmax;
18
19 P=310*10^(-3);
20
21 Io=(P-P1)/VCC;
22
23 PC=5.6*10^(-3);
24
25 Pmax=P+((Tmax-T)*PC);
26
27 Io=(Pmax-P1)/VCC;
28
29 printf("Maximum Current at 100degC=%.1f mA",Io*10^3)
   ;
```

Scilab code Exa 5.12 Overload Protection Maximum Ratings

```
1  //Example 5.12
2
3  clear;
4
5  clc;
6
7  R6=27;
8
9  b14=250;
10
11 b15=b14;
12
13 Vbe15on=0.7;
14
15 IC14=Vbe15on/R6;
16
17 IB14=IC14/b14;
18
19 i=0.18*10^(-3);
20
21 IC15=i-IB14;
22
23 Isc=IC14+IC15;
24
25 printf("IC14=%.f mA",IC14*10^3);
26
27 printf("\nIB14=%.3f mA",IB14*10^3);
28
29 printf("\nIC15=%.f uA",IC15*10^6);
30
31 printf("\nIsc=%.f mA",Isc*10^3);
```

Chapter 6

Dynamic Op Amp Limitations

Scilab code Exa 6.1.a Closed Loop Response of Non Inverting Amplifier

```
1  //Example 6.1(a)
2
3  clear;
4
5  clc;
6
7  R1=2*10^3;
8
9  R2=18*10^3;
10
11 b=0.1;
12
13 fb=100*10^3;
14
15 emmax=0.01;
16
17 fmax=((((1/(1-emmax))^2)-1)*(fb^2))^(1/2);
18
19 printf("f<=%.1f kHz",fmax*10^(-3));
```

Scilab code Exa 6.1.b Closed Loop Response of Non Inverting Amplifier

```
1 //Example 6.1(b)
2
3 clear;
4
5 clc;
6
7 R1=2*10^3;
8
9 R2=18*10^3;
10
11 b=0.1;
12
13 fb=100*10^3;
14
15 efimax=5;
16
17 fmax=tan(efimax*%pi/180)*fb;
18
19 printf("f<=%.2f kHz",fmax*10^(-3));
```

Scilab code Exa 6.2.a Gain Bandwidth Tradeoff

```
1 //Chapter-6
2 //Page No.-265
3 //Example 6.2(a)
4 //Gain Bandwidth Tradeoff
5
6 A0dB=60;
7
8 A0=10^(A0dB/20);
```

```
9
10 ft=10^6;
11
12 fb=ft/A0;
13
14 A10=A0^(1/2);
15
16 A20=A10;
17
18 fb1=ft/A10;
19
20 fb2=fb1;
21
22 R1=1*10^3;
23
24 R2=(A10-1)*R1;
25
26 printf("Designed Audio Amplifier :");
27
28 printf("\nOperational Amplifier-1 :");
29
30 printf("\nR1=%.2f kohms",R1*10^(-3));
31
32 printf("\nR2=%.1f kohms",(R2*10^(-3))+0.3);
33
34 printf("\n\nOperational Amplifier-2 :");
35
36 printf("\nR1=%.2f kohms",R1*10^(-3));
37
38 printf("\nR2=%.1f kohms",(R2*10^(-3))+0.3);
```

Scilab code Exa 6.2.b Gain Bandwidth Tradeoff

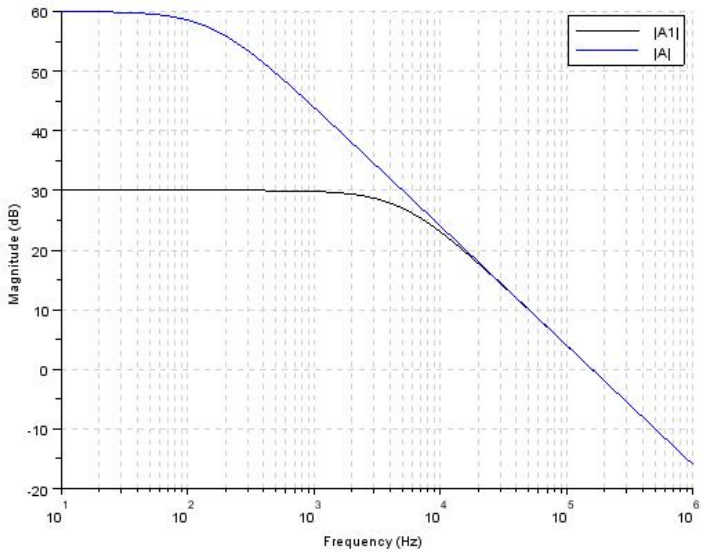

Figure 6.1: Gain Bandwidth Tradeoff

```
1  //Example 6.2(b)
2
3  clear;
4
5  clc;
6
7  A0dB=60;
8
9  A0=10^(A0dB/20);
10
11 ft=10^6;
12
13 fb=ft/A0;
14
15 A10=A0^(1/2);
16
17 A20=A10;
18
19 fb1=ft/A10;
20
21 fb2=fb1;
22
23 f1=1+(%s/fb1);
24
25 A1=A10*(1/f1);
26
27 y1=syslin('c',A1);
28
29
30 f2=1+(%s/fb);
31
32 A=A0*(1/f2);
33
34 y2=syslin('c',A);
35
36 gainplot([y1;y2],10,10^6,['|A1|';'|A|']);
```

Scilab code Exa 6.2.c Gain Bandwidth Tradeoff

```
1  //Example 6.2(c)
2
3  clear;
4
5  clc;
6
7  A0dB=60;
8
9  A0=10^(A0dB/20);
10
11 ft=10^6;
12
13 fb=ft/A0;
14
15 A10=A0^(1/2);
16
17 A20=A10;
18
19 fb1=ft/A10;
20
21 fb2=fb1;
22
23 f1=1+(%s/fb1);
24
25 A1=A10*(1/f1);
26
27 fB=(((((A10^2)*(2^(0.5)))/A0)-1)^(1/2))*fb1;
28
29 printf("Actual Bandwidth (fB)=%.2f kHz",fB*10^(-3));
```

Scilab code Exa 6.4 Input Impedance of Series Topology

```
1  //Example 6.4
2
3  clear;
4
5  clc;
6
7  rd=1*10^6;
8
9  rc=1*10^9;
10
11 a0=10^5;
12
13 ro=100;
14
15 ft=1*10^6;
16
17 R1=2*10^3;
18
19 R2=18*10^3;
20
21 b=R1/(R1+R2);
22
23 fB=b*ft;
24
25 Rs=rd;
26
27 Rd=rd*(1+(a0*b));
28
29 Rp=((2*rc)*Rd)/((2*rc)+Rd);
30
31 Ceq=1/(2*%pi*fB*rd);
32
33 f1=(Rs/Rp)*fB;
34
35 printf("Element Values in the Equivalent Circuit of
     Zi :");
```

```
36
37 printf("\nRs=%.2f Mohms",Rs*10^(-6));
38
39 printf("\nRp=%.2f Gohms",Rp*10^(-9));
40
41 printf("\nCeq=%.2f pF",Ceq*10^12);
42
43 printf("\n\nBreakpoint Frequencies of Magnitude Plot
       :");
44
45 printf("\nfB=%.2f kHz",fB*10^(-3));
46
47 printf("\nf1=%.2f Hz",f1);
```

Scilab code Exa 6.5 Output Impedance of Shunt Topology

```
1 //Example 6.5
2
3 clear;
4
5 clc;
6
7 rd=1*10^6;
8
9 rc=1*10^9;
10
11 a0=10^5;
12
13 ro=100;
14
15 ft=1*10^6;
16
17 R1=2*10^3;
18
19 R2=18*10^3;
```

```
20
21 b=R1/(R1+R2);
22
23 fb=ft/a0;
24
25 fB=b*ft;
26
27 Rp=ro;
28
29 Rs=ro/(1+(a0*b));
30
31 Leq=ro/(2*%pi*fB);
32
33 printf("Element Values in the Equivalent Circuit of
     Zo :");
34
35 printf("\nRs=%.f mohms",Rs*10^(3));
36
37 printf("\nRp=%.2f ohms",Rp);
38
39 printf("\nLeq=%.f uH",Leq*10^6);
40
41 printf("\n\nBreakpoint Frequencies of Magnitude Plot
     :");
42
43 printf("\nfb=%.2f Hz",fb);
44
45 printf("\nft=%.2f MHz",ft*10^(-6));
```

Scilab code Exa 6.6.a Finding Gain Zi and Zo for High Sensitivty I V Converter

```
1 //Example 6.6(a)
2
3 clear;
```

```
4
5  clc;
6
7  R=100*10^3;
8
9  R1=2*10^3;
10
11 R2=18*10^3;
12
13 b=R1/(R1+R2);
14
15 A0=-(1+(R2/R1))*R;
16
17 a0=2*10^5;
18
19 ft=1*10^6;
20
21 ro=100;
22
23 fB=b*ft;
24
25 Ri=[R+((R1*R2)/(R1+R2))]/(1+(a0*b));
26
27 Ro=ro/(1+(a0*b));
28
29 fb=ft/a0;
30
31 printf("A(jf)=(%d V/A)",A0);
32
33 printf("/(1+(jf/%.d))",fB);
34
35 printf("\nZi(jf)=%.d",Ri);
36
37 printf("*(1+j(f/%.d))",fb);
38
39 printf("/(1+(jf/%.d)) ohms",fB);
40
41 printf("\nZo(jf)=%.d",Ro*10^3);
```

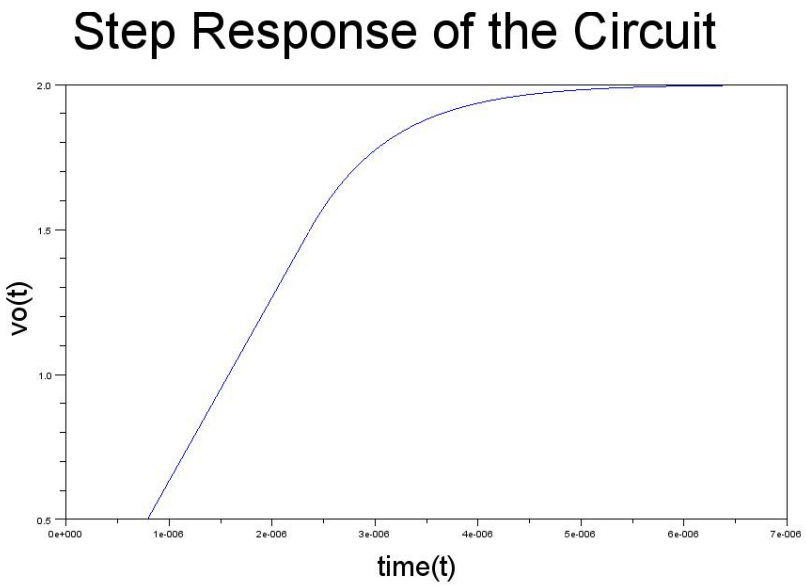

Figure 6.2: Effect of Slew Rate Limiting

```
42
43 printf("*(1+j(f/%.d))",fb);
44
45 printf("/(1+(jf/%.d)) mohms",fB);
```

Scilab code Exa 6.7.a Effect of Slew Rate Limiting

```
1 //Example 6.7(a)
2
3 clear;
4
5 clc;
6
7 IA=19.6*10^(-6);
8
```

```
Cc=30*10^(-12);

SR=0.633*10^6;

R1=3*10^3;

R2=12*10^3;

A0=-(R2/R1);

b=R1/(R1+R2);

a0=2*10^5;

ft=1*10^6;

ro=100;

Vim=-0.5;

tau=1/(2*%pi*b*ft);

Vomcrit=SR*tau;

Voinf=A0*Vim;

V1=Voinf-Vomcrit;

t=[0:2*10^(-8):7*10^(-6)];

t1=V1/SR;

t12=[0:2*10^(-8):tau]

vo3=0;

plot(t12,vo3);

```

```
47 t11=[tau:2*10^(-8):t1+tau];
48
49 vo1=SR*(t11-tau);
50
51 t22=[t1+tau:2*10^(-8):7*10^(-6)];
52
53 vo2=Voinf+((V1-Voinf)*exp(-(t22-t1-tau)/tau));
54
55 plot(t11,vo1);
56
57 plot(t22,vo2);
58
59 xlabel("time(t)","fontsize", 6);
60
61 ylabel("vo(t)","fontsize", 6);
62
63 title("Step Response of the Circuit","fontsize", 8);
```

Scilab code Exa 6.8.a Full Power Bandwidth

```
1 //Example 6.8(a)
2
3 clear;
4
5 clc;
6
7 Vs=15;
8
9 A=10;
10
11 Vim=0.5;
12
13 SR=0.5*10^6;
14
15 Vom=A*Vim;
```

```
16
17 fmax=SR/(2*%pi*Vom);
18
19 printf("fmax=%.f kHz",fmax*10^(-3));
```

Scilab code Exa 6.8.b Full Power Bandwidth

```
1 //Example 6.8(b)
2
3 clear;
4
5 clc;
6
7 Vs=15;
8
9 A=10;
10
11 f=10*10^3;
12
13 SR=0.5*10^6;
14
15 Vommax=SR/(2*%pi*f);
16
17 Vimmax=Vommax/A;
18
19 printf("Maximum Value of Vim before the output
   distorts=%.3f V",Vimmax);
```

Scilab code Exa 6.8.c Full Power Bandwidth

```
1 //Example 6.8(c)
2
3 clear;
```

```
4
5  clc;
6
7  Vs=15;
8
9  A=10;
10
11 Vim=40*10^(-3);
12
13 SR=0.5*10^6;
14
15 fmax=SR/(2*%pi*Vim*A);
16
17 ft=1*10^6;
18
19 fB=ft/A;
20
21 printf("Useful Frequency Range of Operation f<=%.2f
      kHz",fB*10^(-3));
```

Scilab code Exa 6.8.d Full Power Bandwidth

```
1  //Example 6.8(d)
2
3  clear;
4
5  clc;
6
7  Vs=13;
8
9  A=10;
10
11 ft=1*10^6;
12
13 SR=0.5*10^6;
```

```
14
15 f=2*10^3;
16
17 Vommax=SR/(2*%pi*f);
18
19 if Vommax>Vs then
20 Vimmax=Vs/A;
21
22 printf("Useful Input Amplitude Range is Vim<=%.2f V"
   ,Vimmax);
```

Scilab code Exa 6.9 Effect of finite GBP on Integrator Circuits

