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

August 10, 2013

[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

List of Figures

Chapter 1

Operational Amplifier Fundamentals

Scilab code Exa 1.1.a Amplifier Fundamentals

```
//Example 1.1(a)

clear;

clc;

Ri=100*10^3;//Input Resistance

Aoc=100;//Open Circuit Gain

Ro=1;//Output Resistance

Rs=25*10^3;//Source Resistance

RL=3;//Load Resistance

Av=(Ri/(Rs+Ri))*Aoc*(RL/(Ro+RL));//Overall Gain

Vredin=(Ri/(Ri+Rs))*100;//Percentage Reduction in
```

```
    Source Voltage due to Input Loading

Vredo=(RL/(Ro+RL))*100;//Percentage Reduction in
    Output Voltage due to output loading

printf("Overall Gain (Av)=%.2f V/V",Av);

printf("\nPercentage Input Loading=%.2f",Vredin);

printf("\nPercentage Output Loading=%.2f",Vredo);
```

Scilab code Exa 1.1.b Amplifier Fundamentals

```
//Example 1.1(b)

clear;

clc;

Ri=100*10^3;//Input Resistance

Aoc=100;//Open Circuit Gain

Ro=1;//Output Resistance

Rs=50*10^3;//Source Resistance

RL=4;//Load Resistance

Av=(Ri/(Rs+Ri))*Aoc*(RL/(Ro+RL));//Overall Gain

Vredin=(Ri/(Ri+Rs))*100;//Percentage Reduction in
    Source Voltage due to Input Loading

Vredo=(RL/(Ro+RL))*100;//Percentage Reduction in
```

```
    Output Voltage due to output loading

printf("Overall Gain (Av)=%.2f V/V",Av);

printf("\nPercentage Input Loading=%.2f (Not
    mentioned in book)",Vredin);

printf("\nPercentage Output Loading=%.2f (Not
    mentioned in book)",Vredo);
```

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

```
//Example 1.2(a)

clear;

clc;

Vi=1;//Input Voltage

R1=2*10^3;

R2=18*10^3;

a=10^2;//Open Loop Gain

A=(1+(R2/R1))*(1+(1+(R2/R1))/a)^(-1);//Overall Gain

Vo=Vi*A;//Output Voltage

printf("Output Voltage (Vo)=%.3f V",Vo);
```

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

```
//Example 1.2(b)

clear;

clc;

Vi=1;//Input Voltage

R1=2*10^3;

R2=18*10^3;

a=10^4;//Open Loop Gain

A=(1+(R2/R1))*(1+(1+(R2/R1))/a)^(-1);//Overall Gain

Vo=Vi*A;//Output Voltage

printf("Output Voltage (Vo)=%.3f V",Vo);
```

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

```
//Example 1.2(c)

clear;

clc;

Vi=1;//Input Voltage

R1=2*10^3;

R2=18*10^3;

a=10^6;//Open Loop Gain
```

```
A=(1+(R2/R1))*(1+(1+(R2/R1))/a)^(-1);//Overall Gain

Vo=Vi*A;//Output Voltage

printf("Output Voltage (Vo)=%.4f V",Vo);
```

Scilab code Exa 1.3 Inverting Amplifier Ideal Closed Loop Characterstics

```
//Example 1.3

clear;

clc;

R1=10*10^3;

R2=100*10^3;

Ri=R1;//Input Resistance

Ro=0;//Output Resistance

A=-(R2/R1);// Ideal Overall Gain

printf("Ri=%.2f kohms",(Ri/1000));

printf("\nRo=%.f ohms",Ro);

printf("\nA=%.2f V/V",A);
```

Scilab code Exa 1.4 Designing Summing Amplifiers

```
//Example 1.4

clear;

clc;

Rf=120*10^3;//Assuming feedback resistance Rf
   =120*10^3
//Imposing in equation Vo=-((Rf/R1)V1+(Rf/R2)V2+(RF/
   R3)V3)

R1=Rf/6;//From coefficient of V1

R2=Rf/8;//From coefficient of V2

R3=Rf/4;//From coefficient of V3

printf("Designed Summing Amplifier :");

printf("\n R1=%.2f kohms",(R1/1000));

printf("\n R2=%.2f kohms",(R2/1000));

printf("\n R3=%.2f kohms",(R3/1000));

printf("\n Rf=%.2f kohms",(Rf/1000));
```

Scilab code Exa 1.5 Designing Function Generators

```
//Example 1.5

clear;

clc;

```

```
Rf=100*10^3;//Assuming Feedback Resistance Rf

Vee=-15;

//Imposing Vo=-(Rf/R1)Vi-(Rf/R2)(-15)=-10Vi+5

R1=Rf/10;

R2=(Rf*15)/5;

printf("Designed Function Generator :");

printf("\n R1=%.2f kohms",(R1/1000));

printf("\n R2=%.2f kohms",(R2/1000));

printf("\n Rf=%.2f kohms",(Rf/1000));
```

Scilab code Exa 1.6 Designing Difference Amplifier

```
//Example 1.6

clear;

clc;

Ri1=100*10^3;

Ri2=100*10^3;

//Using standard equation for difference amplifier
//vo=(R2/R1)(((1+(R1/R2))/(1+(R3/R4)))v2-v1)=v2-3v1
//Ri1=R1, Ri2=R3+R4, Ro=0

R1=Ri1;
```

```
R2=3*R1;

//Solving the equations R3+R4=Ri2=100kohms and 3[(1+(1/3))/(1+(R3/R4))]=1
//3[4/3((R3+R4)/R4)]=1
//As R3+R4=100 -> 4/(100/R4)=1 -> (4R4)/100=1 -> R4/25=1 -> R4=25kohms

R4=25*10^3;//By solving the equations mentioned above

R3=Ri2-R4;//From standard equations

printf("Designed Difference Amplifier :");

printf("\nR1=%.2f kohms",(R1/1000));

printf("\nR2=%.2f kohms",(R2/1000));

printf("\nR3=%.2f kohms",(R3/1000));

printf("\nR4=%.2f kohms",(R4/1000));
```

Scilab code Exa 1.7.a Application of Negative Feedback

```
//Example 1.7(a)

clear;

clc;

//1.7(a)
Gerrormax=0.1;// Maximum Gain Error Percentage

```

```
//But Gerror100/T ->Gerrormax=100/Tmin -> Tmin=100/Gerrormax

Tmin=100/Gerrormax;

printf("Loop Gain (T)>=%.2f",Tmin);
```

Scilab code Exa 1.7.b Application of Negative Feedback

```
//Example 1.7(b)

clear;

clc;

Gerrormax=0.1;// Maximum Gain Error Percentage

//But Gerror100/T ->Gerrormax=100/Tmin -> Tmin=100/Gerrormax

Tmin=100/Gerrormax;

Aideal=100;

b=1/Aideal;//Feedback Factor

amin=Tmin/b;//Minimum Open Loop Gain

printf("\nOpen Loop Gain (a)>=%.f",amin);
```

Scilab code Exa 1.7.c Application of Negative Feedback

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

```
clear;

clc;

Gerrormax=0.1;// Maximum Gain Error Percentage

//But Gerror100/T ->Gerrormax=100/Tmin -> Tmin=100/
    Gerrormax

Tmin=100/Gerrormax;

Aideal=100;

b=1/Aideal;//Feedback Factor

amin=Tmin/b;//Minimum Open Loop Gain

//Imposing A=a/(1+ab).We have a=10^5 and Aideal=100
    ->100=10^5/(1+10^5b)

y=poly(0,'x');

z=(100*amin)*y+(100-amin);//Solving the equation
    mentioned in above comment.

b=roots(z);

printf("Feedback Factor (b) for A=100 is =%.5f",b);
```

Scilab code Exa 1.8.a Calculating Gain Desensitivity

```
//Example 1.8(a)

clear;
```

```
clc;

a=10^5;//Open Loop Gain

b=10^(-3);//Feedback Factor

T=a*b;//return ratio or loop gain

d=1+T;//Desensitivity Factor

A=a/d;//Overall Gain

anew=a+(10/100)*a;//Increasing gain by 10%

Tnew=anew*b;//New return ratio or loop gain

dnew=1+Tnew;//New Desensitivity Factor

Anew=anew/dnew;//Overall Gain

Achange=((Anew-A)/A)*100;//Percentage Change in
    Overall Gain

printf("Percentage change in A =%.1f percent",
    Achange);
```

Scilab code Exa 1.8.b Calculating Gain Desensitivity

```
//Example 1.8(b)

clear;

clc;

```

```
a=10^5;//Open Loop Gain

b=1;//Feedback Factor

T=a*b;//return ratio or loop gain

d=1+T;//Desensitivity Factor

aperchange=10;//Percentage Change in a

Achange=(1/(1+T))*aperchange;//Percentage Change in
    Overall Gain

printf("Percentage change in A =%.4f",Achange);
```

Scilab code Exa 1.9.a Noninverting Configuration Characterstics

```
//Example 1.9(a)

clear;

clc;

rd=2*10^6;//Input Resistance

ro=75;//Output Resistance

a=200*10^3;//Open loop Gain

R1=1*10^3;

R2=999*10^3;

b=R1/(R1+R2);//Feedback Factor

```

```
T=a*b;//reteurn ratio or loop gain
Aapprox=(1+(R2/R1))*(1/(1+(1/T)))//Approximate Gain
Riapprox=rd*(1+T);//Approximate Input Resistance
Roapprox=ro/(1+T);
Anum=((1+(R2/R1))*a)+(ro/rd)//Numerator of exact Gain
Aden=1+a+(R2/R1)+((R2+ro)/rd)+(ro/R1);//Denominator of exact Gain
Aexact=Anum/Aden;//exact Gain
Ri1=rd*(1+(a/(1+((R2+ro)/R1))));
Ri2=(R1*(R2+ro))/(R1+R2+ro);
Riexact=Ri1+Ri2;//Exact Input Resistance
Ronum=ro;
Roden=1+((a+(ro/R1)+(ro/rd))/(1+(R2/R1)+(R2/rd)))
Roexact=Ronum/Roden;//Exact Output Resistance
//Ideal Value of input resistance Ri1 is infinity and ideal value of output resistance Ro1 is 0.
printf("Exact Value of A is =%.2f V/V",Aexact);
printf("\nApproximate Value of A is =%.3f V/V",Aapprox);
printf("\nIdeal Value of A is =%.3f V/V",1000);
```

```
printf("\nExact Value of Ri is =%.3f Mohms",Riexact
    /10^6);

printf("\nApproximate Value of Ri is =%.3f Mohms",
    Riapprox/10^6);

printf("\nIdeal Value of Ri is infinity");

printf("\nExact Value of Ro is =%.3f mohms",Roexact
    *10^3);

printf("\nApproximate Value of Ro is =%.3f mohms",
    Roapprox*10^3);

printf("\nApproximate Value of Ro is =%.3f ohms",0);
```

Scilab code Exa 1.9.b Noninverting Configuration Characterstics

```
//Example 1.9(b)

clear;

clc;

rd=2*10^6;//Input Resistance

ro=75;//Output Resistance

a=200*10^3;//Open loop Gain

printf("Note (as mentioned in the book): Because of
    much larger value, we simply ignore the exact
    calculations and use only the approximations.");

//R12=infinity
```

```
R2=0;

//b2=R12/(R12+R22) (Feedback Factor) will be equal to 1 as R12 tends to infinity and R22 is 0

b=1;//Feedback Factor

T=a*b;//reteurn ratio or loop gain

//Aapprox=(1+(R22/R12))*(1/(1+(1/T2)))(Approximate Gain) but R22/R12=0

Trec=1/T;

Aden=(1+Trec);

Anum=1;

Aapprox=Anum/Aden;//Approximate Gain

Riapprox=rd*(1+T);//Approximate Input Resistance

Roapprox=ro/(1+T);//Approximate Output Resistance

//Ideal Value of input resistance Ri2 is infinity and ideal value of output resistance Ro2 is 0.

printf("\nApproximate Value of A is =%.f V/V", Aapprox);

printf("\nIdeal Value of A is =%.2f V/V",1);

printf("\nApproximate Value of Ri is =%.3f Gohms", Riapprox/10^9);

printf("\nIdeal Value of Ri is infinity");

```

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

printf("\nApproximate Value of Ro is =%.f ohms",0);
```

Scilab code Exa 1.10.a Inverting Configuration Characterstics

```
//Example 1.10(a)

clear;

clc;

R1=100*10^3;

R2=100*10^3;

ro=75;//Output Resistance

a=200*10^3;//Open Loop Gain for ic741

b=R1/(R1+R2);//Feedback Factor

T=a*b;//Loop Gain

Trec=1/T;

Atemp=1+Trec;

Atempr=1/Atemp;

A=(-R2/R1)*Atempr;//Gain

Rnnum=R2+ro;

```

```
Rnden=1+a;

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)

Ri=R1+Rn;//Equivalent Input Resistance

Ro=ro/(1+T);//Equivalent Output Resistance

printf("A=%.5f V/V",A);

printf("\nRn=%.2f ohms ",Rn);//answer in textbook is wrong

printf("\nRi=%.2f kohms",(Ri/1000));

printf("\nRo=%.2f mohms",(Ro*1000));
```

Scilab code Exa 1.10.b Inverting Configuration Characterstics

```
//Example 1.10(b)

clear;

clc;

R1=1*10^3;

R2=1*10^6;

ro=75;//Output Resistance

a=200*10^3;//Open Loop Gain for ic741

```

```
b=R1/(R1+R2);//Feedback Factor

T=a*b;//Loop Gain

Trec=1/T;

Atemp=1+Trec;

Atempr=1/Atemp;

A=(-R2/R1)*Atempr;//Gain

Rnnum=R2+ro;

Rnden=1+a;

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)

Ri=R1+Rn;//Equivalent Input Resistance

Ro=ro/(1+T);//Equivalent Output Resistance

printf("A=%.5f V/V",A);

printf("\nRn=%.2f ohms",Rn);

printf("\nRi=%.3f kohms",(Ri/1000));

printf("\nRo=%.3f ohms",Ro);
```

Scilab code Exa 1.11.a Finding the Loop Gain

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