```
1 //Example 6.9
2
3 clear;
4
5 clc;
6
7 f0=10*10^3;
8
9 Q=25;
10
11 HobpdB=0;
12
13 R1=10*10^3;//Assumption
14
15 R2=R1;//Assumption
16
17 R5=R1;//Assumption
18
19 R6=R1;//Assumption
20
21 R3=250*10^3;//Assumption
22
```

```
23 R4=R3;//Assumption
24
25 C1=1/(2*%pi*f0*R5);//Assumption
26
27 C2=C1;//Assumption
28
29 f0reler=0.01;//as relative error defined for f0=1%
30
31 Qreler=0.01
32
33 ftf0=f0/f0reler;
34
35 ftQ=(4*Q*f0)/Qreler;
36
37 printf("Designed Biquad Filter :")
38
39 printf("\nR1=R2=R5=R6=%.2f kohms",R1*10^(-3));
40
41 printf("\nR3=R4=%.2f kohms",R4*10^(-3));
42
43 printf("\nC1=C2=%.2f nF",C1*10^9);
44
45 if ftf0>ftQ then
46     ft=ftf0;
47
48 else ft=ftQ
49
50 printf("\nGBP>=%.2f MHz",ft*10^(-6));
```

Scilab code Exa 6.10.b Biquad Filter with Phase Compensation

```
1 //Example 6.10(b)
2
3 clear;
4
```

```
5  clc;
6
7  f0=10*10^3;
8
9  Q=25;
10
11 HobpdB=0;
12
13 R1=10*10^3;//Assumption
14
15 R2=R1;//Assumption
16
17 R5=R1;//Assumption
18
19 R6=R1;//Assumption
20
21 R3=250*10^3;//Assumption
22
23 R4=R3;//Assumption
24
25 C1=1/(2*%pi*f0*R5);//Assumption
26
27 C2=C1;//Assumption
28
29 f0reler=0.01;//as relative error defined for f0=1%
30
31 Qreler=0.01
32
33 ftf0=f0/f0reler;
34
35 ftQ=(4*Q*f0)/Qreler;
36
37 ft=1*10^6;
38
39 //Changing the component values using Phase
     Compensation
40
41 ch=f0/ft;
```

```
42
43 C1new=C1-(C1*ch);
44
45 C2new=C1new;
46
47 printf("Designed Biquad Filter :")
48
49 printf("\nR1=R2=R5=R6=%.2f kohms",R1*10^(-3));
50
51 printf("\nR3=R4=%.2f kohms",R4*10^(-3));
52
53 printf("\nC1=C2=%.3f nF",C1new*10^9);
```

Scilab code Exa 6.11 Effect of finite GBP on first order filter

```
1 //Example 6.11
2
3 clear;
4
5 clc;
6
7 C=(5/%pi)*10^(-9);
8
9 R1=10*10^3;
10
11 R2=30*10^3;
12
13 GBP=1*10^6;
14
15 Hreler=0.01;//Departure of H from Hideal
16
17 ft=1*10^6;
18
19 fx=ft/(1+(R2/R1));
20
```

```
21 fmax=((1/((1-Hreler)^2)-1)^(1/2))*fx;
22
23 f0=1/(2*%pi*R1*C);
24
25 fmin3dB=(1/((1/(f0^2))-(1/(fx^2))-(1/((f0^2)*(fx^2))
   )))^(1/2);//f(-3dB)
26
27 f3dBer=((fmin3dB-f0)/fmin3dB)*100;
28
29 printf("Percentage Deviation of cut off frequency=%
   .2f",f3dBer*2);
```

Scilab code Exa 6.12 Effect of finite GBP on second order filter

```
1 //Example 6.12
2
3 clear;
4
5 clc;
6
7 C=10*10^(-9);
8
9 H0bpdB=0;
10
11 f0=10*10^3;
12
13 Q=10;
14
15 H0bp=10^(H0bpdB/20);
16
17 R1=Q/(2*%pi*f0*C*H0bp);
18
19 R2=(R1/((2*(Q^2))/(H0bp)))-1;
20
21 R3=(2*Q)/(2*%pi*f0*C);
```

```
22
23 BW=f0/Q;
24
25 BWer=0.01;//BW deviation from its design value is 1%
26
27 GBPmin=(2*Q*f0)/BWer;
28
29 printf("Components for the mentioned circuit :");
30
31 printf("\nR1=%.2f kohms",R1*10^(-3));
32
33 printf("\nR2=%.2f ohms",R2);
34
35 printf("\nR3=%.2f kohms",R3*10^(-3));
36
37 printf("\nGBP>=%.2f MHz",GBPmin*10^(-6));
```

Scilab code Exa 6.14 Parameters for Current Feedback Amplifier

```
1 //Example 6.14
2
3 clear;
4
5 clc;
6
7 zo=0.71*10^6;
8
9 Req=zo;
10
11 fb=350*10^3;
12
13 Ceq=1/(2*%pi*Req*fb);
14
15 vo=5;
16
```

```
17 iN=vo/Req;
18
19 printf("Ceq=%.2f  pF",Ceq*10^12);
20
21 printf("\niN=%.2f  uA",iN*10^6);
```

Scilab code Exa 6.15 Current Feedback Amplifier Dynamics

```
1  //Example 6.15
2
3  clear;
4
5  clc;
6
7  ft=100*10^6;
8
9  brec=1.5*10^3;
10
11 R2=1.5*10^3;
12
13 rn=50;
14
15 A01=1;
16
17 A02=10;
18
19 A03=100;
20
21 //R11=R2/(A01-1) ->R1=infinity
22
23 R12=R2/(A02-1);
24
25 R13=R2/(A03-1);
26
27 fB1=ft/(1+(A01/30));
```

```
28
29 fB2=ft/(1+(A02/30));
30
31 fB3=ft/(1+(A03/30));
32
33 tR1=2.2/(2*%pi*fB1);
34
35 tR2=2.2/(2*%pi*fB2);
36
37 tR3=2.2/(2*%pi*fB3);
38
39 printf("Values of R1, fB and tR for A0=1 :")
40
41 printf("\nR1=infinity");
42
43 printf("\nfB=%.2f MHz",fB1*10^(-6));
44
45 printf("\ntR=%.2f nS",tR1*10^9);
46
47 printf("\n\nValues of R1, fB and tR for A0=10 :")
48
49 printf("\nR1=%.2f ohms",R12);
50
51 printf("\nfB=%.2f MHz",fB2*10^(-6));
52
53 printf("\ntR=%.2f nS",tR2*10^9);
54
55 printf("\n\nValues of R1, fB and tR for A0=100 :")
56
57 printf("\nR1=%.2f ohms",R13);
58
59 printf("\nfB=%.2f MHz",fB3*10^(-6));
60
61 printf("\ntR=%.2f nS",tR3*10^9);
```

Scilab code Exa 6.16 Compensation of B W Reduction in Current Feedback Amplifier

```
1 //Example 6.16
2
3 clear;
4
5 clc;
6
7 A0=10;
8
9 fB=100*10^6;
10
11 brec=1.5*10^3;
12
13 rn=50;
14
15 R2=brec-(rn*A0);
16
17 R1=R2/(A0-1);
18
19 printf("(a) Redisigned Current Feedback Amplifier of
     Example 6.15 :");
20
21 printf("\n     R1=%.f ohms",R1);
22
23 printf("\n     R2=%.2f kohms",R2*10^(-3));
24
25 z0=0.75*10^6;
26
27 T0=(1/brec)*z0;
28
29 epsilon=-100/T0;
30
31 printf("\n\n(b) Percentage dc gain error=%.1f",
    epsilon);
```

Chapter 7

Noise

Scilab code Exa 7.1.a Noise Properties

```
1  //Example 7.1(a)
2
3  clear;
4
5  clc;
6
7  fL=0.1;
8
9  fH=100;
10
11 enw=20*10^(-9);
12
13 fce=200;
14
15 En=enw*sqrt((fce*log(fH/fL))+fH-fL);
16
17 printf("Estimated RMS input voltage=%.2f uV",En
     *10^6);
```

Scilab code Exa 7.1.b Noise Properties

```
1  //Example 7.1(b)
2
3  clear;
4
5  clc;
6
7  fL=20;
8
9  fH=20*10^3;
10
11 enw=20*10^(-9);
12
13 fce=200;
14
15 En=enw*sqrt((fce*log(fH/fL))+fH-fL);
16
17 printf("Estimated RMS input voltage=%.2f uV",En
     *10^6);
```

Scilab code Exa 7.1.c Noise Properties

```
1  //Example 7.1(c)
2
3  clear;
4
5  clc;
6
7  fL=0.1;
8
9  fH=1*10^6;
10
11 enw=20*10^(-9);
12
```

```
13 fce=200;
14
15 En=enw*sqrt((fce*log(fH/fL))+fH-fL);
16
17 printf("Estimated RMS input voltage=%.1f uV",En
     *10^6);
```

Scilab code Exa 7.3 Graphical Representation of Noise Dynamics

```
1  //Example 7.3
2
3  clear;
4
5  clc;
6
7  fL1=1;
8
9  fH1=1*10^3;
10
11 fL2=fH1;
12
13 fH2=10*10^3;
14
15 fL3=fH2;
16
17 //fH3=infinity
18
19 enw=20*10^(-9);
20
21 fce=100;
22
23 Eno1=enw*sqrt((fce*log(fH1/fL1))+fH1-fL1);
24
25 eno=enw/fL2;
26
```

```
27 Eno2=sqrt(integrate("(eno*f)^2",'f',fL2,fH2));
28
29 f0=100*10^3;
30
31 enw3=200*10^(-9);
32
33 Eno3=enw3*sqrt((1.57*f0)-fL3);
34
35 Eno=sqrt((Eno1^2)+(Eno2^2)+(Eno3^2));
36
37 printf("Estimated rms noise voltage=%.1f uV",Eno
   *10^6);
```

Scilab code Exa 7.4 Calculation of Thermal Noise

```
1 //Example 7.4
2
3 clear;
4
5 clc;
6
7 R=10*10^3;
8
9 k=1.38*10^(-23);
10
11 T=25+273;//Room Temperature in Kelvin
12
13 eR=sqrt(4*k*T*R);
14
15 printf("(a) Noise Voltage(eR)=%.2f nV/sqrt(Hz)",eR
   *10^9);
16
17 iR=eR/R;
18
19 printf("\n(b) Noise Current(iR)=%.2f pA/sqrt(Hz)",iR
```

```
   *10^12);
20
21 fH=20*10^3;
22
23 fL=20;
24
25 ER=eR*sqrt(fH-fL);
26
27 printf("\n(c) rms noise voltage over audio range=%.2
   f uV",ER*10^6);
```

Scilab code Exa 7.5.a Calculation of Shot Noise

```
1 //Example 7.5(a)
2
3 clear;
4
5 clc;
6
7 ID=1*10^(-6);
8
9 fH=1*10^6;
10
11 q=1.602*10^(-19);
12
13 In=sqrt(2*q*ID*fH);
14
15 SNR=20*log10(ID/In);
16
17 printf("Signal to Noise Ratio=%.1f dB",SNR);
```

Scilab code Exa 7.5.b Calculation of Shot Noise

```
1  //Example 7.5(b)
2
3  clear;
4
5  clc;
6
7  ID=1*10^(-9);
8
9  fH=1*10^6;
10
11 q=1.602*10^(-19);
12
13 In=sqrt(2*q*ID*fH);
14
15 SNR=20*log10(ID/In);
16
17 printf("Signal to Noise Ratio=%.1f dB",SNR);
```

Scilab code Exa 7.7.a Total Output Noise in an Op Amp

```
1  //Example 7.7(a)
2
3  clear;
4
5  clc;
6
7  R1=100*10^3;
8
9  R2=200*10^3;
10
11 R3=68*10^3;
12
13 enw=20*10^(-9);
14
15 fce=200;
```

```
16
17 ft=1*10^6;
18
19 inw=0.5*10^(-12);
20
21 fci=2*10^3;
22
23 Rp=(R1*R2)/(R1+R2);
24
25 Ano=1+(R2/R1);
26
27 fB=ft/Ano;
28
29 fL=0.1;
30
31 Enoe=Ano*enw*sqrt([{fce*log(fB/fL)}+{1.57*fB}-fL]);
32
33 Enoi=Ano*[{(R3^2)+(Rp^2)}^(1/2)]*inw*([(fci*log(fB/
     fL))+(1.57*fB)]^(1/2));
34
35 k=1.38*10^(-23);
36
37 T=25+273;//Room temperature in Kelvin
38
39 EnoR=Ano*[{(4*k*T)*(R3+Rp)*1.57*fB}^(1/2)];
40
41 Eno=sqrt((Enoe^2)+(Enoi^2)+(EnoR^2));
42
43 printf("RMS Output Noise Voltage=%.f uV",Eno*10^6);
44
45 printf("\nPeak to Peak Noise Voltage=%.2f mV",6.6*
     Eno*10^3);
```

Scilab code Exa 7.8 Improvement in the Circuit to find the Total Output Noise

```
1  //Example 7.8
2
3  clear;
4
5  clc;
6
7  R1=100*10^3;
8
9  R2=200*10^3;
10
11 R3=68*10^3;
12
13 enw=20*10^(-9);
14
15 fce=200;
16
17 ft=1*10^6;
18
19 inw=0.5*10^(-12);
20
21 fci=2*10^3;
22
23 Rp=(R1*R2)/(R1+R2);
24
25 Ano=1+(R2/R1);
26
27 fB=ft/Ano;
28
29 fL=0.1;
30
31 Enoeold=Ano*enw*sqrt([{fce*log(fB/fL)}+{1.57*fB}-fL
      ]);
32
33 Enoiold=Ano*[{(R3^2)+(Rp^2)}^(1/2)]*inw*([(fci*log(
      fB/fL))+(1.57*fB)]^(1/2));
34
35 k=1.38*10^(-23);
36
```