```
clear;

clc;

R1=1*10^6;

R2=1*10^6;

R3=100*10^3;

R4=1*10^3;

RL=2*10^3;//Load Resistance

A=-(R2/R1)*(1+(R3/R2)+(R3/R4));//Ideal Gain

printf("Ideal Gain of of the op amp (A)=%.2f V/V",A)
    ;
```

Scilab code Exa 1.11.b Finding the Loop Gain

```
//Example 1.11(b)

clear;

clc;

R1=1*10^6;

R2=1*10^6;

R3=100*10^3;

R4=1*10^3;
```

```
RL=2*10^3;//Load ResistanceR1=1*10^6;
R2=1*10^6;
R3=100*10^3;
R4=1*10^3;
RL=2*10^3;//Load Resistance
A=-(R2/R1)*(1+(R3/R2)+(R3/R4));//Ideal Gain
rd=1*10^6;//Internal input resistance
a=10^5;//Open Loop Gain
ro=100;
RA=(R1*rd)/(R1+rd);
RB=RA+R2;
RC=(RB*R4)/(RB+R4);
RD=RC+R3;
RE=(RD*RL)/(RD+RL);
RF=RE+ro;
c1=-(RA/RB);//vD=c1*v1
c2=(RC/RD);//v1=c2*vo
c3=(RE/RF);//vo=c3*vT
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

T=-c4;//T=(-vR/vT)=-c4 (Loop Gain)

Trec=1/T;

Atemp=1+Trec;

Aactual=A/Atemp;//Actual Gain

Adev=((Aactual-A)/A)*100;//Deviation in Gain

printf("Actual Gain of op amp=%.1f V/V",Aactual);

printf("\nPercentage Departure of Actual Gain from
    Ideal gain=%.2f",Adev);
```

Scilab code Exa 1.12.a Feedback Factor for Negative Feedback

```
//Example 1.12(a)

clear;

clc;

rd=1*10^6;//Internal Input Resistance

a=10^4;//Open Loop Gain

ro=100;//Internal Output Resistance

R1=10*10^3;//shown in Fig. 1.34a

R2=20*10^3;//shown in Fig. 1.34a

```

```
R3=30*10^3;//shown in Fig. 1.34a
R4=300*10^3;//Feedback Resistance (shown in Fig. 1.34a)
RL=2*10^3;//Load Resistance
RArec=((1/R1)+(1/R2)+(1/R3)+(1/rd))//Reciprocal of RA(parallel combination of R1, R2, R3 and rd)
RA=1/RArec;
RB=RA+R4;
RC=(RB*RL)/(RB+RL);
RD=RC+ro;
c1=(RA/RB);//vN=c1*vo
c2=(RC/RD);//vo=c2*vT
b=c1*c2;//Feedback Factor b=vN/vT=c1*c2
T=a*b;//Loop Gain
printf("b=%.3f V/V",b);
printf("\nT=%.2f",T);
```

Scilab code Exa 1.12.b Feedback Factor for Negative Feedback

```
//Example 1.12(b)
clear;
```

```
clc;

rd=1*10^6;//Internal Input Resistance

a=10^4;//Open Loop Gain

ro=100;//Internal Output Resistance

R1=10*10^3;//shown in Fig. 1.34a

R2=20*10^3;//shown in Fig. 1.34a

R3=30*10^3;//shown in Fig. 1.34a

R4=300*10^3;//Feedback Resistance (shown in Fig.
    1.34a)

RL=2*10^3;//Load Resistance

RArec=((1/R1)+(1/R2)+(1/R3)+(1/rd))//Reciprocal of
    RA(parallel combination of R1, R2, R3 and rd)

RA=1/RArec;

RB=RA+R4;

RC=(RB*RL)/(RB+RL);

RD=RC+ro;

c1=(RA/RB);//vN=c1*vo

c2=(RC/RD);//vo=c2*vT

b=c1*c2;//Feedback Factor b=vN/vT=c1*c2

T=a*b;//Loop Gain
```

```
//1.12(b)

p1=-(R4/R1);

p2=-(R4/R2);

p3=-(R4/R3);

//vo(ideal)=p1*v1+p2*v2+p3*v3

Trec=1/T;

ctempr=1+Trec;

ctemp=1/ctempr;

p1act=-(R4/R1)*ctemp;

p2act=-(R4/R2)*ctemp;

p3act=-(R4/R3)*ctemp;

printf("Ideal Transfer Characterstic of the circuit
    vo=-(%.2f*v1",-p1);

printf("+%.2f*v2",-p2);

printf("+%.2f*v3)",-p3);

printf("\nActual Transfer Characterstic of the
    circuit vo=-(%.2f*v1",-p1act);

printf("+%.2f*v2",-p2act);

printf("+%.2f*v3)",-p3act);
```

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

```
//Example 1.13

clear;

clc;

R1=30*10^3;//From Fig. 1.13b

R3=20*10^3;//Feedback Resistance obtained from Fig.
    1.13b

R2=10*10^3;//Load Resistance obtained from Fig. 1.13
    b

rd=100*10^3;//Internal Input Resistance

ro=100;//Internal Output Resistance

bNnum=((R1*rd)/(R1+rd))+R3;

bNden=ro+R2+bNnum;

bN=bNnum/bNden;

bPnum=R3;

bPden=bNden;

bP=bPnum/bPden;

b=bN-bP;//Feedback Factor
```

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

Scilab code Exa 1.14.a Current Flow and Power Dissipation

```
//Example 1.14(a)

clear;

clc;

R1=10*10^3;

R2=20*10^3;

RL=2*10^3;///Load Resistance

vI=3;//Input Voltage

IQ=0.5*10^(-3);

vO=-(R2/R1)*vI;//Output Voltage

iL=-vO/RL;//Current through RL

i1=vI/R1;//Cuurent through R1

i2=i1;//Current through R2 (as current sunk by the
    op amp is 0)

iO=i2+iL;//Output Current

iCC=IQ;

iEE=iCC+iO;
```

```
printf("iCC=%.2f mA",(iCC*1000));

printf("\niEE=%.2f mA",(iEE*1000));

printf("\niO=%.2f mA",(iO*1000));
```

Scilab code Exa 1.14.b Current Flow and Power Dissipation

```
//Example 1.14(b)

clear;

clc;

R1=10*10^3;

R2=20*10^3;

RL=2*10^3;///Load Resistance

vI=3;//Input Voltage

IQ=0.5*10^(-3);

vO=-(R2/R1)*vI;//Output Voltage

iL=-vO/RL;//Current through RL

i1=vI/R1;//Cuurent through R1

i2=i1;//Current through R2 (as current sunk by the
    op amp is 0)

iO=i2+iL;//Output Current
```

```
iCC=IQ;

iEE=iCC+iO;

VCC=15;

VEE=-15;

IQ=0.5*10^(-3);

pOA=(VCC-VEE)*IQ+(vO-VEE)*iO;//Power Dissipated in
    the Op Amp

printf("Power Dissipated inside the op amp=%.2f mW"
    ,(pOA*1000));
```

Scilab code Exa 1.15.a Designing variable dc source

```
//Example 1.15(a)

clear;

clc;

Rp=100*10^3;//Potentiometer Resistance

VCC=15;

VEE=-15;

//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)

//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.

y=poly(0,'x');

p=5*(y+50*(10^3))-(15*y);

RA=roots(p);

RB=RA;

printf("Designed Source :");

printf("\nRA=%.2f kohms",(RA/1000));//mentioned in the diagram

printf("\nRB=%.2f kohms",(RB/1000));//mentioned in the diagram

printf("\nRpot=%.2f kohms",(Rp/1000));//mentioned in the diagram
```

Scilab code Exa 1.15.b Designing variable dc source

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

```
clear;

clc;

Rp=100*10^3;//Potentiometer Resistance

VCC=15;

VEE=-15;

//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)

//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.

y=poly(0,'x');

p=5*(y+50*(10^3))-(15*y);

RA=roots(p);

RB=RA;

RL=1*10^3;//Load Resistance

```

```
vS=10;//Source voltage

iL=vS/RL;//Current drawn by the load

a=200*10^3;//Open Loop Gain (defined for 741)

b=1;//Feedback Factor (Refer Fig. 1.38)

T=a*b;//Loop Gain

ro=75;//Internal Output Resistance (defined for 741)

Ro=ro/(1+T);//Ouput Resistance

vSchange=Ro*iL;//Change in Voltage

printf("Change in Vs=%.3f uV",(vSchange*10^6));
```

Scilab code Exa 1.16 Inverting Amplifier driven into saturation

```
//Example 1.16

clear;

clc;

//Part(I)
//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.
```

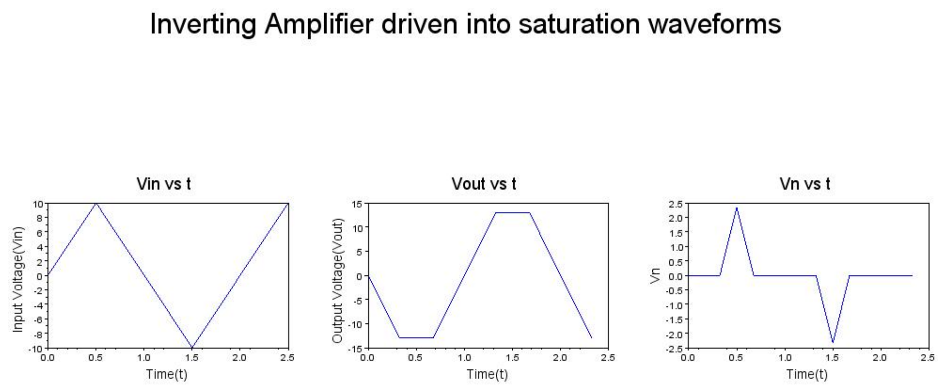

Figure 1.1: Inverting Amplifier driven into saturation

```

VCC=13;

VEE=-13;

A=-2;//Gain

t1=[0:10^(-4):0.5];

t2=[0.5:10^(-4):1.5];

t3=[1.5:10^(-4):2.5];

vt1=20*t1;

vt2=20*(1-t2);

vt3=20*(t3-2);

```

```
subplot(131);

title("Inverting Amplifier driven into saturation waveforms ","fontsize",6);

subplot(334);

plot(t1,vt1);

plot(t2,vt2);

plot(t3,vt3);

xlabel("Time(t)","fontsize",3);

ylabel("Input Voltage(Vin)","fontsize",3);

title("Vin vs t","fontsize",4);

//Part(II)
//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.

vIbor=VCC/2;

to1min=0;

to1max=6.5/20;

to2min=1-(6.5/20);

to2max=1+(6.5/20);

```

```
to3min=2-(6.5/20);

to3max=2+(6.5/20);

to1=[to1min:10^(-4):to1max];

to2=[to2min:10^(-4):to2max];

to3=[to3min:10^(-4):to3max];

to1imin=to1max;

to1imax=to2min;

to2imin=to2max;

to2imax=to3min;

to1i=[to1imin:10^(-4):to1imax];

to2i=[to2imin:10^(-4):to2imax];

vo1=-13*(to1-to1min)/(to1max-to1min);

vo1i=-13;

vo2=(((13+13)/(to2max-to2min))*(to2-to2min))-13;

vo2i=13;

vo3=(((13+13)/(to3min-to3max))*(to3-to3max))-13;

subplot(335);

plot(to1,vo1);

plot(to1i,vo1i);
```

```
plot(to2,vo2);

plot(to2i,vo2i);

plot(to3,vo3);

ylabel("Output Voltage(Vout)","fontsize",3);

xlabel("Time(t)","fontsize",3);

title("Vout vs t","fontsize",4);

//Part(III)
//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.

vIbor=VCC/2;

tNi1min=0;

tNi1max=6.5/20;

tNi2min=1-(6.5/20);

tNi2max=1+(6.5/20);

tNi3min=2-(6.5/20);

tNi3max=2+(6.5/20);

tNi1=[tNi1min:10^(-4):tNi1max];

```

```
tNi2=[tNi2min:10^(-4):tNi2max];

tNi3=[tNi3min:10^(-4):tNi3max];

tN11min=tNi1max;

tN11max=(tNi2min+tNi1max)/2;

tN12min=tN11max;

tN12max=tNi2min;

tN21min=tNi2max;

tN21max=(tNi2max+tNi3min)/2;

tN22min=tN21max;

tN22max=tNi3min;

tN11=[tN11min:10^(-4):tN11max];

tN12=[tN12min:10^(-4):tN12max];

tN21=[tN21min:10^(-4):tN21max];

tN22=[tN22min:10^(-4):tN22max];

vNi1=0;

vN11=(2.33/(tN11max-tN11min))*(tN11-tN11min);

vN12=-(2.33/(tN12max-tN12min))*(tN12-tN12max);

vNi2=0;

vN21=-(2.33/(tN21max-tN21min))*(tN21-tN21min);

```

```
vN22=(2.33/(tN22max-tN22min))*(tN22-tN22max);

vNi3=0;

subplot(336);

plot(tNi1,vNi1);

plot(tN11,vN11);

plot(tN12,vN12);

plot(tNi2,vNi2);

plot(tN21,vN21);

plot(tN22,vN22);

plot(tNi3,vNi3);

xlabel("Time(t)","fontsize",3);

ylabel("Vn","fontsize",3);