```
37 T=25+273;//Room temperature in Kelvin
38
39 EnoRold=Ano*[{(4*k*T)*(R3+Rp)*1.57*fB}^(1/2)];
40
41 Enoold=sqrt((Enoeold^2)+(Enoiold^2)+(EnoRold^2));
42
43 Enonew=50*10^(-6);//New Value of Eno mentioned in
      problem
44
45 Enoisum=(Enonew^2)-(Enoeold^2);//sum of (Enoi^2) and
      (EnoR^2)
46
47 Enoinewsq=(Ano^2)*(inw^2)*[(fci*log(fB/fL))+(1.57*fB
   )];//(Enoinew^2)/(R^2)
48
49 EnoRnewsq=(Ano^2)*((4*k*T)*1.57*fB);
50
51 r=poly(0,'x');
52
53 p=(Enoinewsq*(r^2))+(EnoRnewsq*r)-Enoisum;
54
55 [r1]=roots(p);
56
57 R=r1(2,1)
58
59 R3new=R/2;
60
61 R1new=(3*R3new)/2;
62
63 R2new=2*R1new;
64
65 printf("Resistances after scaling are :");
66
67 printf("\nR1=%.2f kohms",R1new*10^(-3));
68
69 printf("\nR2=%.1f kohms",R2new*10^(-3));
70
71 printf("\nR3=%.1f kohms",R3new*10^(-3));
```

Scilab code Exa 7.9 Calculation of Signal to Noise Ratio

```
1 //Example 7.9
2
3 clear;
4
5 clc;
6
7 R1=100*10^3;//From Example 7.7(a)
8
9 R2=200*10^3;//From Example 7.7(a)
10
11 Aso=-(R2/R1);
12
13 Eno=154*10^(-6);//From Example 7.9
14
15 Eni=Eno/abs(Aso);
16
17 Vipa=0.5;//Peak amplitude of input ac signal
18
19 Virms=Vipa/sqrt(2);
20
21 SNR=20*log10(Virms/Eni);
22
23 printf("SNR of the circuit of Example 7.7=%.1f dB",
   SNR);
```

Scilab code Exa 7.10 Calculation of Noise in Current Feedback Amplifier

```
1
2 //Example 7.10
```

```
3
4  clear;
5
6  clc;
7
8  z0=710*10^3;
9
10 fb=350*10^3;
11
12 rn=50;
13
14 enw=2.4*10^(-9);
15
16 fce=50*10^3;
17
18 inpw=3.8*10^(-12);
19
20 fcip=100*10^3;
21
22 innw=20*10^(-12);
23
24 fcin=100*10^3;
25
26 R1=166.7;
27
28 R2=1.5*10^3;
29
30 R3=100;//internal resistance
31
32 fL=0.1;
33
34 Rp=(R1*R2)/(R1+R2);
35
36 ft=(z0*fb)/R2;
37
38 fB=ft/[1+(rn/((R1*R2)/(R1+R2)))];
39
40 Ano=1+(R2/R1);
```

```
41
42 Enoe=enw*sqrt([{fce*log(fB/fL)}+{1.57*fB}-fL]);
43
44 Enoi=R3*inpw*sqrt(((fcip*log(fB/fL))+(1.57*fB)-fL));
45
46 Enop=Rp*innw*sqrt({(fcin*log(fB/fL))+(1.57*fB)-fL});
47
48 k=1.38*10^(-23);
49
50 T=25+273;//Room temperature in Kelvin
51
52 EnoR=[{(4*k*T)*(R3+Rp)*((1.57*fB)-fL)}^(1/2)];
53
54 Eno=Ano*sqrt((Enoe^2)+(Enoi^2)+(EnoR^2)+(Enop^2));
55
56 c=6.6*10^3;
57
58 Eno1=Eno*c;
59
60 printf("RMS Noise Voltage (Eno)=%.2f uV",Eno*10^6);
   //answer in textbook is wrong
61
62 printf("\nPeak to Peak Noise Voltage (Eno)=%.2f mV",
   Eno1);//answer in textbook is wrong
```

Scilab code Exa 7.11 Noise in Photodiode Amplifiers

```
1 //Example 7.11
2
3 clear;
4
5 clc;
6
7 ft=16*10^6;
8
```

```
 9 enw=4.5*10^(-9);
10
11 fce=100;
12
13 IB=1*10^(-12);
14
15 fL=0.01;
16
17 R1=100*10^(9);
18
19 C1=45*10^(-12);
20
21 R2=10*10^6;
22
23 C2=0.5*10^(-12);
24
25 b0rec=1;
26
27 binfrec=91;
28
29 fz=350;
30
31 fp=31.8*10^3;
32
33 fx=176*10^3;
34
35 k=1.38*10^(-23);
36
37 T=25+273;
38
39 iR2=sqrt((4*k*T)/R2);
40
41 q=1.602*10^(-19);
42
43 in=sqrt(2*q*IB);
44
45 Enoe=binfrec*enw*sqrt(((%pi/2)*fx)-fp);
46
```

```
47 EnoR=R2*iR2*sqrt((%pi/2)*fp);
48
49 Eno=sqrt((Enoe^2)+(EnoR^2));
50
51 printf("Total Output Noise=%.f uV",Eno*10^6);
```

Scilab code Exa 7.12 Photodiode amplifier with Noise Filtering

```
1  //Example 7.12
2
3  clear;
4
5  clc;
6
7  ft=16*10^6;
8
9  enw=4.5*10^(-9);
10
11 fce=100;
12
13 IB=1*10^(-12);
14
15 fL=0.01;
16
17 R1=100*10^(9);
18
19 C1=45*10^(-12);
20
21 R2=10*10^6;
22
23 C2=0.5*10^(-12);
24
25 b0rec=1;
26
27 binfrec=91;
```

```
28
29 fz=350;
30
31 fp=31.8*10^3;
32
33 fx=176*10^3;
34
35 k=1.38*10^(-23);
36
37 T=25+273;
38
39 Cc=0.5*10^(-12);//Assumed
40
41 C2=Cc;
42
43 C3=10*10^(-9);
44
45 R3=(R2*Cc)/C3;
46
47 printf("Cc=C2=%.1f  pF",Cc*10^(12));
48
49 printf("\nR3=%.f  ohms",R3);
50
51 printf("\nC3=%.f  nF",C3*10^(9));
```

Scilab code Exa 7.13 Designing T Feedback Photodiode Amplifiers

```
1 //Example 7.13
2
3 clear;
4
5 clc;
6
7 C1=2*10^(-9);
8
```

```
9  binfrec=4000;
10
11 inw=0.566*10^(-15);
12
13 T=1*10^(9);
14
15 ft=16*10^6;
16
17 R1=100*10^(9);
18
19 C2=0.5*10^(-12);
20
21 fx=(1/binfrec)*ft;
22
23 enw=4.5*10^(-9);
24
25 Enoe=binfrec*enw*sqrt((%pi*fx)/2);
26
27 EnoRmax=Enoe/3;
28
29 k=1.38*10^(-23);
30
31 Temp=25+273;
32
33 ex=((EnoRmax^2)*C2)/(k*Temp);
34
35 R2=T/ex;
36
37 R3=1*10^3;//Assumed
38
39 R4=(ex-1)*R3;
40
41 printf("(a) Designed T Network :");
42
43 printf("\n      R1=%.2f Gohms",R1*10^(-9));
44
45 printf("\n      R2=%.1f Mohms",R2*10^(-6));
46
```

```
47 printf("\n      R3=%.2f  kohms",R3*10^(-3));
48
49 printf("\n      R4=%.2f  kohms",R4*10^(-3));
50
51 printf("\n      C1=%.2f  nF",C1*10^9);
52
53 printf("\n      C2=%.2f  pF",C2*10^12);
54
55 fp=1/(2*%pi*ex*R2*C2);
56
57 fB=fp;
58
59 Rp=(R1*R2)/(R1+R2);
60
61 Enoi=((1.57*fB)^(1/2))*inw;
62
63 Eno=sqrt((Enoe^2)+(Enoi^2)+(EnoRmax^2));
64
65 printf("\n\n(b) Total rms Output Noise=%.2f mV",Eno
      *10^3);
66
67 printf("\n      Bandwidth(fB)=%.d Hz",fB);
```

Chapter 8

Stability

Scilab code Exa 8.1 Gain Margin and Phase Margin of an op amp system

```
1  //Example 8.1
2
3  clear;
4
5  clc;
6
7  T0=10^4;
8
9  f1=100;
10
11 f2=10^6;
12
13 f3=10*10^6;
14
15 w1=2*%pi*f1;
16
17 w2=2*%pi*f2;
18
19 w3=2*%pi*f3;
20
21 h=syslin('c',T0/[(1-(%s/w1))*(1-(%s/w2))*(1-(%s/w3))
```

```
    ]);
22
23 gm=g_margin(h);
24
25 pm=p_margin(h);
26
27 printf("(a) Gain Margin(GM)=%.2f dB",gm);
28
29 printf("\n(b) Phase Margin(PM)=%.1f degrees",-pm);
30
31 f=512*10^3;
32
33 w=2*%pi*f;
34
35 T1=T0/[(1-((%i*w)/w1))*(1-((%i*w)/w2))*(1-((%i*w)/w3
    ))];
36
37 den=1/(abs(T1)/T0);
38
39 printf("\n(c) T0 for PM=60 degrees=%.f",den);
```

Scilab code Exa 8.2 Stability in Differentiator Circuits

```
1 //Example 8.2
2
3 clear;
4
5 clc;
6
7 R=159*10^3;
8
9 C=10*10^(-9);
10
11 f0=1/(2*%pi*R*C);
12
```

```
13 ft=10^6;
14
15 fx=sqrt(f0*ft);
16
17 Q=sqrt(ft/f0);
18
19 d=-90-((180/%pi)*atan(fx/f0));
20
21 pm=180+d;
22
23 printf("fx=%.2f kHz",fx*10^(-3));
24
25 printf("\nQ=%.f",Q);
26
27 printf("\nPhase Margin (PM)=%.1f degrees",pm);
```

Scilab code Exa 8.3 Stray Input Capacitance Compensation for inverting configuration

```
1 //Example 8.3
2
3 clear;
4
5 clc;
6
7 R1=30*10^3;
8
9 R2=R1;
10
11 Cext=3*10^(-12);
12
13 GBP=20*10^6;
14
15 Cd=7*10^(-12);
16
```

```
17 Cc=12*10^(-12);
18
19 Cn=Cext+Cd+(Cc/2);
20
21 Rp=(R1*R2)/(R1+R2);
22
23 Cf1=0;
24
25 fz1=1/(2*%pi*Rp*(Cn+Cf1));
26
27 ft=20*10^6;
28
29 Q=sqrt((ft)/(2*fz1));
30
31 pm=(180/%pi)*acos((sqrt(1+(1/(4*Q^4))))-(1/(2*Q^2)))
      ;
32
33 Cf2=(R1/R2)*Cn;
34
35 fp=1/(2*%pi*R2*Cf2);
36
37 x=poly(0,'f');
38
39 A=-1/[(1+(%i*(x/fp)))*(1+(%i*(x/(0.5*ft))))];
40
41 printf("(a) Phase Margin with Cf absent=%.1f degrees
      ",pm);
42
43 printf("\n(b) Cf for PM=90 degrees=%.2f pF",Cf2
      *10^12);
44
45 printf("\n(c) A(jf)=");
46
47 disp(A);
```

Scilab code Exa 8.4 Stray Input Capacitance Compensation for non inverting configuration

```
1  //Example 8.4
2
3  clear;
4
5  clc;
6
7  R1=30*10^3;
8
9  R2=R1;
10
11 ft=20*10^6;
12
13 Cext=3*10^(-12);
14
15 GBP=20*10^6;
16
17 Cd=7*10^(-12);
18
19 Cc=12*10^(-12);
20
21 Cf=(R1/R2)*((Cc/2)+Cext);
22
23 Cn=Cext+Cd+(Cc/2);
24
25 fx=ft/(1+(Cn/Cf));
26
27 x=poly(0,'f');
28
29 A=(1+(R2/R1))/(1+(%i*(x/fx)));
30
31 printf("A(jf)=");
32
33 disp(A);
34
35 printf("V/V");
```

Scilab code Exa 8.5 Stabalizing a capacitively loaded op amp circuit

```
1  //Example 8.5
2
3  clear;
4
5  clc;
6
7  GBP=10*10^6;
8
9  ro=100;
10
11 A0=-2;
12
13 CL=5*10^(-9);
14
15 R1=10*10^3;
16
17 R2=20*10^3;
18
19 Rs=(R1/R2)*ro;
20
21 Cf=((1+(R1/R2))^2)*(ro/R2)*CL;
22
23 f3dB=1/(2*%pi*R2*Cf);
24
25 b=1/3;
26
27 fx=b*GBP;
28
29 printf("(a) Rs=%.f ohms",Rs);
30
31 printf("\n      Cf=%.f pF",Cf*10^12);
32
```

```
33 x=poly(0,'f');
34
35 A=A0/((1+(%i*(x/fx)))*(1+(%i*(x/f3dB))));
36
37 printf("\n\n(b) A(jf)=");
38
39 disp(A);
40
41 printf("V/V");
```

Scilab code Exa 8.6 Internal Frequency Compensation

```
1  //Example 8.6
2
3  clear;
4
5  clc;
6
7  a0=3600;
8
9  f1=1*10^6;
10
11 f2=4*10^6;
12
13 f3=40*10^6;
14
15 fmin135=4.78*10^6;
16
17 fmin180=14.3*10^6;
18
19 gbp1=abs(a0/[(1+(%i*(fmin135/f1)))*(1+(%i*(fmin135/
      f3)))*(1+(%i*(fmin135/f3)))])-256;
20
21 gbp2=abs(a0/[(1+(%i*(fmin180/f1)))*(1+(%i*(fmin180/
      f3)))*(1+(%i*(fmin180/f3)))])-158.97561;
```

```
22
23 printf("|a(j*fmin135)|=%.d V/V",gbp1);
24
25 printf("\n|a(j*fmin180)|=%.1f V/V",gbp2);
```

Scilab code Exa 8.7 Dominant Pole Compensation

```
1  //Example 8.7
2
3  clear;
4
5  clc;
6
7  PM=45;
8
9  anganewjfx=-180+PM;
10
11 a0=3600;
12
13 f1=1*10^6;
14
15 f2=4*10^6;
16
17 f3=40*10^6;
18
19 angajfx=anganewjfx+90;
20
21 fx=683*10^3;
22
23 ajf=a0/((1+(%i*(fx/f1)))*(1+(%i*(fx/f2)))*(1+(%i*(fx
     /f3))));
24
25 ang=(180/%pi)*atan(imag(ajf)/real(ajf));
26
27 mag=abs(ajf);
```

```
28
29 fd=sqrt((fx^2)/((mag^2)-1));
30
31 printf("fd=%.f Hz",fd);
```

Scilab code Exa 8.8 Shunt Capacitance Compensation