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

```
//Example 2.1

clear;

clc;

R=1*10^6;

a=200*10^3;//Open Loop Gain for ic741

rd=2*10^6;//defined for 741

ro=75;//internal output resistance defined for 741

Tnum=a*rd;

Tden=rd+R+ro;

T=Tnum/Tden;//Loop Gain
```

```
Anum=-R;

Aden=1+(1/T);

A=Anum/Aden;//Oveall Gain

Rinumn=rd*(R+ro);

Rinumd=rd+R+ro;

Rinum=Rinumn/Rinumd;

Riden=1+T;

Ri=Rinum/Riden;//Input resistance

Ronum=ro;

Roden=1+T;

Ro=Ronum/Roden;//Ouput Resistance (Value obtained
    for Ro in the book is wrong)

printf("T=%.f",T);

printf("\nA=%.6f V/uA",A*10^(-6));

printf("\nRi=%.1f ohms",Ri);

printf("\nRo=%.3f mohms",(Ro*(10^3)));//answer in
    textbook is wrong
```

Scilab code Exa 2.2 Designing High Sensitivity I V Conveter

```
//Example 2.2

clear;

clc;

sen=0.1*10^9;//sensitivity in V/A

R=1*10^6;//Assumption

//sen=k*R ->k=sen/R

k=sen/R;

R1=1*10^3;//Assumption

//k=1+(R2/R1)+(R2/R) ->R2=(k-1)/((1/R1)+(1/R))

R2num=k-1;

R2den=((1/R1)+(1/R));

R2=R2num/R2den;

printf("Designed High Sensitivity I-V Converter :");

printf("\nR=%.f Mohms",(R*10^(-6)));

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

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

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

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

```
clear;

clc;

vI=5;//Input Voltage

R=10*10^3;

Vsat=13;//Saturation Voltage

iO=vI/R;//(from right to left for Fig.2.4(a) and
   from left to right for Fig.2.4(b))

printf("iO=%.1f mA",iO*10^3);
```

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

```
//Example 2.3(b)

clear;

clc;

vI=5;//Input Voltage

R=10*10^3;

Vsat=13;//Saturation Voltage

iO=vI/R;//iO for Circuit shown in Fig.2.4(a) (from
   right to left)

//For Circuit shown in Fig.2.4(a) VoL1<vL1<VoH1

```

```
VoL1=-Vsat-vI;

VoH1=Vsat-vI;

printf("For Circuit shown in Fig.2.4(a) %.1f V< vL <
    ",VoL1);

printf("%.1f V",VoH1);

//For Circuit shown in Fig.2.4(b) VoL2<vL2<VoH2

VoL2=-Vsat;

VoH2=Vsat;

printf("\nFor Circuit shown in Fig.2.4(b) %.1f V< vL
    <",VoL2);

printf("%.1f V",VoH2);
```

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

```
//Example 2.3(c)

clear;

clc;

vI=5;//Input Voltage

R=10*10^3;

Vsat=13;//Saturation Voltage

iO=vI/R;//iO for Circuit shown in Fig.2.4(a) (from
```

```
right to left)

//For Circuit shown in Fig.2.4(a)

VoL1=-Vsat-vI;

VoH1=Vsat-vI;

//For Circuit shown in Fig.2.4(b) VoL2<vL2<VoH2

VoL2=-Vsat;

VoH2=Vsat;

RLmax1=VoH1/iO;//Maximum Possible value of RL

//For Circuit shown in Fig.2.4(b)

RLmax2=VoH2/iO;//Maximum Possible Value of Rl

printf("Max Value of RL for Circuit shown in Fig.2.4(a)= %.f kohms",RLmax1*10^(-3));

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

```
//Example 2.4

clear;

clc;

```

```
Vsat=15;//Saturation Voltage

vL=10;

iL=10^(-3);//Load Current

vI=Vsat;//Assuming Vsat as the input Voltage

R1=vI/iL;//(Tolerance-1%)

//vL<=(R1/(R1+R2))*Vsat, Vsat=15V ->10<=(R1/(R1+R2))*15 or 10=(R1/(R1+R2))*13 (approx)
//->R2=((13*R1)/vL)-R1

R2=((13*R1)/vL)-R1;//(Tolerance-1%)

R3=R1;//(Tolerance-1%)

R4=R2;//(Tolerance-1%)

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

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

printf("\nR2=%.1f kohms",R2*10^(-3));

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

printf("\nR4=%.1f kohms",R4*10^(-3));
```

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

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

```
clear;

clc;

R1=15*10^3;//From the result of Example 2.4

p=0.01;//For 1% tolerance p=t/100=1/100=0.01

emax=4*p;//imbalace factor

Romin=R1/emax;

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

```
//Example 2.5(b)

clear;

clc;

R1=15*10^3;//From the result of Example 2.4

p=0.001;//For 1% tolerance p=t/100=1/100=0.01

emax=4*p;//imbalace factor

Romin=R1/emax;

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

```
//Example 2.5(c)

clear;

clc;

R1=15*10^3;//From the result of Example 2.4

Romin=50*10^6;

emax=R1/Romin;

p=emax/4;

pper=p*100;

printf("Resistance tolerance Required=%.5f percent",
   pper);
```

Scilab code Exa 2.6 Howland Circuit Calibration

```
//Example 2.6

clear;

clc;

//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
```

```
clear;

clc;

R1=15*10^3;

R2=3*10^3;

R3=R1;

R4=R2;

a=200*10^3;

Ro1num=R1*R2;

Ro1den=R1+R2;

Ro1=Ro1num/Ro1den;

Ro2num=a;

Ro2den=(1+(R2/R1));

Ro2=Ro2num/Ro2den;

Ro=Ro1*(1+Ro2);//Output resistance

printf("Output Resistance (Ro)=%.f Mohms",Ro*10^(-6)
   );
```

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

```
//Example 2.8(a)

```

```
clear;

clc;

R1=10*10^3;

R3=R1;

R2=100*10^3;

R4=R2;

//For first pair of inputs (v1, v2)=(-0.1 V, +0.1V)
v11=-0.1;

v21=0.1;

vo1=(R2/R1)*(v21-v11);

vcm1=(v11+v21)/2;

//For Second pair of inputs (v1, v2)=(4.9 V, 5.1V)

v12=4.9;

v22=5.1;

vo2=(R2/R1)*(v22-v12);

vcm2=(v12+v22)/2;

//For Third pair of inputs (v1, v2)=(9.9 V, 10.1 V)

v13=9.9;

v23=10.1;

vo3=(R2/R1)*(v23-v13);
```

```
vcm3=(v13+v23)/2;

printf("vo for (-0.1 V,+0.1 V)=%.2f V",vo1);

printf("\nvo for (4.9 V,5.1 V)=%.2f V",vo2);

printf("\nvo for (9.9 V,10.1 V)=%.2f V",vo3);
```

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

```
//Example 2.8(b)

clear;

clc;

R1=10*10^3;

R2=98*10^3;

R3=9.9*10^3;

R4=103*10^3;

//For first pair of inputs (v1, v2)=(-0.1 V, +0.1V)
v11=-0.1;

v21=0.1;

vo1=(R2/R1)*(v21-v11);

vcm1=(v11+v21)/2;

//For Second pair of inputs (v1, v2)=(4.9 V, 5.1V)
```

```
v12=4.9;

v22=5.1;

vo2=(R2/R1)*(v22-v12);

vcm2=(v12+v22)/2;

//For Third pair of inputs (v1, v2)=(9.9 V, 10.1 V)

v13=9.9;

v23=10.1;

vo3=(R2/R1)*(v23-v13);

vcm3=(v13+v23)/2;

//vO=A2*v2-A1*v1

A2num=(1+(R2/R1));

A2den=(1+(R3/R4));

A2=A2num/A2den;

A1=R2/R1;

//For first pair of inputs (v1, v2)=(-0.1 V, +0.1V)

vo1m=A2*v21-A1*v11;

//For Second pair of inputs (v1, v2)=(4.9 V, 5.1V)

vo2m=A2*v22-A1*v12;

//For Third pair of inputs (v1, v2)=(9.9 V, 10.1 V)
```

```
vo3m=A2*v23-A1*v13;

printf("vo for (-0.1 V,+0.1 V)=%.3f V",vo1m);

printf("\nvo for (4.9 V,5.1 V)=%.3f V",vo2m);

printf("\nvo for (9.9 V,10.1 V)=%.3f V",vo3m);
```

Scilab code Exa 2.9 Common Mode Rejection Ratio for op amp

```
//Example 2.9

clear;

clc;

R1=10*10^3;

R3=R1;

R2=100*10^3;

R4=R2;

p=0.01;

emax=4*p;

Adm1=R2/R1;

Adm2n=emax*(R1+2*R2);

Adm2d=2*(R1+R2);

```

```
Adm2=1-(Adm2n/Adm2d);

Admin=Adm1*Adm2;

Acmax=(R2/(R1+R2))*emax;

cmrrm=20*log10(Admin/Acmax);

printf("(a) CMRR min=%.1f dB",cmrrm);

//2.9(b)

vdm=0;

vcm=10;

vO=vcm*Acmax+vdm*Admin;

printf("\n(b) Output Error vO=%.3f V",vO);

//2.9(c)

//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))

emax1=(1+(R2/R1))/(10^(4));

p=emax1/4;

pper=p*100;

printf("\n(c) Required Resistance Tolerance=%.4f percent",pper);
```

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

```
//Example 2.10(a)

clear;

clc;

Amin=1;

Amax=10^3;

AI=0.5;

R1=100*10^3;//Tolerance (1%)

R2=AI*R1;//Tolerance (1%)

AImin=Amin/AI;

AImax=Amax/AI;

//AImin<=AI<=AImax
//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%)

printf("Designed Instrumentation Amplifier :");

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

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

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

printf("\nR4=%.f ohms",R4);
```

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

```
//Example 2.10(b)

clear;

clc;

Amin=1;

Amax=10^3;

AI=0.5;

R1=100*10^3;//Tolerance (1%)

R2=AI*R1;//Tolerance (1%)

AImin=Amin/AI;

AImax=Amax/AI;

```

```
//AImin<=AI<=AImax
//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%)

p=0.01;

e=4*p*R2;

R5=100*10^3;

R2red=R2-e-500;//to be on the safer side 0.5 kohms more is reduced

Rpot=2*(R2-R2red);//Potentiometer Resistance

//Circuit is shown in Fig.2.21 in the book

printf("Designed Instrumentation Amplifier with trimmed resistances :");

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

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

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

```
printf("\nR4=%.f ohms",R4);

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

printf("\nR6=%.2f kohms",R2red*10^(-3));

printf("\nR7=%.2f kohms",Rpot*10^(-3));
```

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

```
//Example 2.10(c)

clear;

clc;

Amin=1;

Amax=10^3;

AI=0.5;

R1=100*10^3;//Tolerance (1%)

R2=AI*R1;//Tolerance (1%)

AImin=Amin/AI;

AImax=Amax/AI;

//AImin<=AI<=AImax
//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;
```

```
R3=R0*(1+alpha*T3);

printf("\nR(-15 deg Celsius)=%.2f ohms",R3);
```

Scilab code Exa 2.11.c Study of Resistance Temperarure Detector

```
//Example 2.11(c)

clear;

clc;

R0=100;

alpha=0.00392;

dT=10;

delta=alpha*dT;

deltaper=delta*100;

dR=R0*delta;

printf("Change in R=%.2f ohms",dR);

printf("\nPercentage Deviation=%.2f percent",
   deltaper);
```

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

```
//Example 2.12(a)

clear;

clc;

R0=100;//Data taken from Example 2.11

alpha=0.00392;//Data taken from Example 2.11

Vref=15;

Prtd=0.2*10^(-3);

i=(Prtd/R0)^(0.5)-(0.41*10^(-3));

R1=(Vref/i);

delta=alpha*1;//Fractional Deviation for 1 degree
    celsius change in temperature

s=0.1;//Output Voltage for 1 degree Celsius
    temperature change

A1=s*(2+(R1/R0)+(R0/R1));

A2=Vref*delta;

A=(A1/A2)+1.0555913;

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

printf("\nA=%.1f V/V",A);
```

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

```
//Example 2.12(b)

clear;

clc;

R0=100;//Data taken from Example 2.11

alpha=0.00392;//Data taken from Example 2.11

Vref=15;

Prtd=0.2*10^(-3);

i=(Prtd/R0)^(0.5)-(0.41*10^(-3));

R1=(Vref/i);

delta=alpha*1;//Fractional Deviation for 1 degree
    celsius change in temperature

s=0.1;//Output Voltage for 1 degree Celsius
    temperature change

A1=s*(2+(R1/R0)+(R0/R1));

A2=Vref*delta;

A=A1/A2;

dT=100;

d2=alpha*dT;

vO1num=A*Vref*d2;
```

```
v01den=1+(R1/R0)+((1+(R0/R1))*(1+d2));

v01=v01num/v01den;

v02num=A*Vref*d2;

v02den=(2+(R1/R0)+(R0/R1));

v02=v02num/v02den;

v0change=v02-v01;

printf("vO(100 deg Celsius)=%.3f V",v01);

Tchange=v0change/s;

printf("\nEquivalent Temperature Error=%.2f deg
    Celsius",Tchange);
```

Scilab code Exa 2.13.a Transducer Bridge Calibration

```
//Example 2.13(a)

clear;

clc;

R0=100;//Data taken from Example 2.11

alpha=0.00392;//Data taken from Example 2.11

Vref=15;

P=0.2*10^(-3);
```

```

i=(P/R0)^(0.5)-(0.41*10^(-3));

pV=0.05;

Vrefc=pV*Vref+0.25;

Vrefr=Vref-Vrefc;

R3=2/(2*i);

//R0+R1+(R2/2)=Vrefr/i;

Rtot=Vrefr/i;

p=0.01;

R2=(2*p*Rtot)+221.1748472;//220 ohms are added to be
    on the safe side

R1=(Rtot-(R2/2)-R0)+108.15494;//Tolerance 1%

v0=9.97;//Data from Example 2.12

R1u=R1+(R2/2);

dT=1;//obtained from Example 2.12

d2=alpha*dT;

v0=0.1;//Sensitivity (Refer Example 2.12)

Anum=v0*(2+(R1u/R0)+(R0/R1u));

Aden=Vrefr*d2;