```
1  //Example 8.8
2
3  clear;
4
5  clc;
6
7  rd=1*10^6;
8
9  g1=2*10^(-3);
10
11 R1=100*10^(3);
12
13 g2=10*10^(-3);
14
15 R2=50*10^3;
16
17 ro=100;
18
19 f1=100*10^3;
20
21 f2=1*10^6;
22
23 f3=10*10^3;
24
25 PM=45;
26
27 a0=g1*R1*g2*R2;
28
```

```
C1=1/(2*%pi*f1*R1);

b1=1;

f1new1=f2/(b1*a0);

Cc1=1/(2*%pi*R1*f1new1);

printf("(a)  fd=%.f  Hz",f1new1);

printf("\n      Cc=%.f  nF",Cc1*10^9);

b2=0.5;

f1new2=f2/(b2*a0);

Cc2=1/(2*%pi*R1*f1new2);

printf("\n\n(b)  fd=%.f  Hz",f1new2);

printf("\n      Cc=%.1f  nF",Cc2*10^9);
```

Scilab code Exa 8.9 Miller Compensation

```
//Example 8.9

clear;

clc;

rd=1*10^6;

g1=2*10^(-3);

R1=100*10^(3);
```

```
12
13 g2=10*10^(-3);
14
15 R2=50*10^3;
16
17 ro=100;
18
19 f1=100*10^3;
20
21 f2=1*10^6;
22
23 f3=10*10^6;
24
25 PM=45;
26
27 a0=g1*R1*g2*R2;
28
29 C1=1/(2*%pi*f1*R1);
30
31 b1=1;
32
33 C21=1/(2*%pi*f2*R2);
34
35 f2newap1=g2/[2*%pi*(C1+C21)];
36
37 fx1=f3;
38
39 f1new1=f3/(b1*a0);
40
41 Cc1=1/(2*%pi*R1*g2*R2*f1new1);
42
43 f2new1=(g2*Cc1)/(2*%pi*((C1*C21)+(Cc1*C1)+(Cc1*C21))
      );
44
45 fz1=g2/(2*%pi*Cc1);
46
47 printf("(a) f1new=%.f Hz",f1new1);
48
```

```
49 printf("\n      f2new=%.f  MHz",f2new1*10^(-6));
50
51 printf("\n      Cc=%.1f  pF",Cc1*10^12);
52
53 b2=0.5;
54
55 C22=1/(2*%pi*f2*R2);
56
57 f2newap2=g2/[2*%pi*(C1+C22)];
58
59 fx2=f3;
60
61 f1new2=f3/(b2*a0);
62
63 Cc2=1/(2*%pi*R1*g2*R2*f1new2);
64
65 f2new2=(g2*Cc2)/(2*%pi*((C1*C22)+(Cc2*C1)+(Cc2*C22))
     );
66
67 fz2=g2/(2*%pi*Cc2);
68
69 printf("\n\n(b)  f1new=%.f  Hz",f1new2);
70
71 printf("\n      f2new=%.f  MHz",f2new2*10^(-6));
72
73 printf("\n      Cc=%.1f  pF",Cc2*10^12);
```

Scilab code Exa 8.10 Pole Zero Compensation

```
1 //Example 8.10
2
3 clear;
4
5 clc;
6
```

```
7  PM=45;
8
9  b=1;
10
11 rd=1*10^6;
12
13 g1=2*10^(-3);
14
15 R1=100*10^(3);
16
17 g2=10*10^(-3);
18
19 R2=50*10^3;
20
21 ro=100;
22
23 f1=100*10^3;
24
25 f2=1*10^6;
26
27 f3=10*10^6;
28
29 a0=g1*R1*g2*R2;
30
31 C1=1/(2*%pi*f1*R1);
32
33 Cc=(b*a0)/(2*%pi*R1*f3);
34
35 Rc=1/(2*%pi*Cc*f2);
36
37 f4=1/(2*%pi*Rc*C1);
38
39 printf("Cc=%.1f  nF",Cc*10^9);
40
41 printf("\nRc=%.f  ohms",Rc);
42
43 printf("\nR1=%.f  kohms",R1*10^(-3));//The value of
     R1 is not provided in the textbook
```

```
44
45 printf("\nC1=%.2f pF",C1*10^12);//The value of R1 is
       not provided in the textbook
```

Scilab code Exa 8.11 Frequency Compensation via Loop Gain Reduction

```
1  //Example 8.11
2
3  clear;
4
5  clc;
6
7  a0=10^5;
8
9  f1=10*10^3;
10
11 f2=3*10^6;
12
13 f3=30*10^6;
14
15 R1=10*10^3;
16
17 R2=100*10^3;
18
19 PM=45;
20
21 ajf=a0/((1+(%i*(f2/f1)))*(1+(%i*(f2/f2)))*(1+(%i*(f2
       /f3))));
22
23 ajf2mag=abs(ajf);
24
25 Rc1=R2/(ajf2mag-(1+(R2/R1)));
26
27 printf("(a) Rc=%.1f ohms",Rc1);
28
```

```
29 Rc2=430;
30
31 brec=1+(R2/R1)+(R2/Rc2);
32
33 a0b=a0/brec;
34
35 dcge=-100/(a0b);
36
37 printf("\n\n(b) DC Gain Error=%.2f percent",dcge);
38
39 EI=1*10^(-3);
40
41 EO=brec*EI;
42
43 printf("\n\n(c) DC Output Error=%.f mV",EO*10^3);
44
45 fmin3dB=f2;
46
47 printf("\n\n(d) f-3dB=%.f MHz",fmin3dB*10^(-6));
```

Scilab code Exa 8.12 Input Lag Compensation

```
1 //Example 8.12
2
3 clear;
4
5 clc;
6
7 a0=10^5;
8
9 f1=10*10^3;
10
11 f2=3*10^6;
12
13 f3=30*10^6;
```

```
14
15 R1=10*10^3;
16
17 R2=100*10^3;
18
19 PM=45;
20
21 Rc=447.4;
22
23 Cc=5/(%pi*Rc*f2);
24
25 printf("(a) Rc=%.1f ohms",Rc);
26
27 printf("\n     Cc=%.3f nF",Cc*10^9);
28
29 b0rec=1+(R2/R1);
30
31 a0b0=a0*(1/b0rec);
32
33 dcge=-100/(a0b0);
34
35 printf("\n\n(b) DC Gain Error=%.3f percent",dcge);
36
37 EI=1*10^(-3);
38
39 EO=b0rec*EI;
40
41 printf("\n\n(c) DC Output Error=%.f mV",EO*10^3);
42
43 fmin3dB=f2;
44
45 printf("\n\n(d) f-3dB=%.f MHz",fmin3dB*10^(-6));
46
47 f=2.94*10^6;
48
49 T=(410*[1+(%i*(f/(0.1*f2)))])/[(1+((%i*f)/f1))*(1+((
      %i*f)/f2))*(1+((%i*f)/f3))*(%i*(f/(0.1*f2)))];
50
```

```
51 Tang=-(180-(180/%pi)*atan(imag(T)/real(T)));
52
53 PM1=180+Tang;
54
55 printf("\n\n(e) Actual Phase Margin=%.1f degrees",
     PM1);
```

Scilab code Exa 8.13 Feedback Lead Compensation

```
1  //Example 8.13
2
3  clear;
4
5  clc;
6
7  a0=10^5;
8
9  f1=1*10^3;
10
11 f2=100*10^3;
12
13 f3=5*10^6;
14
15 A0=20;
16
17 R1=1.05*10^3;
18
19 R2=20*10^3;
20
21 b0=1/(1+(R2/R1));
22
23 a0b0=a0*b0;
24
25 f=700*10^3;
26
```

```
27 T=a0b0/[(1+((%i*f)/f1))*(1+((%i*f)/f2))*(1+((%i*f)/
   f3))];
28
29 Tang=-(180-(180/%pi)*atan(imag(T)/real(T)));
30
31 PM=180+Tang;
32
33 printf("(a) PM=%.1f degrees indicating a circuit in
   bad need of compensation.",PM);
34
35 amod=sqrt(20);
36
37 aang=-192.3;
38
39 fx=1.46*10^6;
40
41 Cf=sqrt(1+(R2/R1))/(2*%pi*R2*fx);
42
43 PM1=180+aang-(90-(2*(180/%pi)*atan(sqrt(1+(R2/R1))))
   );
44
45 printf("\n\n(b) PM after compensation=%.1f degrees",
   PM1);
46
47 f3dB=(1/(2*%pi*R2*Cf))+1000;
48
49 printf("\n\n(c) f-3dB=%.f kHz",f3dB*10^(-3));
```

Scilab code Exa 8.14 Configuring a Decompensated op amp as a Unity Gain Voltage Follower

```
1 //Example 8.14
2
3 clear;
4
```

```
clc;

A0=1;

brecmin=5;

Rc=3*10^3;

Rf=Rc*(brecmin-1);

GBP=20*10^6;

fx=(1/brecmin)*GBP;

Cc=brecmin/(%pi*Rc*fx);

printf("(a) Rc=%.f kohms",Rc*10^(-3));

printf("\n      Rf=%.f kohms",Rf*10^(-3));

printf("\n      Cc=%.f pF",Cc*10^12);

printf("\n\n(b) A(jf)=1/[1+jf/(%.f MHz)] V/V",fx
   *10^(-6));
```

Scilab code Exa 8.15 Input Stray Capacitance Compensation in CFA Circuits

```
//Example 8.15

clear;

clc;

zo=750*10^3;
```

```
8
9  fb=200*10^3;
10
11 rn=50;
12
13 R2=1.5*10^3;
14
15 Cn=100*10^(-12);
16
17 PM=45;
18
19 Cf=sqrt((rn*Cn)/(2*%pi*R2*zo*fb));
20
21 printf("Cf=%.2f  pF",Cf*10^12);
```

Scilab code Exa 8.16 Feedback Lead Compensation for Composite Amplifier

```
1  //Example 8.16
2
3  clear;
4
5  clc;
6
7  R1=1*10^3;
8
9  R2=99*10^3;
10
11 PM=45;
12
13 ft1=1*10^6;
14
15 ft2=ft1;
16
17 Cf=sqrt((1+(R2/R1))/(ft1*ft2))/(2*%pi*R2);
```

```
18
19 a0=2*10^5;
20
21 T0=(a0^2)/100;
22
23 fp=(1/(2*%pi*R2*Cf));
24
25 fB=fp;
26
27 PMs=PM*2;
28
29 T0s=a0/100;
30
31 fBs=ft1/100;
32
33 printf("(a) Composite Amplifier with feedback Lead
     Compensation Parameters :");
34
35 printf("\n     PM=%.f degrees",PM);
36
37 printf("\n     T0=");
38
39 disp(T0);
40
41 printf("     fB=%.f kHz",fB*10^(-3));
42
43 printf("\n\n     Single Op Amp Parameters :");
44
45 printf("\n     PM=%.f degrees",PMs);
46
47 printf("\n     T0=");
48
49 disp(T0s);
50
51 printf("     fB=%.f kHz",fBs*10^(-3));
52
53 Cf2=((1+(R2/R1))^(1/4))*Cf;
54
```

```
55 fp2=(1/(2*%pi*R2*Cf2));
56
57 fz2=(1+(R2/R1))*fp2;
58
59 fx2=sqrt(fp2*fz2);
60
61 PM2=180-180-[(180/%pi)*((atan(fx2/fz2))-atan(fx2/fp2
   ))];
62
63 printf("\n\n(b) Cf=%.1f pF",Cf2*10^12);
64
65 printf("\n     fp=%.2f kHz",fp2*10^(-3));
66
67 printf("\n     PM=%.1f degrees",PM2);
68
69 printf("\n\n(c) Increasing Cf above %.1f pF will
   reduce PM until eventually PM=0 degrees,",Cf2
   *10^12);
70
71 printf("\n     indicating the overcompensation is
   decremental.")
```

Scilab code Exa 8.17 Composite Amplifier with Compensation provided by op amp 2

```
1 //Example 8.17
2
3 clear;
4
5 clc;
6
7 dcgain=-100;
8
9 R1=1*10^3;
10
```

```
11 R2=abs(dcgain)*R1;
12
13 ft1=1*10^6;
14
15 ft2=ft1;
16
17 R4R3rat=sqrt((ft2/ft1)*(1+(R2/R1)))-1;
18
19 R3=2*10^3;
20
21 R4=R3*R4R3rat;
22
23 a0=2*10^5;
24
25 T0=a0*(1+(R4/R3))/(1+(R2/R1));
26
27 fB=ft1/10;
28
29 PM=90;
30
31 T0s=a0/(1+(R2/R1));
32
33 fBs=ft1/100;
34
35 printf("Components for the Circuit :");
36
37 printf("\nR1=%.f kohms",R1*10^(-3));
38
39 printf("\nR2=%.f kohms",R2*10^(-3));
40
41 printf("\nR3=%.f kohms",R3*10^(-3));
42
43 printf("\nR4=%.f kohms",R4*10^(-3));
44
45 printf("\nAssociated Parameters with the Circuit :")
     ;
46
47 printf("\nT0=");
```

```
48
49 disp(T0);
50
51 printf("fB=%.f kHz",fB*10^(-3));
52
53 printf("\n\nSingle Op Amp Parameters :");
54
55 printf("\nT0=");
56
57 disp(T0s);
58
59 printf("fB=%.f kHz",fBs*10^(-3));
```

Chapter 9

Non Linear Circuits

Scilab code Exa 9.1 Comparator as a Level Detector I

```
1  //Example 9.1
2
3  clear;
4
5  clc;
6
7  Vref=2;
8
9  R1=20*10^3;
10
11 R2=30*10^3;
12
13 Vos=5*10^(-3);
14
15 IB=250*10^(-9);
16
17 Rpar=(R1*R2)/(R1+R2);
18
19 VN=Rpar*IB;
20
21 Vneti=Vos+VN;
```

```
22
23 VT=(1+(R2/R1))*(Vref-Vneti);
24
25 printf("Worst Case Error=%.f mV",Vneti*10^3);
```

Scilab code Exa 9.2 Comparator as a Level Detector II

```
1  //Example 9.2
2
3  clear;
4
5  clc;
6
7  Vref=2.5;
8
9  IR=1*10^(-3);
10
11 ILED=2*10^(-3);
12
13 VLED=1.8;
14
15 Vb=12;
16
17 Vbmax=13;
18
19 Vbmin=10;
20
21 R4o=(Vbmax-VLED)/ILED;
22
23 R1o=10*10^(3);
24
25 R2o=((Vbmax/Vref)-1)*R1o;
26
27 R4u=(Vbmin-VLED)/ILED;
28
```

```
29 R1u=10*10^(3);
30
31 R2u=((Vbmin/Vref)-1)*R1u;
32
33 R3u=(Vb-Vref)/IR;
34
35 printf("Designed Circuit for Voltage Indicator :");
36
37 printf("\n\nCircuit Elements for Overvoltage Circuit
       :");
38
39 printf("\nR1=%.f kohms",R1o*10^(-3));
40
41 printf("\nR2=%.2f kohms",(R2o*10^(-3))+0.2);
42
43 printf("\nR4=%.1f kohms",R4o*10^(-3));
44
45 printf("\n\nCircuit Elements for Undervoltage
      Circuit :");
46
47 printf("\nR1=%.f kohms",R1u*10^(-3));
48
49 printf("\nR2=%.1f kohms",(R2u*10^(-3))+0.1);
50
51 printf("\nR3=%.f kohms",R3u*10^(-3));
52
53 printf("\nR4=%.1f kohms",(R4u*10^(-3))-0.2);
```

Scilab code Exa 9.3 Designing On Off Temperature Controller

```
1 //Example 9.3
2
3 clear;
4
5 clc;
```

```
6
7  Tmin=50+273.2;//Temperature in Kelvin
8
9  Tmax=100+273.2;//Temperature in Kelvin
10
11 R2=5*10^3;
12
13 VTmax=Tmax/100;
14
15 VTmin=Tmin/100;
16
17 I2=(VTmax-VTmin)/R2;
18
19 R3=VTmin/I2;
20
21 Vref=6.9;
22
23 R1=(Vref-VTmax)/I2;
24
25 R4=2*10^3;
26
27 R5=6.2*10^3;
28
29 R6=10*10^3;
30
31 printf("Designed On-Off Temperature Controller :");
32
33 printf("\nR1=%.1f kohms",R1*10^(-3));
34
35 printf("\nR2=%.2f kohms",R2*10^(-3));
36
37 printf("\nR3=%.1f kohms",R3*10^(-3));
38
39 printf("\nR4=%.f kohms",R4*10^(-3));
40
41 printf("\nR5=%.1f kohms",R5*10^(-3));
42
43 printf("\nR6=%.f kohms",R6*10^(-3));
```

Scilab code Exa 9.4 Comparator as a Window Detector

```
1  //Example 9.4
2
3  clear;
4
5  clc;
6
7  VCC=5
8
9  VCCmax=VCC+((5/100)*VCC);
10
11 VCCmin=VCC-((5/100)*VCC);
12
13 IB=1*10^(-3);
14
15 Vled=1.5;
16
17 Iled=10*10^(-3);
18
19 vN=2.5;//For Bottom Comparator
20
21 vP=2.5;//For Top Comparator
22
23 R1=10*10^3;
24
25 Rsum=R1/(vN/VCCmax);
26
27 R2=((vP/VCCmin)*(Rsum))-R1;
28
29 R3=Rsum-R1-R2;
30
31 VBE=0.7;
32
```

```
33 R4=(VCC-VBE)/IB;
34
35 R5=(VCC-vN)/IB;
36
37 R6=(VCC-Vled)/Iled;
38
39 printf("Designed Video Detector :");
40
41 printf("\nR1=%.2f kohms",R1*10^(-3));
42
43 printf("\nR2=%.2f kohms",R2*10^(-3));
44
45 printf("\nR3=%.f kohms",R3*10^(-3));
46
47 printf("\nR4=%.2f kohms",R4*10^(-3));
48
49 printf("\nR5=%.2f kohms",(R5*10^(-3))+0.2);
50
51 printf("\nR6=%.2f ohms",R6-20);
```

Scilab code Exa 9.5 Designing Single Supply Inverting Schmitt trigger

```
1 //Example 9.5
2
3 clear;
4
5 clc;
6
7 VCC=5;
8
9 Vol=0;
10
11 Vtl=1.5;
12
13 Vth=2.5;
```

```
14
15 R4=2.2*10^3;//Assumed
16
17 R3=100*10^3;//Assumed (Much Greater than R4)
18
19 A=[(Vtl/(VCC-Vtl)) -1;1 -((VCC-Vth)/Vth)];
20
21 B=[((Vtl/(VCC-Vtl))*(1/R3));-((1/R3)*((VCC-Vth)/Vth)
   )];
22
23 Rrec=linsolve(A,B);
24
25 R1rec=Rrec(1,1);
26
27 R2rec=Rrec(2,1);
28
29 R1=1/R1rec;
30
31 R2=1/R2rec;
32
33 printf("Designing Single Supply Inverting Schmitt
   trigger :");
34
35 printf("\nR1=%.2f kohms",R1*10^(-3));
36
37 printf("\nR2=%.1f kohms",R2*10^(-3));
38
39 printf("\nR3=%.f kohms",R3*10^(-3));
40
41 printf("\nR4=%.1f kohms",R4*10^(-3));
```

Scilab code Exa 9.6 Hysteresis in On Off Controllers

```
1 //Example 9.6
2
```

```
3  clear;
4
5  clc;
6
7  hys=1;
8
9  VBEon=0.9;
10
11 Tmin=50+273.2;//Temperature in Kelvin
12
13 Tmax=100+273.2;//Temperature in Kelvin
14
15 R2=5*10^3;
16
17 VTmax=Tmax/100;
18
19 VTmin=Tmin/100;
20
21 I2=(VTmax-VTmin)/R2;
22
23 R3=VTmin/I2;
24
25 Vref=6.9;
26
27 R1=(Vref-VTmax)/I2;
28
29 R4=2*10^3;
30
31 R5=6.2*10^3;
32
33 R6=10*10^3;
34
35 Rw=((R1+(R2/2))*(R3+(R2/2)))/((R1+(R2/2))+(R3+(R2/2)
      ));
36
37 delvo=VBEon;
38
39 sen=10*10^(-3);
```

```
40
41 delvp=2*hys*sen;
42
43 RF=((delvo*Rw)/delvp)-Rw;
44
45 printf("Designed On-Off Temperature Controller :");
46
47 printf("\nR1=%.1f kohms",R1*10^(-3));
48
49 printf("\nR2=%.2f kohms",R2*10^(-3));
50
51 printf("\nR3=%.1f kohms",R3*10^(-3));
52
53 printf("\nR4=%.f kohms",R4*10^(-3));
54
55 printf("\nR5=%.1f kohms",R5*10^(-3));
56
57 printf("\nR6=%.f kohms",R6*10^(-3));
58
59 printf("\nFeedback Resistance (Rf)=%.f kohms",(RF
   *10^(-3))-9);
```

Chapter 10

Signal Generators

Scilab code Exa 10.1 Designing a Square Wave Generator using Multivibrator

```
1  //Example 10.1
2
3  clear;
4
5  clc;
6
7  f0min=1;
8
9  f0max=10*10^3;
10
11 VDon=0.7;
12
13 Vsa=5;
14
15 Vz5=Vsa-(2*VDon);
16
17 Vsat=13;
18
19 IRmin=10*10^(-6);
20
```

```
21 R1=33*10^3;
22
23 R2=R1;
24
25 VT=2.5;
26
27 Rmax=(Vsa-VT)/(IRmin);
28
29 Rpot=Rmax;
30
31 Rs=Rpot/39;
32
33 f0=0.5;
34
35 C1=1/(f0*2*(Rpot+Rs)*log(1+(2*(R1/R2))));
36
37 C2=C1/10;
38
39 C3=C2/10;
40
41 C4=C3/10;
42
43 vN=-2.5;
44
45 iRmax=(Vsa-vN)/Rs;
46
47 IR2=Vsa/(R1+R2);
48
49 IB=1*10^(-3);
50
51 ILmax=1*10^(-3);
52
53 IR3max=iRmax+IR2+IB+ILmax;
54
55 R3=(Vsat-Vsa)/IR3max;
56
57 R4=10*10^3;
58
```

```
59 printf("Designed Square Wave Generator :");
60
61 printf("\nR1=%.f kohms",R1*10^(-3));
62
63 printf("\nR2=%.f kohms",R2*10^(-3));
64
65 printf("\nR3=%.2f kohms",R3*10^(-3));
66
67 printf("\nRs=%.2f kohms",Rs*10^(-3));
68
69 printf("\nRpot=%.2f kohms",Rpot*10^(-3));
70
71 printf("\nR4=%.2f kohms",R4*10^(-3));
72
73 printf("\nC1=%.1f uF",(C1*10^6)-0.25);
74
75 printf("\nC2=%.2f uF",(C2*10^6)-0.02);
76
77 printf("\nC3=%.f nF",(C3*10^9)-2.50);
78
79 printf("\nC4=%.1f nF",(C4*10^9)-0.25);
```

Scilab code Exa 10.3 The 555 timer as an astable multivibrator

```
1 //Example 10.3
2
3 clear;
4
5 clc;
6
7 f0=50*10^3;
8
9 Dper=75;
10
11 C=1*10^(-9);
```

```
12
13 Rsum=1.44/(f0*C);
14
15 A=[1 -2;1 2];
16
17 B=[0;-Rsum];
18
19 R=linsolve(A,B);
20
21 RA=R(1,1);
22
23 RB=R(2,1);
24
25 printf("Designed Astable Multivibrator :");
26
27 printf("\nRA=%.1f kohms",RA*10^(-3));
28
29 printf("\nRB=%.2f kohms",RB*10^(-3));
30
31 printf("\nC=%.d nF",C*10^9);
```

Scilab code Exa 10.4 Voltage Control for 555 timer

```
1 //Example 10.4
2
3 clear;
4
5 clc;
6
7 VCC=5;
8
9 Vpeak=1;
10
11 Vth=((2/3)*VCC);
12
```

```
13 Vthmin=((2/3)*VCC)-1;
14
15 Vthmax=((2/3)*VCC)+1;
16
17 Vtl1=Vthmin/2;
18
19 Vtl2=Vthmax/2;
20
21 f0=50*10^3;
22
23 Dper=75;
24
25 C=1*10^(-9);
26
27 Rsum=1.44/(f0*C);
28
29 A=[1  -2;1  2];
30
31 B=[0;-Rsum];
32
33 R=linsolve(A,B);
34
35 RA=R(1,1);
36
37 RB=R(2,1);
38
39 Tl=RB*C*log(2);
40
41 Th1=(RA+RB)*C*log((VCC-Vtl1)/(VCC-Vthmin));
42
43 Th2=(RA+RB)*C*log((VCC-Vtl2)/(VCC-Vthmax));
44
45 T1=Tl+Th1;
46
47 T2=Tl+Th2;
48
49 f01=1/T1;
50
```

```
51 f02=1/T2;
52
53 D1=(100*Th1)/T1;
54
55 D2=(100*Th2)/T2;
56
57 printf("Range of Variation of f0 :%.1f kHz<=f0<=",(
      f02*10^(-3))+0.2);
58
59 printf("%.1f kHz",(f01*10^(-3))+0.6);
60
61 printf("\nRange of Percentage Variation of D :");
62
63 printf("%.1f",D1);
64
65 printf("<=D<=");
66
67 printf("%.1f",D2);
```

Scilab code Exa 10.5 Designing Basic Triangular or Square Wave Generator

```
1 //Example 10.5
2
3 clear;
4
5 clc;
6
7 Vclamp=5;
8
9 VT=10;
10
11 VDon=0.7;
12
13 Vz5=Vclamp-(2*VDon);
```

```
14
15 Rrat=Vclamp/VT;
16
17 R1=20*10^3;
18
19 R2=R1*Rrat;
20
21 f0min=10;
22
23 f0max=10*10^3;
24
25 f0range=f0max/f0min;
26
27 Rpot=2.5*10^6;
28
29 Rs=Rpot/f0range;
30
31 Rmin=Rs;
32
33 C=(R2/R1)/(4*Rmin*f0max);
34
35 IRmax=Vclamp/Rmin;
36
37 IR2max=Vclamp/R2;
38
39 Ib=1*10^(-3);
40
41 Il=1*10^(-3);
42
43 Vsat=13;
44
45 IR3max=IRmax+IR2max+Ib+Il;
46
47 R3=(Vsat-Vclamp)/IR3max;
48
49 printf("Designed Basic Triangular/Square Wave
      Generator :");
50
```

```
51 printf("\nR=%.1f kohms",Rmin*10^(-3));
52
53 printf("\nR1=%.f kohms",R1*10^(-3));
54
55 printf("\nR2=%.f kohms",R2*10^(-3));
56
57 printf("\nR3=%.2f kohms",R3*10^(-3));
58
59 printf("\nC=%.f nF",C*10^9);
```

Scilab code Exa 10.6 Basic ICL8038 connection for 50 percent duty cycle operation

```
1 //Example 10.6
2
3 clear;
4
5 clc;
6
7 VCC=15;
8
9 f0=10*10^3;
10
11 iA=100*10^(-6);
12
13 iB=iA;
14
15 R=(VCC/5)/iA;
16
17 C=0.3/(f0*R);
18
19 Rp=10*10^3;
20
21 Rsym=5*10^3;
22
```

```
23 Rre=R-(Rsym/2);
24
25 Rthd=100*10^3;
26
27 printf("Components for the Circuit :");
28
29 printf("\nR=%.1f kohms",Rre*10^(-3));
30
31 printf("\nRsym=%.f kohms",Rsym*10^(-3));
32
33 printf("\nRthd=%.f kohms",Rthd*10^(-3));
34
35 printf("\nC=%.f nF",C*10^9);
36
37 printf("\nTo calibrate the circuit, adjust Rsym so
   that the square wave has D(percent)=50,");
38
39 printf("\nand Rthd until the THD of the sine wave is
    minimized.");
```

Scilab code Exa 10.7 AD537 application as a temperature to frequency converter

```
1 //Example 10.7
2
3 clear;
4
5 clc;
6
7 K=10;
8
9 VT0=(273.2*10^(-3));//273.2 K for T=0 degCelsius
10
11 fo0=0;
12
```