A=Anum/Aden;//Overall Gain by using Eq.2.47

```

```
printf("Designed Circuit for Calibration :");

printf("\nR1=%.1f kohms",R1*10^(-3));

printf("\nR2=%.f ohms",R2);

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

printf("\nA=%.1f V/V",A);
```

Scilab code Exa 2.13.b Transducer Bridge Calibration

```
//Example 2.13(a)

clear;

clc;

R0=100;//Data taken from Example 2.11

alpha=0.00392;//Data taken from Example 2.11

Vref=15;

P=0.2*10^(-3);

i=(P/R0)^(0.5)-(0.41*10^(-3));

pV=0.05;

Vrefc=pV*Vref+0.25;

Vrefr=Vref-Vrefc;

R3=2/(2*i);
```

```
//R0+R1+(R2/2)=Vrefr/i;

Rtot=Vrefr/i;

p=0.01;

R2=(2*p*Rtot)+220;//220 ohms are added to be on the
   safe side

R1=Rtot-(R2/2)-R0;//Tolerance 1%

vO=9.97;//Data from Example 2.12

R1u=R1+(R2/2);

dT=1;//obtained from Example 2.12

d2=alpha*dT;

vO=0.1;//Sensitivity (Refer Example 2.12)

Anum=vO*(2+(R1u/R0)+(R0/R1u));

Aden=Vrefr*d2;

A=Anum/Aden;//Overall Gain by using Eq.2.47

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

```
//Example 2.14

clear;

clc;

//2.14(a)

Rs=120;

Vref=15;

imax=20*10^(-3);

Vb=2*Rs*imax;

Vtap=Vb/2;

Vtapch=0.01*Vtap;

v1=Vtap+Vtapch;

v2=Vtap-Vtapch;

v1ch=v1-v2;

i=v1ch/((Rs*Rs)/(Rs+Rs));

R1=(Vtap/i)-630;

R2=1000;

i3=2*imax+(4.8/R2);

R3=(2/i3)+6-0.642857 ;

R4=((Vref-(R3/2)*i3-Vb)/i3)-3;

```

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

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

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

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

//2.14(b)

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

```
//Example 3.1

clear;

clc;

R=10;

C=40*10^(-6);

L=5*10^(-3);

Hsnum=(R/L)*%s;

Hsden=((%s^(2))+(R/L)*%s+(1/(L*C)));

Hs=Hsnum/Hsden;//Transfer Function

h=syslin('c',Hs);
```

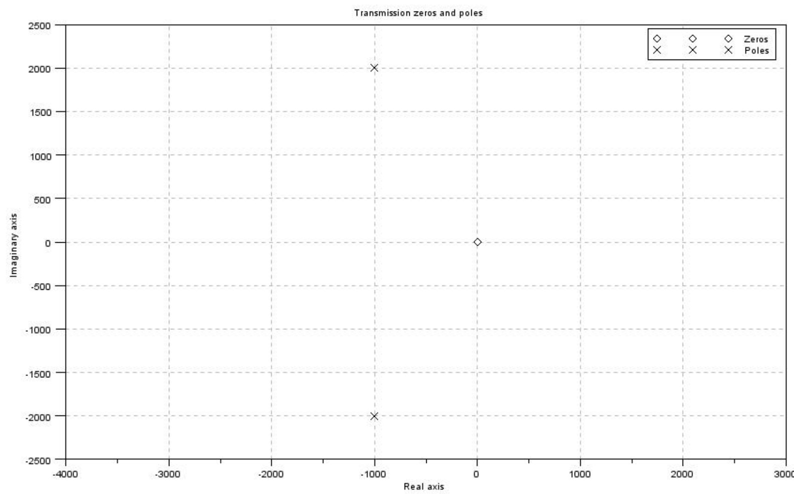

Figure 3.1: Pole Zero Response of Transfer Function

```

plzr(h);

zeroes=roots(Hsnum);

poles=roots(Hsden);
```

Scilab code Exa 3.2 Finding Impulse Response of a given circuit

```
//Example 3.2

clear;

clc;

printf("\nThe problem requires to find Laplace
    Transform which is not possible in scilab. Hence
```

```
    standard procedure");

printf("\nfor finding the Integral Transforms has
    been used")

syms s t;

R=10;

C=40*10^(-6);

L=5*10^(-3);

Hsnum=(R/L)*s;

Hsden=((s^(2))+(R/L)*s+(1/(L*C)));

Hs=Hsnum/Hsden;//Transfer Function

vot=ilaplace(Hs);//Impulse Response of Circuit
```

Scilab code Exa 3.3 Steady State Response of a Circuit

```
//Example 3.3

clear;

clc;

R=10;

C=40*10^(-6);

```

```
L=5*10^(-3);

s=%i*10^3;

Hsnum=(R/L)*s;

Hsden=((s^(2))+(R/L)*s+(1/(L*C)));

Hs=Hsnum/Hsden;//Transfer Function

Hsmag=10*abs(Hs);

Hsphase1=atan(imag(Hs)/real(Hs));

Hsphase=(Hsphase1*(180/%pi))+45;

printf("vO(t)=%.3f",Hsmag);

printf("cos((10^3)t+%.2f",Hsphase);

printf(") V");

//vot=Hsmag*cos(10^3*t+Hsphase);
```

Scilab code Exa 3.4.a Low pass filter with Gain

```
//Example 3.4(a)

clear;

clc;

dcgaindB=20;//Gain in dB

dcgain=10^(20/20);
```

```
f0=10^3;

//We need R2=dcgain*R1;

R1approx=20*10^(3);

R2approx=dcgain*R1approx;

Capprox=1/(2*%pi*f0*R2approx);

n=(Capprox*10^9);

C=Capprox/n;

R2=(R2approx*n)-1154.9431;

R1=R2/dcgain;

printf("Components for achieving the mentioned requirements :");

printf("\nR1=%.1f kohms",R1*10^(-3));

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

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

Scilab code Exa 3.4.b Low pass filter with Gain

```
//Example 3.4(b)

clear;

clc;
```

```
dcgaindB=20; //Gain in dB

dcgain=10^(20/20);

f0=10^3;

//We need R2=dcgain*R1;

R1approx=20*10^(3);

R2approx=dcgain*R1approx;

Capprox=1/(2*%pi*f0*R2approx);

n=(Capprox*10^9);

C=Capprox/n;

R2=R2approx*n;

R1=R2/dcgain;

//Hs=-(R2/R1)*(1/(R2Cs+1))

Hmag=1;

H0=(R2/R1);

f=(((H0/Hmag)^(2)-1)*(f0^2))^(1/2);

s=%i*f;

Hs=-(R2/R1)*(1/(R2*C*s+1));

Hsph=180-(atan(f/f0)*(180/%pi));

printf("The frequency at which gain drops to 0dB=%.3
```

```
    f kHz",f*10^(-3));

printf("\nCorresponding phase=%.2f deg",Hsph);
```

Scilab code Exa 3.5 Designing Wideband Band Pass Filter

```
//Example 3.5

clear;

clc;

GdB=20;

G=10^(20/20);

//->R2/R1=G

R1approx=10*10^3;

R2approx=G*R1approx;

f1=20;

w1=2*%pi*f1;

Capprox1=1/(w1*R1approx);

n=Capprox1/(10^(-6));

C1=Capprox1/n;

R1=(R1approx*n)-87.747155;

R2=R1*G;
```

```
f2=20*10^3;

w2=2*%pi*f2;

C2=1/(R2*w2);

printf("Designed Wideband Band Pass Filter :");

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

printf("\nR2=%.1f kohms",R2*10^(-3));

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

printf("\nC2=%.f pF",C2*10^(12));
```

Scilab code Exa 3.6 Designing Phono Amplifier

```
//Example 3.6

clear;

clc;

GdB=40;

GdBf2=GdB+20;

Gf2=10^(GdBf2/20);

//->((R2+R3)/R1)=Gf2

C2=10*10^(-9);//Assumed Value of C2

```

```
f1=500;

f2=50;

f3=2122;

w1=2*%pi*f1;

w2=2*%pi*f2;

w3=2*%pi*f3;

R2=(1/(w2*C2))-2309.8862;

C3=((1/R2)-(w1*C2))/(w1-w3);

R3=(1/(w3*C3))+(0.94*10^3);

R1=((R2+R3)/Gf2)-4;

C1=(1/(2*%pi*20*R1))+(10*10^(-6));//Here f=20 Hz as
   it is the lower limit of the audio range

printf("Designed RIAA phono Amplifier :");

printf("\nR1=%.f ohms",R1);

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

printf("\nR3=%.1f kohms",R3*10^(-3));

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

printf("\nC2=%.f nF",C2*10^9);

printf("\nC3=%.1f nF",(C3*10^9)-0.1);
```

Scilab code Exa 3.7 Designing a bass or treble control

```
//Example 3.7

clear;

clc;

GdB=20;

fB=30;

fT=10*10^3;

G=10^(GdB/20);

//->((R2+R1)/R1)=G and ((R1+R3+2R5)/R3)=G

R2=100*10^3;//Assume R2 be a 100 kohms pot

R1=R2/(G-1);

R5=R1;//Arbitraly chosen value

R3=((R1+(2*R5))/(G-1))-(0.1*10^3);

//R4>>(R1+R3+2R5)

R4min=R1+R3+2*R5+400;

R4=500*10^(3);//Let R4 be a 500 kohms pot

C1=(1/(2*%pi*R2*fB));

```

```
C2=(1/(2*%pi*R3*fT))+0.9*10^(-9);//0.6 nF is added
    for standardisation

printf("Designed Bass/Trebble Control :");

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

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

printf("\nR3=%.1f kohms",R3*10^(-3));

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

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

printf("\nC1=%.f nF",(C1*10^9)-2.05);

printf("\nC2=%.1f nF",(C2*10^9)-0.22);
```

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

```
//Example 3.8

clear;

clc;

f0=1*10^3;

Q=5;

C=10*10^(-9);//Arbitrarily chosen value

R=1/(2*%pi*f0*C);
```

```
K=3-(1/Q);//DC gain

//->RB/RA=K-1

RA=10*10^3;//Assumed value of RA

RB=((K-1)*RA)-200;

C1=C;

C2=C;

printf("Designed Equal Component Second Order Low
    Pass Filter :");

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

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

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

printf("\nC=%.2f nF",C*10^9);

printf("\n\ndc gain (K)=%.2f V/V",K)
```

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

```
//Example 3.9

clear;

clc;

```

```
//Applying Thevenin's theorem
//Anew=(R1B/(R1A+R1B))Aold and R1A || R1B =R1

AnewdB=0;

Anew=10^AnewdB;

C=10*10^(-9);

Aold=2.8;//Obtained from Example 3.8

RA=10*10^3;//Assumed value of RA

RB=17.8*10^3;

R1=15915.494;//obtained from Example 3.8

R2=R1;

R1A=R1*(Aold/Anew);

R1B=R1/(1-(Anew/Aold));

printf("Designed Second Order Low Pass Filter for 0
    dB dc gain :");

printf("\nR1A=%.2f kohms",R1A*10^(-3));

printf("\nR1B=%.2f kohms",R1B*10^(-3));

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

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

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

printf("\nC=%.2f nF",C*10^9);
```

Scilab code Exa 3.10 Designing a Unity Gain Low Pass Filter

```
//Example 3.10

clear;

clc;

C=1*10^(-9);

Q=2;

n=(4*Q^(2))+4;

C1=n*C;

C2=C;

f0=10*10^(3);

k=(n/(2*(Q^(2))))-1;

m=k+((k^2-1)^(0.5));

k1=(m*n)^(0.5);

R=1/(k1*2*%pi*f0*C);

R2=R;

R1=m*R;

printf("Designed Unity Gain Low Pass Filter :");

```

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

Scilab code Exa 3.11.a Designing Butterworth Low Pass Filter

```
//Example 3.11(a)
clear;
clc;
m=1;//Q is maximised at m=1
n=2;//Order of filter
f0=10*10^(3);
Qnum=(m*n)^(1/2);
Qden=m+1;
Q=Qnum/Qden;
C=1*10^(-9);//Assuming C=1 nF
C2=C;
C1=n*C;
R=1/(Qnum*C*2*%pi*f0);
```

```
R2=R;

R1=m*R;

printf("Designed Second Order Low Pass Butterworth
    Filter :")

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

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

printf("\nC1=%.f nF",C1*10^9);

printf("\nC2=%.f nF",C2*10^9);
```

Scilab code Exa 3.11.b Designing Butterworth Low Pass Filter

```
//Example 3.11(b)

clear;

clc;

m=1;//Q is maximised at m=1

n=2;//Order of filter

f0=10*10^(3);

Qnum=(m*n)^(1/2);

Qden=m+1;

Q=Qnum/Qden;
```

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

C2=C;

C1=n*C;

R=1/(Qnum*C*2*%pi*f0);

R2=R;

R1=m*R;

w=4*%pi*10^4;

f=2*10^4;

Hw=1/(1-(w^(2)*R1*R2*C1*C2)+%i*w*((R1*C2)+(R2*C2)));

Vom=10*abs(Hw);

an=atan(imag(Hw)/real(Hw));

theta=180-(an*(180/%pi));

theta0=theta-90;

printf("vo(t)=%.3f cos(4*pi*(10^4)*t+",Vom);

printf("%.2f) V",theta0);
```

Scilab code Exa 3.12 Designing High Pass KRC Filters

```
//Example 3.12

```

```
clear;

clc;

//To minimize the component count, choose the unity
   gain option, for which RA=infinity and RB=0.

C=0.1*10^(-6);

C1=C;

C2=C;

n=C1/C2;

Q=1.5;

f0=200;

m=n/(((n+1)*Q)^2);

R=1/(2*%pi*f0*((m*n)^(1/2))*C);

R2=R;

R1=m*R;

printf("Designed High Pass KRC Filter :");

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

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

printf("\nC1=%.1f uF",C1*10^6);

printf("\nC2=%.1f uF",C2*10^6);
```

Scilab code Exa 3.13.a Designing KRC Bandpass Filter

```
//Example 3.13(a)

clear;

clc;

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

C1=C;

C2=C;

f0=1*10^3;

BW=100;

R=(2^(1/2))/(2*%pi*f0*C);

R1=R;

R2=R;

R3=R;

Q=f0/BW;

K=4-((2^(1/2))/Q);

RA=10*10^3;

RB=(K-1)*RA;

```

```
RG=K/(4-K);

printf("Designed KRC Second Order Band Pass filter")
    ;

printf("\nR1=R2=R3=%.1f kohms",R*10^(-3));

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

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

printf("\nC1=C2=%.2f nF",C*10^9);

printf("\n\nResonance Gain=%.2f V/V",RG);
```

Scilab code Exa 3.13.b Designing KRC Bandpass Filter

```
//Example 3.13(b)

clear;

clc;

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

C1=C;

C2=C;

f0=1*10^3;

BW=100;

R=(2^(1/2))/(2*%pi*f0*C);

```

```
R1=R;

R2=R;

R3=R;

Q=f0/BW;

K=4-((2^(1/2))/Q);

RA=10*10^3;

RB=(K-1)*RA;

RG=K/(4-K);

RG1dB=20;

RG1=10^(RG1dB/20);

R1A=(R1*(RG/RG1))+488.81355;

R1B=(R1/(1-(RG1/RG)))+169.90124;

printf("Designed KRC Second Order Band Pass filter
    with 20 dB Resonance Gain");

printf("\nR1A=%.2f kohms",R1A*10^(-3));

printf("\nR1B=%.2f kohms",R1B*10^(-3));

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

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

printf("\nC1=C2=%.2f nF",C*10^9);
```

Scilab code Exa 3.14 Designing Band Reject KRC Filter

```
//Example 3.14

clear;

clc;

C=100*10^(-9);//Assuming C=100 nF

C1=C;

C2=2*C;

f0=60;

BW=5;

R=1/(2*%pi*f0*C);

R1=R;

R2=R/2;

Q=f0/BW;

K=(4-(1/Q))/2;

RA=10*10^3;

RB=(K-1)*RA;

printf("Designed Second Order Notch Filter :")

```

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

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

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

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

printf("\nC1=%.2f nF",C1*10^9);

printf("\nC2=%.2f nF",C2*10^9);

printf("\n\nLow and High Frequency Gain=%.2f V/V",K)
    ;
```

Scilab code Exa 3.15 Designing Multiple Feedback Band Pass Filter

```
//Example 3.15

clear;

clc;

C=10*10^(-9);

C1=C;

C2=C;

f0=1*10^3;

Q=10;

H0dB=20;

```

```
H0=10^(H0dB/20);

R2=(2*Q)/(2*%pi*f0*C);

R1A=Q/(H0*2*%pi*f0*C);

R1B=R1A/((2*Q^2/H0)-1);

printf("Designed Multiple Feedback Band Pass Filter
    :")

printf("\nR1A=%.2f kohms",R1A*10^(-3));

printf("\nR1B=%.2f ohms",R1B);

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

printf("\nC1=%.2f nF",C1*10^(9));

printf("\nC2=%.2f nF",C2*10^(9));
```

Scilab code Exa 3.16 Designing Multiple Feedback Low Pass Filter

```
//Example 3.16

clear;

clc;

H0=2;

f0=10*10^3;

Q=4;

```

```
nmin=4*(Q^2)*(1+H0);

n=nmin+8;//Assuming n=nmin+8

C2=1*10^(-9);//Assuming C2

C1=C2*n;

R3num1=nmin/n;

R3num2=(1-R3num1)^(1/2);

R3num=1+R3num2;

R3den=2*2*%pi*f0*Q*C2;

R3=R3num/R3den;

R1=R3/H0;

R2=1/(((2*%pi*f0)^2)*R3*C1*C2);

printf("Designed Multiple Feedback Low Pass Filter :
    ")

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

printf("\nR2=%.2f ohms",R2);//Answer in textbook is
    wrong

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

printf("\nC1=%.2f uF",C1*10^(6));

printf("\nC2=%.2f nF",C2*10^(9));
```

Scilab code Exa 3.17 Designing Multiple Feedback Notch Filters

```
//Example 3.17

clear;

clc;

f0=1*10^3;

Q=10;

HondB=0;

Hon=10^(HondB/20);

C=10*10^(-9);//Assuming C=10 nF

C1=C;

C2=C;

R3=10*10^3;

R4=R3/Hon;

R5=Hon*R4;

R2=(2*Q)/(2*%pi*f0*C);

R1A=Q/(Hon*2*%pi*f0*C);

R1B=R1A/((2*Q^2/Hon)-1);

```

```
printf("Designed Multiple Feedback Notch Filter :");

printf("\nR1A=%.2f kohms",R1A*10^(-3));

printf("\nR1B=%.2f ohms",R1B);

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

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

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

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

printf("\nC1=C2=%.2f nF",C*10^9);
```

Scilab code Exa 3.18 Designing State Variable Filter for Bandpass Response

```
//Example 3.18

clear;

clc;

C=10*10^(-9);//Assuming C=10 nF

C1=C;

C2=C;

f0=1*10^3;

BW=10;

```

```
R=(1/(2*%pi*f0*C))-(0.12*10^3);

Q=f0/BW;

R1=1*10^3;//Assuming R1=1 kohms

R2=((3*Q)-1)*R1;

R3=R;

R4=R;

R5=R;

Hobp=Q;

printf("Designed State-Variable Filter for Bandpass
    Response :");

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

printf("\nR2=%.2f kohms",R2*10^(-3));//Answer in
    textbook is wrong

printf("\nR3=R4=R5=%.2f kohms",R*10^(-3));

printf("\nC1=C2=%.2f nF",C*10^9);

printf("\n\nResonance Gain=%.2f V/V",Hobp);
```

Scilab code Exa 3.19 Designing a Biquad Filter

```
//Example 3.19

clear;
```

```
clc;

C=1*10^(-9);//Assuming C=1 nF

C1=C;

C2=C;

f0=8*10^3;

BW=200;

R=1/(2*%pi*f0*C);

R4=R;

R5=R;

Q=f0/BW;

R2=Q*R;

HobpdB=20;

Hobp=10^(HobpdB/20);

R1=(R2/Hobp)- 877.47155;

R3=R2;

Holp=R/R1;

HolpdB=20*log10(Holp);

printf("Designed Biquad Filter :");

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

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

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

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

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

printf("\nC1=C2=%.2f nF",C*10^9);

printf("\n\nResonance Gain (Holp)=%.2f dB",HolpdB);
```

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

```
//Example 3.20

clear;

clc;

f0=1*10^3;

fz=2*10^3;

Q=10;

C=10*10^(-9);//Assume C=10 nF

R=(1/(2*%pi*f0*C))-120;

w0=2*%pi*f0;

wz=2*%pi*fz;
```

```
R1=Q*R;

R2=100*10^3;//Assumption

R3=R2;

R4num=R2*(w0^2);

R4den=Q*abs((w0^2)-(wz^2));

R4=R4num/R4den;

R5=R2*((w0/wz)^2);//as fz>f0

Hohp=R5/R2;

HohpdB=20*log10(Hohp);

printf("\nDesigned Biquad Filter for a low pass
   notch response :");

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

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

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

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

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

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

printf("\nC=%.2f nF",C*10^9);

printf("\n\nHigh Frequency Gain (Hohp)=%.2f dB",
   HohpdB);
```

Scilab code Exa 3.21.a KRC Filter Sensitivities

```
//Example 3.21(a)

clear;

clc;

//From the result of Example 3.8 :

RA=10*10^3;

RB=18*10^3;

f0=1*10^3;

Q=5;

C=10*10^(-9);

C1=C;

C2=C;

R=15915.494;

K=2.8;

SR=(Q-(1/2));

SC=((2*Q)-(1/2));

SK=(3*Q)-1;

```

```
SRA=1-(2*Q);

printf("Sensitivities for Example 3.8 :");

printf("\nSR=%.2f percent",SR);

printf("\nSC=%.2f percent",SC);

printf("\nSRA=%.2f percent",SRA);

printf("\nSK=%.2f percent",SK);
```

Scilab code Exa 3.21.b KRC Filter Sensitivities

```
//Example 3.21(b)

clear;

clc;

R1=5758.2799;

R2=2199.4672;

C1=2.000D-08;

C2=1.000D-09;

SC1=1/2;

r=R1/R2;

SR1=(1-r)/(2*(1+r));

printf("Sensitivities for Example 3.10 :");
```

```

printf("\nSR=%.2f percent",SR1);

printf("\nSC=%.2f percent",SC1);
```

Chapter 4

Active Filters Part II

Scilab code Exa 4.1 Butterworth Filter Approximations

```
//Example 4.1

clear;

clc;

fc=1*10^3;

fs=2*10^3;

AmaxdB=1;

AmindB=40;

e=((10^(AmaxdB/20))^(2)-1)^(1/2);

n1=((10^(AmindB/10))-1)/(e^2);

n=log(n1)/(2*log(fs/fc))+0.4;//0.4 is added in order
    to obtain a integer

```

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

Scilab code Exa 4.2 Cascade Designing of Chebyshev Low Pass Filter

```
//Example 4.2(a)

clear;

clc;

n=6;

fc=13*10^3;

//For a 1dB ripple Chebyshev low pass filter with n
   =6 requires 3 second order stages with :
//f01=0.995*fc, Q1=8
//f02=0.747*fc, Q2=2.20
//f03=0.353*fc, Q3=0.761

f03=0.995*fc;

Q1=0.761;

f02=0.747*fc;

Q2=2.20;

f01=0.353*fc;

Q3=8.00;

n1=(4*Q1^(2))+0.0016978;

C1=2.2*10^(-9);
```

```
C11=n1*C1;

C21=C1;

k1=(n1/(2*(Q1^(2))))-1;

m1=k1+(((k1^2)-1)^(0.5));

k11=(m1*n1)^(0.5);

R1=1/(k11*2*%pi*f01*C1);

R11=m1*R1;

R21=R1;

n2=(4*Q2^(2))+0.2478431;

C2=510*10^(-12);

C12=n2*C2;

C22=C2;

k2=(n2/(2*(Q2^(2))))-1;

m2=k2+(((k2^2)-1)^(0.5));

k12=(m2*n2)^(0.5);

R2=1/(k12*2*%pi*f02*C2);

R12=m2*R2;

R22=R2;

n3=(4*Q3^(2))+25.818182;
```

```
C3=220*10^(-12);

C13=n3*C3;

C23=C3;

k3=(n3/(2*(Q3^(2))))-1;

m3=k3+(((k3^2)-1)^(0.5));

k13=(m3*n3)^(0.5);

R3=1/(k13*2*%pi*f03*C3);

R13=m3*R3;

R23=R3;

printf("Designed Chebyshev Filter :");

printf("\nSection I :")

printf("\nR1=%.2f kohms",R11*10^(-3));

printf("\nR2=%.2f kohms",R21*10^(-3));

printf("\nC1=%.2f nF",C11*10^9);

printf("\nC2=%.2f nF",C21*10^9);

printf("\n\nSection II :")

printf("\nR1=%.2f kohms",R12*10^(-3));

printf("\nR2=%.2f kohms",R22*10^(-3));

printf("\nC1=%.2f nF",C12*10^9);
```

```
printf("\nC2=%.2f pF",C22*10^12);

printf("\n\nSection III :")

printf("\nR1=%.2f kohms",R13*10^(-3));

printf("\nR2=%.2f kohms",R23*10^(-3));

printf("\nC1=%.2f nF",C13*10^9);

printf("\nC2=%.2f pF",C23*10^12);
```

Scilab code Exa 4.3 Cascade Designing of Cauer Low Pass Filter

```
//Example 4.3

clear;

clc;

fc=1*10^(3);

fs=1.3*10^(3);

AmaxdB=0.1;

Amax=10^(AmaxdB/20);

AmindB=40;

Amin=10^(AmindB/20);

f01=648.8;

```

```
fz1=4130.2;

Q1=0.625;

f02=916.5;

fz2=1664.3;

Q2=1.789;

f03=1041.3;

fz3=1329;

Q3=7.880;

C1=2.2*10^(-9);

R1=1/(2*%pi*f01*C1);

w01=2*%pi*f01;

wz1=2*%pi*fz1;

R11=Q1*R1;

R21=100*10^3;//Assumption

R41num=R21*(w01^2);

R41den=Q1*abs((w01^2)-(wz1^2));

R41=R41num/R41den;

R51=R21*((w01/wz1)^2);//as fz1>f01

R31=R21;

```

```
C2=2.2*10^(-9);

R2=1/(2*%pi*f02*C2);

w02=2*%pi*f02;

wz2=2*%pi*fz2;

R12=Q2*R2;

R22=100*10^3;//Assumption

R42num=R22*(w02^2);

R42den=Q2*abs((w02^2)-(wz2^2));

R42=R42num/R42den;

R52=R22*((w02/wz2)^2);//as fz2>f02

R32=R22;

C3=2.2*10^(-9);

R3=1/(2*%pi*f03*C3);

w03=2*%pi*f03;

wz3=2*%pi*fz3;

R13=Q3*R3;

R23=100*10^3;//Assumption

R43num=R23*(w03^2);

R43den=Q3*abs((w03^2)-(wz3^2));
```

```
R43=R43num/R43den;
R53=R23*((w03/wz3)^2);//as fz3>f03
R33=R23;
printf("Designed Cauer Low Pass Filter :");
printf("\nSection I :");
printf("\nR=%.f kohms",(R1*10^(-3))-1.5);
printf("\nR1=%.2f kohms",R11*10^(-3));
printf("\nR2=%.f kohms",R21*10^(-3));
printf("\nR3=%.f kohms",R31*10^(-3));
printf("\nR4=%.2f kohms",R41*10^(-3));
printf("\nR5=%.2f kohms",R51*10^(-3));
printf("\nC=%.2f nF",C1*10^9);
printf("\n\nSection II :");
printf("\nR=%.2f kohms",R2*10^(-3));
printf("\nR1=%.f kohms",(R12*10^(-3))-1.21);
printf("\nR2=%.f kohms",R22*10^(-3));
printf("\nR3=%.f kohms",R32*10^(-3));
printf("\nR4=%.1f kohms",R42*10^(-3));
printf("\nR5=%.2f kohms",R52*10^(-3));
```