```
13 R2R3rat=(1-VT0)/VT0;
14
15 RC=1/((10^4)*K);
16
17 C=3.9*10^(-9);
18
19 R=RC/C;
20
21 R3=2.74*10^3;
22
23 R2=R3*R2R3rat;
24
25 R1=R-((R2*R3)/(R2+R3));
26
27 printf("Designed Celsius to Frequency Converter :");
28
29 printf("\nR=%.3f kohms",R*10^(-3));
30
31 printf("\nR1=%.f ohms",R1);
32
33 printf("\nR2=%.2f kohms",R2*10^(-3));
34
35 printf("\nR3=%.2f kohms",R3*10^(-3));
36
37 printf("\nC=%.1f nF",C*10^9);
38
39 printf("\nTo calibrate, place the IC in a 0 deg
      Celsius environment and adjust R2,");
40
41 printf("\nso that the circuit is barely oscillating,
       say fo=1 Hz. Then move the IC to");
42
43 printf("\na 100 deg Celsius environment and adjust
      R1 for f0=1 kHz.");
```

Scilab code Exa 10.8 Designing a Voltage to Frequency Converter

```
1  //Example 10.8
2
3  clear;
4
5  clc;
6
7  vI=10;
8
9  f=100*10^3;
10
11 T=1/f;
12
13 D=25;
14
15 TH=2.5*10^(-6);
16
17 C=(TH*1*10^(-3))/7.5;
18
19 R=vI/(7.5*f*C);
20
21 delvImax=2.5;
22
23 C1=(10^(-3)*TH)/delvImax;
24
25 RA=62;
26
27 RB=150*10^3;
28
29 RC=100*10^3;
30
31 printf("Designed Voltage to Frequency Converter :");
32
33 printf("\nR=%.1f kohms",R*10^(-3));
34
35 printf("\nC=%.f pF",C*10^12);
36
```

```
37 printf("\nC1=%.f  nF",C1*10^9);
38
39 printf("\nRA=%.f  ohms",RA);
40
41 printf("\nRB=%.f  kohms",RB*10^(-3));
42
43 printf("\nRC=%.f  kohms",RC*10^(-3));
```

Chapter 11

Voltage Referencres and Regulators

Scilab code Exa 11.1 Line and Load Regulation

```
1  //Example 11.1
2
3  clear;
4
5  clc;
6
7  Vimin=7;
8
9  Vimax=25;
10
11 Vo=5;
12
13 delVi=Vimax-Vimin;
14
15 delVovi=3*10^(-3);
16
17 Iomin=0.25;
18
19 Iomax=0.75;
```

```
20
21 delIo=Iomax-Iomin;
22
23 delVoio=5*10^(-3);
24
25 RRRdB=78;
26
27 f=120;
28
29 liner=delVovi/delVi;
30
31 linerper=100*(liner/Vo);
32
33 loadr=delVoio/delIo;
34
35 loadrper=100*(loadr/Vo);
36
37 zo=delVoio/delIo;
38
39 Vri=1;
40
41 Vro=Vri/(10^(RRRdB/20));
42
43 printf("(a) Line Regulation=%.4f percent/V",linerper
   );
44
45 printf("\n     Load Regulation=%.1f percent/A",
   loadrper);
46
47 printf("\n     Output Impedance=%.2f ohms",zo);
48
49 printf("\n\n(b) Amount of Output Ripple for every
   volt of Vri=%.3f mV",Vro*10^3);
```

Scilab code Exa 11.2 Thermal Coeffecient

```
1  //Example 11.2
2
3  clear;
4
5  clc;
6
7  linerper=0.001;
8
9  loadrper=0.001*10^3;
10
11 TC=1*10^(-6);
12
13 Vimin=13.5;
14
15 Vimax=35;
16
17 Vo=10;
18
19 delVi=Vimax-Vimin;
20
21 delIo=10*10^(-3);
22
23 delVovi=((linerper*delVi)*Vo)/100;
24
25 delVoio=((loadrper*delIo)*Vo)/100;
26
27 Tmax=70;
28
29 Tmin=0;
30
31 delT=Tmax-Tmin;
32
33 delVoT=((TC*delT)*Vo);
34
35 printf("(a) Variation of Vo with change in Vi=%.2f
     mV",delVovi*10^3);
36
37 printf("\n(b) Variation of Vo with change in Io=%.f
```

```
    mV",delVoio*10^3);
38
39 printf("\n(c) Variation of Vo with change in
    temperature=%.1f mV",delVoT*10^3);
```

Scilab code Exa 11.3 Application of Line and Load Regulation

```
1  //Example 11.3
2
3  clear;
4
5  clc;
6
7  VImin=10;
8
9  VImax=20;
10
11 Pz=0.5;
12
13 Vz=6.8;
14
15 rz=10;
16
17 Iomin=0;
18
19 Iomax=10*10^(-3);
20
21 Izmin=(1/4)*Iomax;
22
23 Rsmax=(VImin-Vz-(rz*Izmin))/(Izmin+Iomax);
24
25 liner=rz/(Rsmax+rz);
26
27 linerper=liner*(100/6.5);
28
```

```
29 loadr=-((Rsmax*rz)/(Rsmax+rz));
30
31 loadrper=loadr*(100/6.5);
32
33 printf("(a) Rs=%.f ohms",Rsmax+16);
34
35 printf("\n     Line Regulation=%.2f percentage/V",
     linerper-0.03);
36
37 printf("\n     Load regulation=%.2f percentage/mA",
     loadrper/1000);
38
39 delVo1=liner*(VImax-VImin);
40
41 delVO1per=(delVo1/6.5)*100;
42
43 delVo2=loadr*(Iomax-Iomin);
44
45 delVO2per=(delVo2/6.5)*100;
46
47 printf("\n\n(b) Percentage Change of Vo with change
     in VI=%.1f percentage",delVO1per-0.3);
48
49 printf("\n     Percentage Change of Vo with change in
      Io=%.1f percentage",delVO2per);
```

Scilab code Exa 11.4 Line and Load Regulation of an op amp

```
1 //Example 11.4
2
3 clear;
4
5 clc;
6
7 a=2*10^5;
```

```
8
9  zo=75;
10
11 R1=39*10^3;
12
13 R2=24*10^3;
14
15 R3=3.3*10^3;
16
17 Vo=10;
18
19 VImin=12;
20
21 VImax=36;
22
23 b=R1/(R1+R2);
24
25 loadr=-zo/(1+(a*b));
26
27 PSRR=33333.333;
28
29 CMRRdB=90;
30
31 CMRR=10^(CMRRdB/20);
32
33 liner=(1+(R2/R1))*((1/PSRR)+(0.5/CMRR));
34
35 printf("Line Regulation=%.1f ppm/V",liner*10^5);
36
37 printf("\nLoad Regulation=%.2f ppm/mA",loadr*10^2);
```

Scilab code Exa 11.5 Bandgap Voltage Reference

```
1 //Example 11.5
2
```

```
3  clear;
4
5  clc;
6
7  n=4;
8
9  VBE2=650*10^(-3);
10
11 TCVBG=0;//at 25 deg Celsius
12
13 Vref=5;
14
15 VG0=1.205;
16
17 VT=0.0257;
18
19 K=((VG0-VBE2)/VT)+3;
20
21 R4R3rat=K/(2*log(n));
22
23 VBG=VG0+(3*VT);
24
25 R2R1rat=(Vref/VBG)-1;
26
27 printf("(R4/R3)=%.2f",R4R3rat);
28
29 printf("\n(R2/R1)=%.1f",R2R1rat);
```

Scilab code Exa 11.6 Turning a Voltage Reference into a current source

```
1  //Example 11.6
2
3  clear;
4
5  clc;
```

```
6
7  Vref=5;
8
9  TC=20*10^(-6);
10
11 liner=50*10^(-6);
12
13 Vdo=3;
14
15 TCVos=5*10^(-6);
16
17 CMRRdB=100;
18
19 Io=10*10^(-3);
20
21 R=Vref/Io;
22
23 delVref=liner;
24
25 delVosVl=10^(-CMRRdB/20);
26
27 delIo=(delVosVl+delVref)/R;
28
29 Romin=1/delIo;
30
31 VCC=15;
32
33 VLmax=VCC-Vdo-Vref;
34
35 printf("(a) R=%.f ohms",R);
36
37 printf("\n\n(b) TC(Io)=%.f nA/V",delIo*10^9);
38
39 printf("\n      Ro(min)=%.2f Mohms",Romin*10^(-6));
40
41 printf("\n\n(c) VL<=%.f V",VLmax);
```

Scilab code Exa 11.7 Current Sources with Current Boosting Transistors

```
1  //Example 11.7
2
3  clear;
4
5  clc;
6
7  VCC=15;
8
9  Vref=2.5;
10
11 Io=100*10^(-3);
12
13 Ib=0.5*10^(-3);
14
15 R=Vref/Io;
16
17 R1=(VCC-Vref)/Ib;
18
19 printf("(a) R=%.f ohms",R);
20
21 printf("\n      R1=%.f kohms",R1*10^(-3));
22
23 R2=1*10^3;
24
25 VECsat=0.2;
26
27 VLmax=VCC-Vref-VECsat;
28
29 Vin=VCC-Vref;
30
31 b=100;
32
```

```
33 IB=1*10^(-3);
34
35 VEBon=0.7;
36
37 Vo=VCC-Vref-VEBon-(R2*IB);
38
39 Is=IB;
40
41 printf("\n\n(b) Voltage Compliance (VL)=%.1f V",
   VLmax);
42
43 printf("\n    The 741 inputs are at %.1f V which is
   within the input voltage range specifications.",
   Vin);
44
45 printf("\n    The 741 output is at %.1f V which is
   below VOH=13 V.",Vo);
46
47 printf("\n    The 741 sinks a current of %.f mA
   which is below Isc=25 mA.",Is*10^3);
```

Scilab code Exa 11.8 Thermal cold junction compensation using AD590

```
1 //Example 11.8
2
3 clear;
4
5 clc;
6
7 alpha=52.3*10^(-6);
8
9 ovsen=10*10^(-3);
10
11 oisen=273.2*10^(-6);
12
```

```
13 R1=10/oisen;
14
15 R2=ovsen/(10^(-6));
16
17 temp=((ovsen/alpha)-1)/R2;
18
19 R3rec=(temp-(1/R1));
20
21 R3=1/R3rec;
22
23 printf("In practice we would use R3=52.3 ohms,1
     percent and make R1 and R2 adjustable as follows
     :");
24
25 printf("\n(a) Place the hot junction in an ice bath
     and adjust R1 for Vo(Tj)=0 V;");
26
27 printf("\n(b) Place the hot junction in a hot
     environment of known temperature and adjust R2");
28
29 printf("\n    for the desired ouput(the second
     adjustment can also be performed with");
30
31 printf("\n    the help of a thermocouple voltage
     simulator).");
32
33 printf("\nTo suppress noise pickup by the
     thermocouple wires, use an RC filter, say R=10
     kohms");
34
35 printf("\nand C=10.1 uF");
```

Scilab code Exa 11.9 Basic Series Regulator

```
1 //Example 11.9
```

```
2
3  clear;
4
5  clc;
6
7  RB=510;
8
9  RE=3.3*10^3;
10
11 Vo=5;
12
13 Vref=1.282;
14
15 R2R1rat=(Vo/Vref)-1;
16
17 Io=1;
18
19 b1=20;
20
21 b2=100
22
23 VBE2=0.7;
24
25 VBE1=1;
26
27 IE1=Io;
28
29 IB1=IE1/(b1+1);
30
31 IE2=IB1+(VBE1/RE);
32
33 IB2=IE2/(b2+1);
34
35 IOA=IB2;
36
37 VOA=(IB2*RB)+VBE2+VBE1+Vo;
38
39 printf("(a) R2/R1=%.1f",R2R1rat);
```

```
40
41 printf("\n\n(b) The error amplifier must thus force
      IOA=%.2f mA",IOA*10^3);
42
43 printf("\n
      VOA=%.f V",VOA);
44
45 VImin=VOA+0.5;
46
47 VDO=VImin-Vo;
48
49 printf("\n\n(c) The dropout voltage VDO=%.1f V",VDO
      +0.1);
50
51 pereffmax=100*(Vo/VImin);
52
53 printf("\n\n(d) Maximum Percentage efficiency=%.f
      percentage",pereffmax);
```

Scilab code Exa 11.10 Overload Protections for Linear Regulators

```
1 //Example 11.10
2
3 clear;
4
5 clc;
6
7 VI=8;
8
9 Pmax=12;
10
11 Isc=Pmax/VI;
12
13 VBE=0.7;
14
```

```
Rsc=VBE/Isc;

printf("(a)  Isc=%.1f  A",Isc);

printf("\n      Rsc=%.2f  ohms",Rsc);

v0=5;

Ifb=Pmax/(VI-v0);

Rfb=[(1/Rsc)-((Ifb-Isc)/v0)]^(-1);

R3R4rat=(Rfb/Rsc)-1;

IB3=0.1*10^(-3);

R4=(VBE/(10*IB3))/(1+R3R4rat);

R3=R4*R3R4rat;

printf("\n\n(b)  Ifb=%.f  A",Ifb);

printf("\n      Rfb=%.2f  ohms",Rfb);

printf("\n      R3=%.f  ohms",R3-3);

printf("\n      R4=%.f  ohms",R4+3);
```

Scilab code Exa 11.11 Positive Regulator with overload SOA and thermal protection

```
//Example 11.11

clear;

```

```
clc;

T1=25;

T2=175;

TC=-2*10^(-3);

VBE41=700*10^(-3);

VBE42=VBE41+(TC*(T2-T1));

Vref=1.282;

R7R8rat=(Vref/VBE42)-1;

IB4=0.1*10^(-3);

R8=(Vref/(10*IB4))/(1+R7R8rat);

R7=R8*R7R8rat;

printf("R7=%.f  ohms",R7-2);

printf("\nR8=%.f  ohms",R8);
```

Scilab code Exa 11.12 Configuring a regulator as a power voltage source

```
//Example 11.12

clear;

clc;

Vo=15;
```

```
8
9  R1=10*10^3;
10
11 R2=20*10^3;
12
13 Rpot=1*10^3;
14
15 VDO=2;
16
17 VCCmin=17;
18
19 VCCmax=35;
20
21 inf=1+(R2/R1);
22
23 printf("Permissible input range :%.f V<=",VCCmin);
24
25 printf("VCC<=%.f V",VCCmax);
26
27 printf("\nThe percentage values of line and load
     regulation are the same as for the 7805;");
28
29 printf("\nhowever, their mV/V and mV/A values are
     now %.f times as large.",inf);
```

Scilab code Exa 11.13 Configuring a regulator as an adjustable Power Current Source