```
printf("\nC=%.2f nF",C2*10^9);

printf("\n\nSection III :");

printf("\nR=%.2f kohms",(R3*10^(-3))+0.33);

printf("\nR1=%.2f kohms",(R13*10^(-3))+1.54579);

printf("\nR2=%.f kohms",R23*10^(-3));

printf("\nR3=%.f kohms",R33*10^(-3));

printf("\nR4=%.d kohms",R43*10^(-3));

printf("\nR5=%.2f kohms",(R53*10^(-3))+0.51);

printf("\nC=%.2f nF",C3*10^9);
```

Scilab code Exa 4.4 Designing a Chebyshev High Pass Filter

```
//Example 4.4

clear;

clc;

fc=100;

f01=fc/1.300;

Q1=1.341;

f02=fc/0.969;

H0dB=20;
```

```
H0=10^(H0dB/20);

C=100*10^(-9);

C1=C;

C2=C;

n=C1/C2;

m=n/(((n+1)*Q1)^2);

R=1/(2*%pi*f01*((m*n)^(1/2))*C);

R21=R;

R11=m*R;

//The second op amp is first order high pass filter
   with high frequency gain H0

Rf=154*10^3;//Assumption

R12=Rf/H0;

printf("Designed Chebyshev High Pass Filter :");

printf("\nSecond Order High Pass Section :");

printf("\nR1=%.2f kohms",R11*10^(-3));

printf("\nR2=%.2f kohms",(R21-590.96246)*10^(-3));

printf("\nC=%.2f nF",C*10^9);

printf("\n\nFirst Order High Pass Section :");
```

```
printf("\nR1=%.2f kohms",R12*10^(-3));

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

printf("\nC=%.2f nF",C*10^9);
```

Scilab code Exa 4.5 Cascade Designing of Butterworth Band Pass Filter

```
//Example 4.5

clear;

clc;

f0=1*10^3;

f03=957.6;

Q3=20.02;

f02=1044.3;

Q2=20.02;

f01=1000;

Q1=10;

H0bp3=2;

H0bp2=2;

H0bp1=1;

C1=10*10^(-9);
```

```
C11=C1;
C21=C1;
R21=(2*Q1)/(2*%pi*f01*C1);
R11A=Q1/(H0bp1*2*%pi*f01*C1);
R11B=R11A/((2*Q1^2/H0bp1)-1);
R1pot=200;
C2=10*10^(-9);
C12=C2;
C22=C2;
R22=(2*Q2)/(2*%pi*f02*C2);
R12A=Q2/(H0bp2*2*%pi*f02*C2);
R12B=R12A/((2*Q2^2/H0bp2)-1);
R2pot=100;
C3=10*10^(-9);
C13=C3;
C23=C3;
R23=(2*Q3)/(2*%pi*f03*C3);
R13A=Q3/(H0bp3*2*%pi*f03*C3);
R13B=R13A/((2*Q3^2/H0bp3)-1);
```

```
R3pot=100;

printf("Designed Butterworth Band Pass Filter :");

printf("\nSection I :");

printf("\nR1A=%.f kohms",(R11A*10^(-3))-1.15);

printf("\nR1B=%.f ohms",R11B-101.77);

printf("\nR2=%.f kohms",(R21*10^(-3))-2.31);

printf("\nC1=%.f nF",C11*10^(9));

printf("\nC2=%.f nF",C21*10^(9));

printf("\nPotentiometer Resistance (Rpot)=%.f ohms",
   R1pot);

printf("\n\nSection II :");

printf("\nR1A=%.f kohms",(R12A*10^(-3))+1.44);

printf("\nR1B=%.f ohms",R12B-49.58);

printf("\nR2=%.f kohms",(R22*10^(-3))-6.22);

printf("\nC1=%.f nF",C12*10^(9));

printf("\nC2=%.f nF",C22*10^(9));

printf("\nPotentiometer Resistance (Rpot)=%.f ohms",
   R2pot);

printf("\n\nSection III :");

printf("\nR1A=%.f kohms",(R13A*10^(-3))-1.37);
```

```
printf("\nR1B=%.f ohms",R13B-51.13);

printf("\nR2=%.d kohms",R23*10^(-3));

printf("\nC1=%.f nF",C13*10^(9));

printf("\nC2=%.f nF",C23*10^(9));

printf("\nPotentiometer Resistance (Rpot)=%.f ohms",
   R3pot);
```

Scilab code Exa 4.6 Cascade Designing of Elliptic Band Pass Filter

```
//Example 4.6

clear;

clc;

f01=907.14;

fz1=754.36;

Q1=21.97;

f02=1102.36;

fz2=1325.6;

Q2=21.97

f03=1000;

Q3=9.587;
```

```
//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.

//Ist Stage (high pass notch biquad stage)

C=10*10^(-9);

w01=2*%pi*f01;

wz1=2*%pi*fz1;

R1=1/(2*%pi*f01*C);

R11=Q1*R1;

R21=100*10^3;

R31=100*10^3;

R41num=R21*(w01^2);

R41den=Q1*abs((w01^2)-(wz1^2));

R41=R41num/R41den;

R51=R21;//as fz1<f01

Rex1=14.7*10^3;

Rex1pot=5*10^3;

//IInd Stage (low pass notch biquad stage)

w02=2*%pi*f02;

```

```
wz2=2*%pi*fz2;

R2=1/(2*%pi*f02*C);

R12=Q1*R2;

R22=100*10^3;

R32=100*10^3;

R42num=R22*(w02^2);

R42den=Q2*abs((w02^2)-(wz2^2));

R42=R42num/R42den;

R52=R22*((w02/wz2)^2);//as fz2>f02

Rex2=11.8*10^3;

Rex2pot=5*10^3;

//IIIrd Stage (Multiple feedback band pass stage)

H03=1.23;

R23=(2*Q3)/(2*%pi*f03*C);

R13A=Q3/(H03*2*%pi*f03*C);

R13B=R13A/((2*Q3^2/H03)-1);

Rpot3=200;

printf("Designed Elliptic Band Pass Filter :");

printf("\nStage I (High pass notch biquad stage)");
```

```
printf("\nR=%.1f kohms",(R1*10^(-3))-0.14);

printf("\nR1=%.f kohms",(R11*10^(-3))-2.46);

printf("\nR2=%.f kohms",R21*10^(-3));

printf("\nR3=%.f kohms",R31*10^(-3));

printf("\nR4=%.1f kohms",R41*10^(-3));

printf("\nR5=%.f kohms",R51*10^(-3));

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

printf("\nRex=%.2f kohms",Rex1*10^(-3));

printf("\nRexpot=%.f kohms",Rex1pot*10^(-3));

printf("\n\nStage II (low pass notch biquad stage)")
    ;

printf("\nR=%.2f kohms",(R2*10^(-3))-0.14);

printf("\nR1=%.f kohms",(R12*10^(-3))-1.20);

printf("\nR2=%.f kohms",R22*10^(-3));

printf("\nR3=%.f kohms",R32*10^(-3));

printf("\nR4=%.2f kohms",R42*10^(-3));

printf("\nR5=%.1f kohms",R52*10^(-3));

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

printf("\nRex=%.2f kohms",Rex2*10^(-3));

printf("\nRexpot=%.f kohms",Rex2pot*10^(-3));
```

```
printf("\n\nStage III (Multiple feedback band pass
    stage)");

printf("\nR2=%.d kohms",(R23*10^(-3))+4);

printf("\nR1A=%.f kohms",R13A*10^(-3));

printf("\nR1B=%.f ohms",R13B-103.65);

printf("\nRpot=%.f ohms",Rpot3);

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

Scilab code Exa 4.7 Cascade Designing of Chebyshev Band Reject Filter

```
//Example 4.7

clear;

clc;

f01=3460.05;

fz1=3600;

Q1=31.4;

f02=3745;

fz2=3600;

Q2=31.4;

f03=3600;
```

```
fz3=3600;

Q3=8.72;

//The answer of the Example 4.7 is not given in the
    textbook

//The filter is designed using three biquad sections
    , namely, a high pass notch, followed by a low
    pass notch, followed by a symmetric notch.

//Ist(High pass notch Biquad section)

C=10*10^(-9);

w01=2*%pi*f01;

wz1=2*%pi*fz1;

R1=1/(2*%pi*f01*C);

R11=Q1*R1;

R21=100*10^3;

R31=100*10^3;

R41num=R21*(w01^2);

R41den=Q1*abs((w01^2)-(wz1^2));

R41=R41num/R41den;

R51=R21;//as fz1<f01

Rex1=14.7*10^3;

```

```
Rex1pot=5*10^3;

//IInd Stage (low pass notch biquad stage)

w02=2*%pi*f02;

wz2=2*%pi*fz2;

R2=1/(2*%pi*f02*C);

R12=Q1*R2;

R22=100*10^3;

R32=100*10^3;

R42num=R22*(w02^2);

R42den=Q2*abs((w02^2)-(wz2^2));

R42=R42num/R42den;

R52=R22*((w02/wz2)^2);//as fz2>f02

Rex2=11.8*10^3;

Rex2pot=5*10^3;

//IIIrd Stage (Symmetric Notch Section)

L13=0.84304;

C13=0.62201;

CC130=C13/(2*%pi*f03);

CL130=L13/(2*%pi*f03);

```

```
C03=10*10^(-6);//Assumption

CC13=CC130*C03;

CL13=CL130*C03;

printf("Designed Chebyshev Band Reject Filter :");

printf("\nStage I(High pass notch Biquad section)");

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

printf("\nR1=%.2f kohms",R11*10^(-3));

printf("\nR2=%.2f kohms",R21*10^(-3));

printf("\nR3=%.2f kohms",R31*10^(-3));

printf("\nR4=%.2f kohms",R41*10^(-3));

printf("\nR5=%.2f kohms",R51*10^(-3));

printf("\nC=%.2f nF",C*10^9);

printf("\n\nStage II(Low pass notch Biquad section)"
   );

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

printf("\nR1=%.2f kohms",R12*10^(-3));

printf("\nR2=%.2f kohms",R22*10^(-3));

printf("\nR3=%.2f kohms",R32*10^(-3));

printf("\nR4=%.2f kohms",R42*10^(-3));

printf("\nR5=%.2f kohms",R52*10^(-3));
```

```
printf("\nC=%.2f nF",C*10^9);

printf("\n\nStage III(Symmetric Notch Section)");

printf("\nC0=%.2f uF",C03*10^(6));

printf("\nCC1=%.2f pF",CC13*10^(12));

printf("\nCL1=%.2f pF",CL13*10^(12));
```

Scilab code Exa 4.8 Designing a Dual Amplifier Band Pass Filter

```
//Example 4.8

clear;

clc;

f0=2*10^3;

Q=25;

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

w0=2*%pi*f0;

L=1/((w0^2)*C);

R=Q/((C/L)^(1/2));

//Specifying components of GIC

C2=C;

```

```
R1=(L/C2)^(1/2);

R3=R1;

R4=R1;

R5=R1;

printf("Designed Dual Amplifier Band Pass Filter :")
    ;

printf("\nC=%.2f nF",C*10^9);

printf("\nL=%.2f H",L);

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

printf("\n\nComponents of General Impedance
    Converter :");

printf("\nC2=%.2f nF",C2*10^9);

printf("\nR1=R3=R4=R5=%.2f kohms",R1*10^(-3));
```

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

```
//Example 4.9

clear;

clc;

f0=1*10^3;

```

```
Q=5;
w0=2*%pi*f0;
Rinv=100*10^(-9);
D=Rinv/(Q*w0);
C=D;
L=1/((w0^2)*C);
//Specifying Components for GIC
C1=10*10^(-9);
C2=C1;
C5=C1;
R2=D/(C2*C5);
R3=R2;
R4=R2;
printf("Designed General Impedance Converter Low Pass Filter :");
printf("\nR0=1 Mohms");
printf("\nCapacitance denoted by R inverse=0.1 uF")
printf("\nResistance associated with C=%.2f pohms",C*10^12);
printf("\nResistance associated with L=%.2f kohms",(L*10^(-3))+0.1);
```

```
printf("\nC1=C2=C5=%.f nF",C1*10^9);

printf("\nR2=R3=R4=%.2f kohms",(R2*10^(-3))-0.23);
```

Scilab code Exa 4.10 Direct Designing of Low Pass Filter

```
//Example 4.10

clear;

clc;

f=15*10^3;

w=2*%pi*f;

L1old=1.367;

L2old=0.1449;

L3old=1.785;

L4old=0.7231;

L5old=1.579;

L6old=0.5055;

L7old=1.096;

Rold=1;

C=1*10^(-9);

```

```
kz=Rold/C;

C2old=1.207;

C4old=0.8560;

C6old=0.9143;

R1new=(L1old*kz)/w;

R2new=(L2old*kz)/w;

R3new=(L3old*kz)/w;

R4new=(L4old*kz)/w;

R5new=(L5old*kz)/w;

R6new=(L6old*kz)/w;

R7new=(L7old*kz)/w;

D2new=(1/(kz*w))*C2old;

D4new=(1/(kz*w))*C4old;

D6new=(1/(kz*w))*C6old;

//Finding the elements in FNDR

R4=10*10^3;

R5=R4;

R21=D2new/(C^2);

R22=D4new/(C^2);

```

```
R23=D6new/(C^2);

printf("Designed Low Pass Filter :");

printf("\nR1new=%.2f kohms",(R1new*10^(-3))-0.2);

printf("\nR2new=%.2f kohms",R2new*10^(-3));

printf("\nR3new=%.2f kohms",(R3new*10^(-3))-0.24);

printf("\nR4new=%.2f kohms",R4new*10^(-3));

printf("\nR5new=%.2f kohms",R5new*10^(-3));

printf("\nR6new=%.2f kohms",R6new*10^(-3));

printf("\nR7new=%.2f kohms",(R7new*10^(-3))-0.13);

printf("\nD2new=");

disp(D2new);

printf("\nD4new=");

disp(D4new);

printf("\nD6new=");

disp(D6new);

printf("\nC=%.2f nF",C*10^9);

printf("\nR4=R5=%.2f kohms",R4*10^(-3));

printf("\nR21=%.2f kohms",R21*10^(-3));

printf("\nR22=%.2f kohms",R22*10^(-3));
```

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

Scilab code Exa 4.11 Direct Designing of High Pass Filter

```
//Example 4.11

clear;

clc;

Rnew=100*10^3;

fc=300;

wc=2*%pi*fc;

L1old=1.02789;

L2old=0.15134;

L3old=1.63179;

L4old=0.44083;

L5old=0.81549;

Rold=1;

C2old=1.21517;

C4old=0.93525;

kz=Rnew*Rold;

C1new=1/(kz*wc*L1old);
```

```
C2new=1/(kz*wc*L2old);
C3new=1/(kz*wc*L3old);
C4new=1/(kz*wc*L4old);
C5new=1/(kz*wc*L5old);
L2new=kz/(wc*C2old);
L4new=kz/(wc*C4old);
//Finding the Elements of GIC
C=10*10^(-9);
R1=(L2new/C)^(1/2);
R3=R1;
R4=R1;
R5=R1;
R2=(L4new/C)^(1/2);
R6=R2;
printf("Designed High Pass Filter :");
printf("\nRnew=%.f kohms",Rnew*10^(-3));
printf("\nC1new=%.2f nF",C1new*10^9);
printf("\nC2new=%.2f nF",C2new*10^9);
printf("\nC3new=%.2f nF",C3new*10^9);
```

```
printf("\nC4new=%.2f nF",C4new*10^9);

printf("\nC5new=%.2f nF",C5new*10^9);

printf("\nL2new=%.2f H",L2new);

printf("\nL4new=%.2f H",L4new);

printf("\n\nThe elements for GIC :");

printf("\nR1=R3=R4=R5=%.2f kohms",R1*10^(-3));

printf("\nR2=R6=%.2f kohms",R2*10^(-3));
```

Scilab code Exa 4.12 Designing a Switched Capacitor Biquad Filter

```
//Example 4.12

clear;

clc;

fck=100*10^3;

f0=1*10^3;

Ctotmax=100*10^(-12);

C1=1*10^(-12);//Assumed

C2=C1*(fck/(2*%pi*f0));

Q=0.707;

```

```
C3=C1*(1/Q);

printf("Designed Switched Capacitor Biquad Filter :"
    );

printf("\nC1=%.2f pF",C1*10^12);

printf("\nC2=%.2f pF",C2*10^12);

printf("\nC3=%.2f pF",C3*10^12);
```

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

```
//Example 4.13

clear;

clc;

C1=0.618;

C5=C1;

C3=2.00;

L2=1.618;

L4=L2;

fc=1*10^3;

wc=2*%pi*fc;

fck=100*10^3;
```

```
C0=1*10^(-12);

CC1=(C1/wc)*fck*C0;

CL2=(L2/wc)*fck*C0;

CC5=CC1;

CL4=CL2;

CC3=(C3/wc)*fck*C0;

CRi=C0;

CRo=C0;

printf("Designed Switched Capacitor Low Pass Filter
   for Butterworth Response :");

printf("\nCRi=CRo=C0=%.2f pF",C0*10^12);

printf("\nCC1=CC5=%.2f pF",CC1*10^12);

printf("\nCL2=CL4=%.2f pF",CL2*10^12);

printf("\nCC3=%.2f pF",CC3*10^12);
```

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

```
//Example 4.14

clear;

```

```
clc;

f0=1*10^3;

BW=600;

fck=100*10^3;

C1=0.84304;

L2=0.62201;

BWnorm=BW/f0;

C1norm=C1/BWnorm;

L1norm=BWnorm/C1;

L2norm=L2/BWnorm;

C2norm=BWnorm/L2;

Rs=1;

Ri=Rs;

Ro=Rs;

C0=1*10^(-12);

CRi=C0;

CRo=C0;

CC1=((fck*C1norm)/(2*%pi*f0))*C0;

CL1=((fck*L1norm)/(2*%pi*f0))*C0;

```

```
CC2=((fck*C2norm)/(2*%pi*f0))*C0;

CL2=((fck*L2norm)/(2*%pi*f0))*C0;

printf("Designed Switched Capacitor Band Pass Filter :");

printf("\nRi=R0=Rs=%.2f ohms",Rs);

printf("\nCRi=CRo=C0=%.2f pF",C0*10^12);

printf("\nCC1=%.2f pF",CC1*10^12/C1norm);

printf("\nC1=%.2f pF",CC1*10^12)

printf("\nCL1=%.2f pF",CL1*10^12);

printf("\nCC2=%.2f pF",(CC2*10^12)-0.54);

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

```
//Example 5.1(a)

clear;

clc;

R1=22*10^3;

R2=2.2*10^6;

IB=80*10^(-9);

IOS=20*10^(-9);

Rp=0;

dcgain=(1+(R2/R1));

R=(R1*R2)/(R1+R2);

Ip=((2*IB)+IOS)/2;
```

```

In=((2*IB)-IOS)/2;

Eo=dcgain*((R*IB));

printf("Eo=(+-)%.2f mV",(Eo*10^3)-1);
```

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

```
//Example 5.1(b)

clear;

clc;

R1=22*10^3;

R2=2.2*10^6;

IB=80*10^(-9);

IOS=20*10^(-9);

Rp=(R1*R2)/(R1+R2);

dcgain=(1+(R2/R1));

R=(R1*R2)/(R1+R2);

Ip=((2*IB)+IOS)/2;

In=((2*IB)-IOS)/2;

Eo=dcgain*((R*In)-(Rp*Ip));

```

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

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

```
//Example 5.1(c)

clear;

clc;

R1=22*10^2;

R2=2.2*10^5;

IB=80*10^(-9);

IOS=20*10^(-9);

Rp=(R1*R2)/(R1+R2);

dcgain=(1+(R2/R1));

R=(R1*R2)/(R1+R2);

Ip=((2*IB)+IOS)/2;

In=((2*IB)-IOS)/2;

Eo=dcgain*((R*In)-(Rp*Ip));

printf("Eo=(+-)%.2f mV",-Eo*10^3);
```

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

```
//Example 5.1(d)

clear;

clc;

R1=22*10^2;

R2=2.2*10^5;

IB=80*10^(-9);

IOS=3*10^(-9);

Rp=(R1*R2)/(R1+R2);

dcgain=(1+(R2/R1));

R=(R1*R2)/(R1+R2);

Ip=((2*IB)+IOS)/2;

In=((2*IB)-IOS)/2;

Eo=dcgain*((R*In)-(Rp*Ip));

printf("Eo=(+-)%.1f mV",-Eo*10^3);
```

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

```
//Example 5.2(a)

clear;

clc;
```

```
R=100*10^3;

C=1*10^(-9);

vo0=0;

IB=80*10^(-9);

IOS=20*10^(-9);

Vsat=13;

Rp=0;

Ip=((2*IB)+IOS)/2;

In=((2*IB)-IOS)/2;

vo1=(R*IB)/(R*C);

t=Vsat/vo1;

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

```
//Example 5.2(b)

clear;

clc;

R=100*10^3;
```

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

vo0=0;

IB=80*10^(-9);

IOS=20*10^(-9);

Vsat=13;

Rp=R;

Ip=((2*IB)+IOS)/2;

In=((2*IB)-IOS)/2;

vo1=(R*IB)/(R*C);

t1=Vsat/vo1;

t=t1*(IB/IOS);

printf("Time taken by the op amp to enter saturation
    (t)=%.2f sec",t);
```

Scilab code Exa 5.3 Input Bias Current Drift

```
//Example 5.3

clear;

clc;

T0=25;
```

```
IBT0=1*10^(-12);

T=100;

IBT=IBT0*2^((T-T0)/10);

printf("IB(100degC)=%.2f nA",IBT*10^9);
```

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

```
//Example 5.4(a)

clear;

clc;

R1=10*10^3;

R2=100*10^3;

CMRRdB=90;

CMRRrec=10^(-(CMRRdB/20));//Reciprocal of CMRR

delvi=10;

delvp=(R2/(R1+R2))*delvi;

delVos=CMRRrec*delvp;

dcgain=1+(R2/R1);

delvo=dcgain*delVos;

```

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

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

```
//Example 5.4(b)

clear;

clc;

R1=10*10^3;

R2=100*10^3;

CMRRdB=57;//refer curve of fig.5A.6 at 10 kHz

CMRRrec=10^(-(CMRRdB/20));//Reciprocal of CMRR

delvi=10;

delvp=(R2/(R1+R2))*delvi;

delVos=CMRRrec*delvp;

dcgain=1+(R2/R1);

delvo=dcgain*delVos;

printf("Typical change in vo=%.3f V",delvo);
```

Scilab code Exa 5.5 Error in Input Offset due to PSRR

```
//Example 5.5
```

```
clear;

clc;

R1=100;

R2=100*10^3;

delvs=0.1;

dcgain=1+(R2/R1);

PSRRremin=30*10^(-6);//Minnimum rating of the
    reciprocal of PSRR

PSRRremax=150*10^(-6);

delVosmin=delvs*PSRRremin;

delVosmax=delvs*PSRRremax;

delvomin=delVosmin*dcgain;

delvomax=delVosmax*dcgain;

printf("The output ripple is=%.2f mV(typical)",
    delvomin*10^3);

printf(" and %.2f mV(maximum) peak to peak",delvomax
    *10^3);
```

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

```
//Example 5.6
```

```
clear;

clc;

atyp=10^5;//typical value of a

amin=10^4;//minimum value of a

TCVosavg=3*10^(-6);

CMRRdBtyp=100;//typical value of CMRR in dB

CMRRrectyp=10^(-CMRRdBtyp/20);

PSRRdBtyp=100;//typical value of PSRR in dB

PSRRrectyp=10^(-PSRRdBtyp/20);

CMRRdBmin=80;//minimum value of CMRR in dB

CMRRrecmax=10^(-CMRRdBmin/20);

PSRRdBmin=80;//minimum value of PSRR in dB

PSRRrecmax=10^(-PSRRdBmin/20);

Tmin=0;

Tmax=70;

Vs=15;

vpmin=-1;

vpmax=1;

vomin=-5;
```

```
vomax=5;

Troom=25;

delVos1=TCVosavg*(Tmax-Troom);

delVos2typ=vpmax*CMRRrectyp;

delVos2max=vpmax*CMRRrecmax;

delVos3typ=2*(0.05*Vs)*PSRRrectyp;

delVos3max=2*(0.05*Vs)*PSRRrecmax;

delVos4typ=vomax/atyp;

delVos4max=vomax/amin;

delVoswor=delVos1+delVos2max+delVos3max+delVos4max;

deVospro=((delVos1^2)+(delVos2typ^2)+(delVos3typ^2)
    +(delVos4typ^2))^(1/2);

printf("Worst Change in Vos=(+-)%.2f uV",delVoswor
    *10^6);

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

```
//Example 5.7

```

```
clear;

clc;

As=-10;

Rpot=10*10^3;

Vpot=15;

EImax=15*10^(-3);

Vosmax=6*10^(-3);

Iosmax=200*10^(-9);

Rpmax=(EImax-Vosmax)/Iosmax;//Parallel Combination
    of R1 and R2

R1max=(abs(As)+1)*(Rpmax/abs(As));

R1=R1max-(2.5*10^3);//Standardising R1

R2=abs(As)*R1;

Rp=(R1*R2)/(R1+R2);

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

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

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

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

```
//Example 5.8

clear;

clc;

As=-5;

Ri=30*10^3;

Vs=15;

R1=Ri;

R2=abs(As)*R1;

Rp=(R1*R2)/(R1+R2);

Vosmax=6*10^(-3);

Iosmax=200*10^(-9);

EImax=Vosmax+(Rp*Iosmax);

RA=1*10^3;

Rpc=Rp-RA;

EImaxs=EImax+(4*10^(-3));

RB=RA*(Vs/EImaxs);

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

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

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

```

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

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

printf("\nRB=%.2f Mohms",RB*10^(-6));

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

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

```
//Example 5.9(a)

clear;

clc;

As=5;

Vs=15;

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

R2=(As-1)*R1;

Rp=(R1*R2)/(R1+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=%.2f kohms",R1*10^(-3));

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

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

printf("\nRA=%.f ohms",RA);

printf("\nRB=%.f kohms",(RB*10^(-3))+0.66);

printf("\nRC=%.f kohms",RC*10^(-3));
```

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

```
//Example 5.9(b)

clear;

clc;

As=100;

```

```
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));

```

```
printf("\nRA=%.f ohms",RA+1);

printf("\nRB=%.f kohms",(RB*10^(-3))+15.63);

printf("\nRC=%.f kohms",RC*10^(-3));
```

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

```
//Example 5.10

clear;

clc;

T=25;

Ib=75*10^(-9);

Ios=80*10^(-9);

Vos=100*10^(-6);

Vs=15;

R1=4.99*10^(3);

R2=365;

R3=4.99*10^3;

R4=499;