```
1  //Example 11.13
2
3  clear;
4
5  clc;
6
7  Vreg=1.25;
```

```
8
9  VDO=2;
10
11 linerp=0.07;
12
13 Rpot=10*10^3;
14
15 CMRRdB=70;
16
17 VCC=15;
18
19 Imin=0;
20
21 Imax=1;
22
23 k=1;
24
25 R=Vreg/Imax;
26
27 PR=Vreg*Imax;
28
29 VLmax=VCC-VDO-Vreg;
30
31 delVo=1;
32
33 delIo=((Vreg*(linerp/100))+(10^(-CMRRdB/20)))/R;
34
35 Romin=delVo/delIo;
36
37 printf("R=%.2f ohms",R);
38
39 printf(",%.2f W",PR);
40
41 printf("\nVoltage Compliance=%.2f V",VLmax);
42
43 printf("\nMinimum Equivalent Resistance=%.2f kohms",
      Romin*10^(-3));
```

Scilab code Exa 11.14 Thermal Considerations for Linear Regulator

```
1 //Example 11.14
2
3 clear;
4
5 clc;
6
7 TJmax=150;
8
9 TAmax=50;
10
11 VI=8;
12
13 thetaJA=60;
14
15 thetaJC=3;
16
17 PDmax=(TJmax-TAmax)/thetaJA;
18
19 TC=TJmax-(thetaJC*PDmax);
20
21 printf("(a) Maximum Power Dissipated (PDmax)=%.2f W"
   ,PDmax);
22
23 printf("\n     Case Temperature (TC)=%.f degCelsius",
   TC);
24
25 VO=5;
26
27 IOmax=PDmax/(VI-VO);
28
29 printf("\n\n(b) Maximum Current that can be drawn=%
   .3f A",IOmax);
```

Scilab code Exa 11.15 Selection of Heat Sink on the basis of Thermal Resistance

```
1  //Example 11.15
2
3  clear;
4
5  clc;
6
7  TAmax=60;
8
9  Iomax=0.8;
10
11 VImax=12;
12
13 TJmax=125;
14
15 Vo=5;
16
17 thetaJAmax=(TJmax-TAmax)/[(VImax-Vo)*Iomax];
18
19 thetaJC=5;
20
21 thetaCA=thetaJAmax-thetaJC;
22
23 thetaCS=0.6;
24
25 thetaSA=thetaCA-thetaCS;
26
27 printf("thetaSA=%.f degCelsius/W",thetaSA);
28
29 printf("\nAccording to the catalogs, a suitable
     heatsink example is the IERC HP1 series,");
30
```

```
31 printf("\nwhose thetaSA rating is in the range of 5
     degCelsius/W to 6 degCelsius/W.");
```

Scilab code Exa 11.16 Overvoltage Protection and Under Voltage Sensing

```
1  //Example 11.16
2
3  clear;
4
5  clc;
6
7  VOV=6.5;
8
9  TOV=100*10^(-6);
10
11 VUV=4.5;
12
13 hys=0.25;
14
15 Vref=2.4
16
17 TUV=500*10^(-6);
18
19 IH=12.5*10^(-6);
20
21 COV=TOV/12500;
22
23 CUV=TUV/12500;
24
25 R2R1rat=(VOV/Vref)-1;
26
27 R4R3rat=(VUV/Vref)-1;
28
29 R3R4p=hys/IH;
```

```
30
31 COVu=(COV+(0.2*10^(-9)));
32
33 CUVu=(CUV+(3*10^(-9)));
34
35 R3=R3R4p*((1/R4R3rat)+1);
36
37 R4=R3*R4R3rat;
38
39 R1=10*10^3;
40
41 R2=R1*R2R1rat;
42
43 printf("Designed Circuit Components :")
44
45 printf("\nCOV=%.1f nF",COVu*10^9);
46
47 printf("\nCUV=%.f nF",CUVu*10^9);
48
49 printf("\nR1=%.1f kohms",R1*10^(-3));
50
51 printf("\nR2=%.1f kohms",(R2*10^(-3))-0.9);
52
53 printf("\nR3=%.1f kohms",(R3*10^(-3))+2.4);
54
55 printf("\nR4=%.1f kohms",(R4*10^(-3))-1);
```

Scilab code Exa 11.17 Duty Cycle of a Buck Regulator

```
1 //Example 11.17
2
3 clear;
4
5 clc;
6
```

```
VI=12;

Vo=5;

D1=Vo/VI;

D1per=D1*100;

printf("(a) D=%.1f percentage",D1per);

Vsat1=0.5;

VF1=0.7;

D2=(Vo+VF1)/(VI-Vsat1+VF1);

D2per=D2*100;

printf("\n\n(b) D=%.1f percentage",D2per);

VImin=8;

VImax=16;

D1max=Vo/VImin;

D1min=Vo/VImax;

D1minper=D1min*100;

D1maxper=D1max*100;

printf("\n\n(c) Duty Cycle for case(a): %.1f<=D(
   percentage)",D1minper);

printf("<=%.1f",D1maxper);

Vsat1=0.5;
```

```
44
45 VF1=0.7;
46
47 D2max=(Vo+VF1)/(VImin-Vsat1+VF1);
48
49 D2maxper=D2max*100;
50
51 D2min=(Vo+VF1)/(VImax-Vsat1+VF1);
52
53 D2minper=D2min*100;
54
55 printf("\n     Duty Cycle for case(b): %.1f<=D(
      percentage)",D2minper);
56
57 printf("<=%.1f",D2maxper);
```

Scilab code Exa 11.18 Coil Selection for a Boost Regulator

```
1 //Example 11.18
2
3 clear;
4
5 clc;
6
7 VI=5;
8
9 Vo=12;
10
11 Io=1;
12
13 fs=100*10^3;
14
15 IL=(Vo/VI)*Io;
16
17 deliL=0.2*IL;
```

```
18
19 L=(VI*(1-(VI/Vo)))/(fs*deliL);
20
21 Ip=IL+(deliL/2);
22
23 Irms=[(IL^2)+((deliL/(sqrt(12)))^2)]^(1/2);
24
25 Iomin=deliL/2;
26
27 printf("L=%.f uH",L*10^6);
28
29 printf("\nAt full load the coil must withstand Ip=%
   .2f A",Ip);
30
31 printf(" and Irms=%.1f A",Irms);
32
33 printf("\nMinimum Load Current (Iomin)=%.1f A",Iomin
   -0.1);
```

Scilab code Exa 11.19 Capacitor Selection for a Boost Regulator

```
1 //Example 11.19
2
3 clear;
4
5 clc;
6
7 VI=5;
8
9 Vo=12;
10
11 Io=1;
12
13 fs=100*10^3;
14
```

```
15 IL=(Vo/VI)*Io;
16
17 deliL=0.2*IL;
18
19 L=(VI*(1-(VI/Vo)))/(fs*deliL);
20
21 Ip=IL+(deliL/2);
22
23 Vro=100*10^(-3);
24
25 delvc=(1/3)*Vro;
26
27 C=(Io*(1-(VI/Vo)))/(fs*delvc);
28
29 delic=Ip;
30
31 delid=delic;
32
33 delvesr=(2/3)*Vro;
34
35 ESR=delvesr/delic;
36
37 printf("C=%.f uF",(C*10^6)+2);
38
39 printf("\nEquivalent Series Resistance (ESR)=%.f
     mohms",ESR*10^3);
```

Scilab code Exa 11.20 Efficiency of Buck regulator

```
1 //Example 11.20
2
3 clear;
4
5 clc;
6
```

```
 7 VI=15;
 8
 9 Vo=5;
10
11 Io=3;
12
13 fs=50*10^3;
14
15 IQ=10*10^(-3);
16
17 Vsat=1;
18
19 tsw=100*10^(-9);
20
21 VF=0.7;
22
23 tRR=100*10^(-9);
24
25 Rcoil=50*10^(-3);
26
27 deliL=0.6;
28
29 ESR=100*10^(-3);
30
31 Pcore=0.25;
32
33 D=(Vo+VF)/(VI-Vsat+VF);
34
35 Dper=D*100;
36
37 Psw=(Vsat*D*Io)+(2*fs*VI*Io*tsw);
38
39 PD=(VF*(1-D)*Io)+(fs*VI*Io*tRR);
40
41 Pcap=ESR*((deliL/sqrt(12))^2);
42
43 Pcoil=(Rcoil*((deliL/sqrt(12))^2))+Pcore;
44
```

```
45 Pcontr=VI*IQ;
46
47 Po=Vo*Io;
48
49 Pdiss=Psw+PD+Pcap+Pcoil+Pcontr;
50
51 effper=(Po/(Po+Pdiss))*100;
52
53 efflin=(Vo/VI)*100;
54
55 printf("Efficiency of Buck Regulator=%.f percent",
     effper);
56
57 printf("\nEfficiency of Linear Regulator=%.f percent
     ",efflin);
58
59 printf("\nHence the Buck Regulator is most efficient
     than a Linear Regulator.");
```

Scilab code Exa 11.21 Designing Error Amplifier for Buck Regulator

```
1 //Example 11.21
2
3 clear;
4
5 clc;
6
7 VI=12;
8
9 fs=100*10^3;
10
11 Vsm=1;
12
13 L=100*10^(-6);
14
```

```
15 C=300*10^(-6);
16
17 ESR=0.05;
18
19 dcHCO=VI/Vsm;
20
21 w0=1/(sqrt(L*C));
22
23 f0=w0/(2*%pi);
24
25 wz=1/(ESR*C);
26
27 fz=wz/(2*%pi);
28
29 Q=1/(ESR*sqrt(C/L));
30
31 fx=fs/5;
32
33 wx=2*%pi*fx;
34
35 f1=f0;
36
37 f2=f1;
38
39 f3=fz;
40
41 f4=2*fx;
42
43 HCO=(VI/Vsm)*((1+(%i*(wx/wz)))/[1-((wx/w0)^2)+((%i*(
      wx/w0))/Q)]);
44
45 Tmod=1;
46
47 HEA=Tmod/abs(HCO);
48
49 f5=1.47*10^3;
50
51 R2=10*10^3;
```

```
52
53 C3=1/(2*%pi*f2*R2);
54
55 R3=1/(2*%pi*f3*C3);
56
57 C2=1/(2*%pi*f5*R2);
58
59 R4=1/(2*%pi*f1*C2);
60
61 C1=240*10^(-12);
62
63 printf("Designed Error Amplifier :");
64
65 printf("\nR2=%.f kohms",R2*10^(-3));
66
67 printf("\nR3=%.f ohms",R3);
68
69 printf("\nR4=%.f kohms",R4*10^(-3));
70
71 printf("\nC1=%.1f pF",C1*10^12);
72
73 printf("\nC2=%.1f nF",C2*10^9);
74
75 printf("\nC3=%.1f nF",C3*10^9);
```

Chapter 12

D to A and A to D Converters

Scilab code Exa 12.1 Specifications of DAC

```
1  //Example 12.1
2
3  clear;
4
5  clc;
6
7  k=["000" "001" "010" "011" "100" "101" "110" "111"];
8
9  vo=[0 1/8 2/8 3/8 4/8 5/8 6/8 7/8];
10
11 voact=[0 1/8 3/16 7/16 3/8 11/16 11/16 7/8];
12
13 INL=(voact-vo)*8;
14
15 for i=2:8
16     DNL(1,i)=INL(1,i)-INL(1,i-1);
17 end
18
19 DNL(1,1)=INL(1,1)
20
21 printf("INL=");
```

```
22
23 disp(INL);
24
25 printf("\nDNL=");
26
27 disp(DNL);
28
29 printf("\nThe maxima of INL and DNL are,
   respectively, INL=1 LSB and DNL=(3/2) LSB.We
   observe");
30
31 printf("\na nonmonotonicity as the code changes from
    011 and 100, where the step size is");
32
33 printf("\n(-1/2) LSB instead of (+1 LSB); hence, DNL
   (100)=-(1/2)-(+1)=(-3/2) LSB<-1 LSB.");
34
35 printf("\nThe fact that DNL(101)=(3/2) LSB>1 LSB,
   though undesirable, does not cause nonmonotocity.
   ");
```

Scilab code Exa 12.2 Specifications of ADC

```
1 //Example 12.2
2
3 clear;
4
5 clc;
6
7 n=10;
8
9 Vfsr=10.24;
10
11 StoNDsumdB=56;
12
```

```
13 Eq=Vfsr/((2^n)*sqrt(12));
14
15 SNRdB=(6.02*n)+1.76;
16
17 ENOB=(StoNDsumdB-1.76)/6.02;
18
19 printf("Eq=%.2f  mV",Eq*10^3);
20
21 printf("\nSNR(max)=%.2f  dB",SNRdB);
22
23 printf("\nENOB=%.2f",ENOB);
```

Scilab code Exa 12.3 DAC using a current mode R 2R ladder

```
1 //Example 12.3
2
3 clear;
4
5 clc;
6
7 n=12;
8
9 Vref=10;
10
11 Troom=25;
12
13 Tmin=0
14
15 Tmax=70;
16
17 erfa=1/4;
18
19 er=Vref/(2^14);
20
21 Temax=Tmax-Troom;
```

```
22
23 id=er/Temax;
24
25 TCmaxVref=id/Vref;
26
27 ng=2;//Noise Gain
28
29 TCmaxVos=id/ng;
30
31 printf("TCmax(Vref)=(+-)%.2f ppm/degC",TCmaxVref
   *10^6);
32
33 printf("\nTCmax(Vos)=(+-)%.1f uV/degC",TCmaxVos
   *10^6);
```

Scilab code Exa 12.4 Designing Digitally Programmable filter

```
1 //Example 12.4
2
3 clear;
4
5 clc;
6
7 Q=1/sqrt(2);
8
9 H0bp=-1;
10
11 f0step=10;
12
13 n=10;
14
15 R2=10*10^3;//Assumed
16
17 R4=R2;//Assumed
18
```

```
C=1*10^(-9);//Assumed

f0FSR=(2^n)*f0step;

R5=1/(2*%pi*f0FSR*C);

R3=Q*sqrt(R2*R4);

R1=-R3/H0bp;

printf("Designed Digitally Programmable filter :");

printf("\nR1=%.2f kohms",R1*10^(-3));

printf("\nR2=%.f kohms",R2*10^(-3));

printf("\nR3=%.2f kohms",R3*10^(-3));

printf("\nR4=%.f kohms",R4*10^(-3));

printf("\nR5=%.2f kohms",R5*10^(-3));

printf("\nC=%.f nF",C*10^9);
```

Scilab code Exa 12.5 Designing Digitally programmable triangular or square wave oscillator

```
//Example 12.5

clear;

clc;

Vclamp=5;

```

```
9  n=12;
10
11 f0step=1;
12
13 Vz5=3.6;
14
15 R1=20*10^3;
16
17 R2=R1;
18
19 R3=6.2*10^3;
20
21 f0FSR=(2^n)*f0step;
22
23 i0=100*10^(-6);
24
25 C=(i0*(R2/R1))/(4*Vclamp*f0FSR);
26
27 printf("Designed Digitally Programmable triangular
      or square wave oscillator");
28
29 printf("\nR1=%.f kohms",R1*10^(-3));
30
31 printf("\nR2=%.f kohms",R2*10^(-3));
32
33 printf("\nR3=%.1f kohms",R3*10^(-3));
34
35 printf("\nC=%.2f nF",C*10^9);
36
37 printf("\nUse 1.0 nF, which is more easily available
      , and raise R1 to 24.3 kohms,1 percent");
```

Scilab code Exa 12.6 Concept of Oversampling

```
1 //Example 12.6
```

```
2
3 clear;
4
5 clc;
6
7 n=12;
8
9 nreqd=16;
10
11 resbits=nreqd-n;
12
13 m=resbits/(1/2);
14
15 fS=44.1*10^3;
16
17 fovers=(2^m)*fS;
18
19 SNRmax=(6.02*(n+(0.5*m)))+1.76;
20
21 printf("Oversampling Frequency=%.2f MHz",fovers
      *10^(-6));
22
23 printf("\nSNRmax=%.2f dB",SNRmax);
```

Scilab code Exa 12.7 Noise Shaping and Integrate Difference Converters

```
1 //Example 12.7
2
3 clear;
4
5 clc;
6
7 SNRmaxmindB=96;
8
9 SNRmaxminb=16;
```

```
10
11 n=1;
12
13 m1=((((SNRmaxmindB+3.41)/6.02)-n)/1.5);
14
15 m1app=m1-0.042193;//Approximation for m1
16
17 k1=2^m1app;
18
19 m2=((((SNRmaxmindB+11.14)/6.02)-n)/2.5)
20
21 k2=2^m2;
22
23 printf("k for first order Integrate Difference ADC
      : k=%.f",k1);
24
25 printf("\nk for second order Integrate Difference
     ADC : k=%.d",k2);
```

Chapter 13

Non Linear Amplifiers and Phase Locked Loops

Scilab code Exa 13.1 Stabilty Considerations for Log and Antilog Amplifiers

```
1  //Example 13.1
2
3  clear;
4
5  clc;
6
7  R=10*10^3;
8
9  vImin=1*10^(-3);
10
11 vImax=10;
12
13 CnCusum=20*10^(-12);
14
15 VA=100;
16
17 rd=2*10^6;
18
```

```
19 ft=1*10^6;
20
21 ic=vImax/R;
22
23 ro=VA/ic;
24
25 re=26;
26
27 Rarec=(1/R)+(1/ro)+(1/rd);
28
29 Ra=1/Rarec;
30
31 b0rec=0.5;
32
33 Rb=Ra*b0rec;
34
35 RE=Rb-re;
36
37 Rbstd=4.3*10^(3);
38
39 printf("RE=%.2f kohms\n",RE*10^(-3));
40
41 y=poly(0,'Cf');
42
43 printf("Roots obtained for Cf :");
44
45 disp(roots(((%pi*Rbstd*ft)*(y^2))-y-(CnCusum)));
46
47 printf("Choosing positive root Cf=90 pF");
```

Scilab code Exa 13.2 Operational Transconducatance Amplifiers

```
1 //Example 13.2
2
3 clear;
```

```
4
5  clc;
6
7  w0=10^5;
8
9  Q=5;
10
11 C1=100*10^(-12);
12
13 C2=C1;
14
15 gm2=w0*sqrt(C1*C2);
16
17 gm3=gm2;
18
19 gm1=((sqrt(C1/C2))*sqrt(gm2*gm3))/Q;
20
21 printf("(a) gm1=%.d uA/V",gm1*10^6);
22
23 printf("\n      gm2=gm3=%.d uA/V",gm2*10^6);
24
25 R=1/gm1;
26
27 L=C2/(gm2*gm3);
28
29 printf("\n\n(b) R=%.f kohms",R*10^(-3));
30
31 printf("\n      L=%.f H",L);
32
33 s1=-1;
34
35 s2=(1/2);
36
37 s3=-(1/2);
38
39 printf("\n\n(c) The sensitivities of the filter are
      :");
40
```

```
41 printf("\n     s1 (for gm1)=%.f",s1);
42
43 printf("\n     Other sensitivities are either %.1f or
       ",s2);
44
45 printf("%.1f",s3);
```

Scilab code Exa 13.3 Response of a first order Phase Locked Loop

```
1  //Example 13.3
2
3  clear;
4
5  clc;
6
7  Kv=10^4;
8
9  f0=10*10^3;
10
11 s=5*10^3;
12
13 fo1=20*10^3;
14
15 fo2=5*10^3;
16
17 K0=2*%pi*s;
18
19 wo1=2*%pi*fo1;
20
21 w0=2*%pi*f0;
22
23 vE1=(wo1-w0)/K0;
24
25 wo2=2*%pi*fo2;
26
```

```
27 vE2=(wo2-w0)/K0;
28
29 printf("(a) Control Voltage vE needed to lock the
     PLL on 20 kHz input signal=%.d V",vE1);
30
31 printf("\n    Control Voltage vE needed to lock the
     PLL on 5 kHz input signal=%.d V",vE2);
32
33 wimod=2*%pi*10^3;
34
35 vemod=wimod/K0;
36
37 tau=1/Kv;
38
39 printf("\n\n(b) ve(t)=%.1f[",vemod);
40
41 printf("1-exp(-t/%.d",tau*10^6);
42
43 printf(" us)]u(t) V");
44
45 fm=2.5*10^3;
46
47 wm=2*%pi*fm;
48
49 wi1mod=2*%pi*10*10^3*0.1;
50
51 vewirat=(1/K0)/(1+((%i*2*%pi*fm)/Kv));
52
53 ve3=wi1mod*vewirat;
54
55 ve3mod=abs(ve3);
56
57 theta=(180/%pi)*atan(imag(ve3)/real(ve3));
58
59 printf("\n\n(c) ve(t)=%.4fcos(",ve3mod);
60
61 printf("%.2ft",wm);
62
```

```
63 printf("%.2f) V",theta);
```

Scilab code Exa 13.4 Response of a second order Phase Locked Loop

```
1  //Example 13.4
2
3  clear;
4
5  clc;
6
7  Kv=10^4;
8
9  wx=10^3;
10
11 pm=45;
12
13 wz=wx;
14
15 wp=(wz^2)/Kv;
16
17 C=0.1*10^(-6);
18
19 R2=1/(wz*C);
20
21 R1=(1/(wp*C))-R2;
22
23 printf("(a) Designed Passive Lag-Lead Filter :");
24
25 printf("\n     R1=%.2f kohms",R1*10^(-3));
26
27 printf("\n     R2=%.2f kohms",R2*10^(-3));
28
29 printf("\n     C=%.1f uF",C*10^6);
30
31 wxact=1.27*10^3;
```

```
32
33 T=(1+(%i*(wxact/wz)))/(((%i*wxact)/Kv)*(1+((%i*wxact
   )/wp)));
34
35 Tang=((180/%pi)*atan(imag(T)/real(T)))-180;
36
37 PMact=180+Tang;
38
39 printf("\n\n(b) Actual Value of wx=%.2f krad/s",
   wxact*10^(-3));
40
41 printf("\n     Actual Phase Margin (PM)=%.f deg",
   PMact);
```

Scilab code Exa 13.5 Damping Characterstics of Phase Locked Loop

```
1 //Example 13.5
2
3 clear;
4
5 clc;
6
7 Kv=10^4;
8
9 wz=10^3;
10
11 wp=(wz^2)/Kv;
12
13 wn=sqrt(wp*Kv);
14
15 zeta=(wn/(2*wz))*(1+(wz/Kv));
16
17 wmin3dBh=wn*sqrt(1+(2*(zeta^2))+sqrt(1+((1+(2*(zeta
   ^2)))^2)));
18
```

```
19 tau=1/wn;
20
21 printf("(a) zeta=%.2f",zeta);
22
23 printf("\n     tau=%.d ms",tau*10^3);
24
25 printf("\n     w-3dB=%.1f krad/s",wmin3dBh*10^(-3));
26
27 y=poly(0,'s')
28
29 Hs=((((2*zeta)-(wn/Kv))*(y/wn))+1)/(((y/wn)^2)+(2*
      zeta*(y/wn))+1);
30
31 r=real(roots(((y/wn)^2)+(2*zeta*(y/wn))+1));
32
33 i=imag(roots(((y/wn)^2)+(2*zeta*(y/wn))+1));
34
35 pr=r(1,1);
36
37 pi=abs(i(1,1));
38
39 printf("\n\n(b) Step Response of ve(t)=(|wi|/Ko)
      *[1-(A*exp(%.ft)*cos(",pr);
40
41 printf("%.ft+phi))]",pi);
42
43 wm=1*10^3;
44
45 vewirat=1/(1+(%i*(wm/Kv)));
46
47 ratm=1.286;
48
49 rata=45;
50
51 printf("\n     AC Response of ve(t)=(|wi|/Ko)*%.3f*
      cos(",ratm);
52
53 printf("%.f*t-",wm);
```

```
54
55 printf("%.f degrees)",rata);
```

Scilab code Exa 13.6 Filter Design Criteria

```
1  //Example 13.6
2
3  clear;
4
5  clc;
6
7  w3dB=1*10^3;
8
9  zeta=1/sqrt(2);
10
11 wn=w3dB/2;
12
13 tau=1/wn;
14
15 Kv=10^4;//from Example 13.4
16
17 wp=(wn^2)/Kv;
18
19 wz=wn/(2*zeta);
20
21 C=1*10^(-6);
22
23 R2=(1/(wz*C));
24
25 R1=(1/(wp*C))-R2;
26
27 x=poly(0,'wx');
28
29 y=((1-((x/wn)^2))^2)+(((2*zeta*x)/wn)^2)-(1+(((2*
   zeta*x)/wn)^2))
```

```
30
31 wx=roots(y);
32
33 wxact=wx(1,1);
34
35 s=%i*wxact;
36
37 T=((((2*zeta)-(wn/Kv))*(s/wn))+1)/(((s/wn)^2)+(2*
      zeta*(s/wn))+1);
38
39 Tang=180+(atan(imag(T)/real(T))*(180/%pi));
40
41 PM=180-Tang;
42
43 C2=C/10;
44
45 printf("tau=%.d ms",tau*10^(3));
46
47 printf("\nPM=%.f deg",PM+12);
48
49 printf("\nC2=%.1f uF",C2*10^6);
```

Scilab code Exa 13.7 Designing with PLLs

```
1 //Example 13.7
2
3 clear;
4
5 clc;
6
7 f0=1*10^6;
8
9 fR=((0.5)/2)*10^6;
10
11 vEmax=3.9;
```

```
12
13 vEmin=1.1;
14
15 Ko=(2*%pi*2*fR)/(vEmax-vEmin);
16
17 R1=95.3*10^3;//obtained from PLL's data sheet
18
19 R2=130*10^3;//obtained from PLL's data sheet
20
21 C=100*10^(-12);//obtained from PLL's data sheet
22
23 VDD=5;
24
25 Kd=VDD/%pi;
26
27 Kv=Kd*Ko;
28
29 zeta=0.707;
30
31 fm=1*10^3;
32
33 fmin3dB=fm*10;
34
35 w3dB=2*%pi*fmin3dB;
36
37 wn=w3dB/2;
38
39 wp=(wn^2)/Kv;
40
41 wz=wn/(2*zeta);
42
43 printf("R1=%.1f kohms",R1*10^(-3));
44
45 printf("\nR2=%.f kohms",R2*10^(-3));
46
47 printf("\nC=%.f pF",C*10^12);
48
49 //Filter Components are taken from figure 13.33, as
```

```
    no procedure is mentioned for designing the
    filter
50
51 R3=80.6*10^3;
52
53 R4=2*10^3;
54
55 C1=22*10^(-9);
56
57 C2=10*10^(-9);
58
59 printf("\nFilter Components :");
60
61 printf("\nR3=%.1f kohms",R3*10^(-3));
62
63 printf("\nC1=%.f nF",C1*10^9);
64
65 printf("\nR4=%.f kohms",R4*10^(-3));
66
67 printf("\nC2=%.f nF",C2*10^9);
```

Scilab code Exa 13.8 Designing Frequency Synthesizer using PLL

```
1 //Example 13.8
2
3 clear;
4
5 clc;
6
7 fOmin=1*10^6;
8
9 fI=1*10^3;
10
11 fOmax=2*10^6;
12
```

```
13 Nmin=fOmin/fI;
14
15 Nmax=fOmax/fI;
16
17 f0=(fOmin+fOmax)/2;
18
19 fR=f0/2;
20
21 vEmax=3.9;
22
23 vEmin=1.1;
24
25 Ko=(2*%pi*2*fR)/(vEmax-vEmin);
26
27 R1=28*10^3;
28
29 R2=287*10^3;
30
31 C=110*10^(-12);
32
33 VDD=5;
34
35 Kd=5/(4*%pi);
36
37 Kv=10^4;
38
39 Nmean=sqrt(Nmin*Nmax);
40
41 Kvmean=(Kd*Ko)/Nmean;
42
43 zeta=0.707;
44
45 fI=1*10^3;
46
47 wI=2*%pi*fI;
48
49 wn=wI/20;
50
```

```
wp=(wn^2)/Kv;

wz=wn/(2*zeta);

printf("R1=%.1f kohms",R1*10^(-3));

printf("\nR2=%.f kohms",R2*10^(-3));

printf("\nC=%.f pF",C*10^12);

printf("\nfI=%.d kHz",fI*10^(-3));

R3=6.17*10^3;

R4=3.45*10^3;

C1=1*10^(-6);

//Filter Components are taken from figure 13.33, as
   no procedure is mentioned for designing the
   filter

printf("\nFilter Components :");

printf("\nR3=%.2f kohms",R3*10^(-3));

printf("\nC1=%.f uF",C1*10^6);

printf("\nR4=%.2f kohms",R4*10^(-3));
```