R5=499;

```

```
R6=20*10^3;
R7=19.6*10^3;
R8=100;
R9=100*10^3;
R10=1*10^3;
C=100*10^(-12);
EI1=Vos+(((R1*(R2+(R8/2)))/(R1+(R2+(R8/2))))*Ib);
EI2=EI1;
EI3=Vos+(((R4*R6)/(R4+R6))*Ios);
A=10^3;
Eo=(A*(EI1+EI2))+((R6/R4)*EI3);
Eos=Eo+64*10^(-3);
Vx=Eos;
RB=100*10^3;
RA=RB/abs(Vs/Vx);
RC=100*10^3;///Choosing RC=100 kohms
printf("RA=%.f kohms",RA*10^(-3));
printf("\nRB=%.f kohms",RB*10^(-3));
printf("\nRC=%.f kohms",RC*10^(-3));
```

Scilab code Exa 5.11 Absolute Maximum Ratings

```
//Example 5.11

clear;

clc;

Tmax=70;

T=100;

Iqmax=2.8*10^(-3);

VCC=15;

VEE=-15;

P1=(VCC-VEE)*Iqmax;

P=310*10^(-3);

Io=(P-P1)/VCC;

PC=5.6*10^(-3);

Pmax=P+((Tmax-T)*PC);

Io=(Pmax-P1)/VCC;

printf("Maximum Current at 100degC=%.1f mA",Io*10^3)
    ;
```

Scilab code Exa 5.12 Overload Protection Maximum Ratings

```
//Example 5.12

clear;

clc;

R6=27;

b14=250;

b15=b14;

Vbe15on=0.7;

IC14=Vbe15on/R6;

IB14=IC14/b14;

i=0.18*10^(-3);

IC15=i-IB14;

Isc=IC14+IC15;

printf("IC14=%.f mA",IC14*10^3);

printf("\nIB14=%.3f mA",IB14*10^3);

printf("\nIC15=%.f uA",IC15*10^6);

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

```
//Example 6.1(a)

clear;

clc;

R1=2*10^3;

R2=18*10^3;

b=0.1;

fb=100*10^3;

emmax=0.01;

fmax=((((1/(1-emmax))^2)-1)*(fb^2))^(1/2);

printf("f<=%.1f kHz",fmax*10^(-3));
```

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

```
//Example 6.1(b)

clear;

clc;

R1=2*10^3;

R2=18*10^3;

b=0.1;

fb=100*10^3;

efimax=5;

fmax=tan(efimax*%pi/180)*fb;

printf("f<=%.2f kHz",fmax*10^(-3));
```

Scilab code Exa 6.2.a Gain Bandwidth Tradeoff

```
//Chapter-6
//Page No.-265
//Example 6.2(a)
//Gain Bandwidth Tradeoff

A0dB=60;

A0=10^(A0dB/20);
```

```
ft=10^6;

fb=ft/A0;

A10=A0^(1/2);

A20=A10;

fb1=ft/A10;

fb2=fb1;

R1=1*10^3;

R2=(A10-1)*R1;

printf("Designed Audio Amplifier :");

printf("\nOperational Amplifier-1 :");

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

printf("\nR2=%.1f kohms",(R2*10^(-3))+0.3);

printf("\n\nOperational Amplifier-2 :");

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

printf("\nR2=%.1f kohms",(R2*10^(-3))+0.3);
```

Scilab code Exa 6.2.b Gain Bandwidth Tradeoff

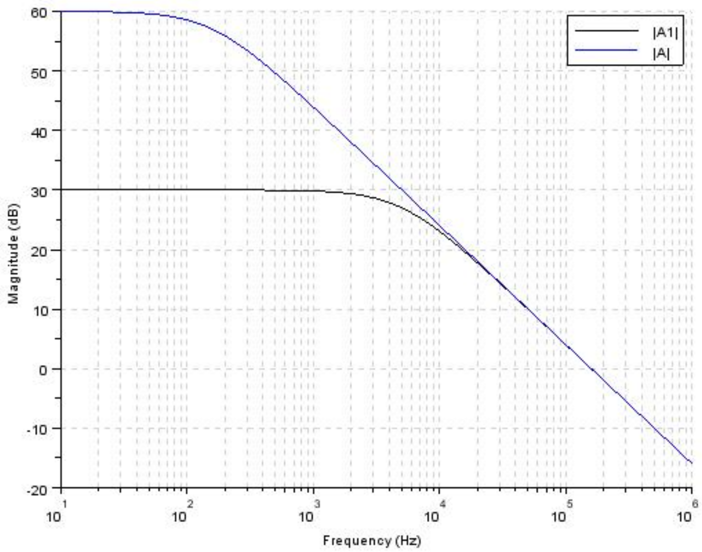

Figure 6.1: Gain Bandwidth Tradeoff

```
//Example 6.2(b)

clear;

clc;

A0dB=60;

A0=10^(A0dB/20);

ft=10^6;

fb=ft/A0;

A10=A0^(1/2);

A20=A10;

fb1=ft/A10;

fb2=fb1;

f1=1+(%s/fb1);

A1=A10*(1/f1);

y1=syslin('c',A1);


f2=1+(%s/fb);

A=A0*(1/f2);

y2=syslin('c',A);

gainplot([y1;y2],10,10^6,['|A1|';'|A|']);
```

Scilab code Exa 6.2.c Gain Bandwidth Tradeoff

```
//Example 6.2(c)

clear;

clc;

A0dB=60;

A0=10^(A0dB/20);

ft=10^6;

fb=ft/A0;

A10=A0^(1/2);

A20=A10;

fb1=ft/A10;

fb2=fb1;

f1=1+(%s/fb1);

A1=A10*(1/f1);

fB=(((((A10^2)*(2^(0.5)))/A0)-1)^(1/2))*fb1;

printf("Actual Bandwidth (fB)=%.2f kHz",fB*10^(-3));
```

Scilab code Exa 6.4 Input Impedance of Series Topology

```
//Example 6.4

clear;

clc;

rd=1*10^6;

rc=1*10^9;

a0=10^5;

ro=100;

ft=1*10^6;

R1=2*10^3;

R2=18*10^3;

b=R1/(R1+R2);

fB=b*ft;

Rs=rd;

Rd=rd*(1+(a0*b));

Rp=((2*rc)*Rd)/((2*rc)+Rd);

Ceq=1/(2*%pi*fB*rd);

f1=(Rs/Rp)*fB;

printf("Element Values in the Equivalent Circuit of
   Zi :");
```

```
printf("\nRs=%.2f Mohms",Rs*10^(-6));

printf("\nRp=%.2f Gohms",Rp*10^(-9));

printf("\nCeq=%.2f pF",Ceq*10^12);

printf("\n\nBreakpoint Frequencies of Magnitude Plot
    :");

printf("\nfB=%.2f kHz",fB*10^(-3));

printf("\nf1=%.2f Hz",f1);
```

Scilab code Exa 6.5 Output Impedance of Shunt Topology

```
//Example 6.5

clear;

clc;

rd=1*10^6;

rc=1*10^9;

a0=10^5;

ro=100;

ft=1*10^6;

R1=2*10^3;

R2=18*10^3;
```

```

b=R1/(R1+R2);

fb=ft/a0;

fB=b*ft;

Rp=ro;

Rs=ro/(1+(a0*b));

Leq=ro/(2*%pi*fB);

printf("Element Values in the Equivalent Circuit of
    Zo :");

printf("\nRs=%.f mohms",Rs*10^(3));

printf("\nRp=%.2f ohms",Rp);

printf("\nLeq=%.f uH",Leq*10^6);

printf("\n\nBreakpoint Frequencies of Magnitude Plot
    :");

printf("\nfb=%.2f Hz",fb);

printf("\nft=%.2f MHz",ft*10^(-6));
```

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

```
//Example 6.6(a)

clear;
```

```
clc;
R=100*10^3;
R1=2*10^3;
R2=18*10^3;
b=R1/(R1+R2);
A0=-(1+(R2/R1))*R;
a0=2*10^5;
ft=1*10^6;
ro=100;
fB=b*ft;
Ri=[R+((R1*R2)/(R1+R2))]/(1+(a0*b));
Ro=ro/(1+(a0*b));
fb=ft/a0;
printf("A(jf)=(%d V/A)",A0);
printf("/(1+(jf/%.d))",fB);
printf("\nZi(jf)=%.d",Ri);
printf("*(1+j(f/%.d))",fb);
printf("/(1+(jf/%.d)) ohms",fB);
printf("\nZo(jf)=%.d",Ro*10^3);
```

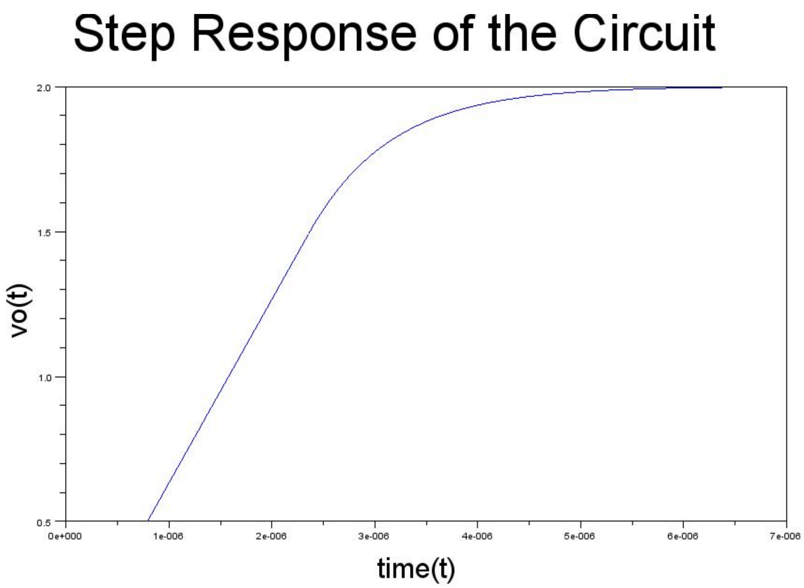

Figure 6.2: Effect of Slew Rate Limiting

```

printf("*(1+j(f/%.d))",fb);

printf("/(1+(jf/%.d))  mohms",fB);
```

Scilab code Exa 6.7.a Effect of Slew Rate Limiting

```
//Example 6.7(a)

clear;

clc;

IA=19.6*10^(-6);

```

```
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);

```

```
t11=[tau:2*10^(-8):t1+tau];

vo1=SR*(t11-tau);

t22=[t1+tau:2*10^(-8):7*10^(-6)];

vo2=Voinf+((V1-Voinf)*exp(-(t22-t1-tau)/tau));

plot(t11,vo1);

plot(t22,vo2);

xlabel("time(t)","fontsize", 6);

ylabel("vo(t)","fontsize", 6);

title("Step Response of the Circuit","fontsize", 8);
```

Scilab code Exa 6.8.a Full Power Bandwidth

```
//Example 6.8(a)

clear;

clc;

Vs=15;

A=10;

Vim=0.5;

SR=0.5*10^6;

Vom=A*Vim;
```

```

fmax=SR/(2*%pi*Vom);

printf("fmax=%.f kHz",fmax*10^(-3));
```

Scilab code Exa 6.8.b Full Power Bandwidth

```
//Example 6.8(b)

clear;

clc;

Vs=15;

A=10;

f=10*10^3;

SR=0.5*10^6;

Vommax=SR/(2*%pi*f);

Vimmax=Vommax/A;

printf("Maximum Value of Vim before the output
    distorts=%.3f V",Vimmax);
```

Scilab code Exa 6.8.c Full Power Bandwidth

```
//Example 6.8(c)

clear;
```

```
clc;

Vs=15;

A=10;

Vim=40*10^(-3);

SR=0.5*10^6;

fmax=SR/(2*%pi*Vim*A);

ft=1*10^6;

fB=ft/A;

printf("Useful Frequency Range of Operation f<=%.2f kHz",fB*10^(-3));
```

Scilab code Exa 6.8.d Full Power Bandwidth

```
//Example 6.8(d)

clear;

clc;

Vs=13;

A=10;

ft=1*10^6;

SR=0.5*10^6;
```

```
f=2*10^3;

Vommax=SR/(2*%pi*f);

if Vommax>Vs then
Vimmax=Vs/A;

printf("Useful Input Amplitude Range is Vim<=%.2f V"
    ,Vimmax);
```

Scilab code Exa 6.9 Effect of finite GBP on Integrator Circuits

```
//Example 6.9

clear;

clc;

f0=10*10^3;

Q=25;

HobpdB=0;

R1=10*10^3;//Assumption

R2=R1;//Assumption

R5=R1;//Assumption

R6=R1;//Assumption

R3=250*10^3;//Assumption

```

```
R4=R3;//Assumption

C1=1/(2*%pi*f0*R5);//Assumption

C2=C1;//Assumption

f0reler=0.01;//as relative error defined for f0=1%

Qreler=0.01

ftf0=f0/f0reler;

ftQ=(4*Q*f0)/Qreler;

printf("Designed Biquad Filter :")

printf("\nR1=R2=R5=R6=%.2f kohms",R1*10^(-3));

printf("\nR3=R4=%.2f kohms",R4*10^(-3));

printf("\nC1=C2=%.2f nF",C1*10^9);

if ftf0>ftQ then
    ft=ftf0;

else ft=ftQ

printf("\nGBP>=%.2f MHz",ft*10^(-6));
```

Scilab code Exa 6.10.b Biquad Filter with Phase Compensation

```
//Example 6.10(b)

clear;

```

```
clc;

f0=10*10^3;

Q=25;

HobpdB=0;

R1=10*10^3;//Assumption

R2=R1;//Assumption

R5=R1;//Assumption

R6=R1;//Assumption

R3=250*10^3;//Assumption

R4=R3;//Assumption

C1=1/(2*%pi*f0*R5);//Assumption

C2=C1;//Assumption

f0reler=0.01;//as relative error defined for f0=1%

Qreler=0.01

ftf0=f0/f0reler;

ftQ=(4*Q*f0)/Qreler;

ft=1*10^6;

//Changing the component values using Phase
   Compensation

ch=f0/ft;
```

```
C1new=C1-(C1*ch);

C2new=C1new;

printf("Designed Biquad Filter :")

printf("\nR1=R2=R5=R6=%.2f kohms",R1*10^(-3));

printf("\nR3=R4=%.2f kohms",R4*10^(-3));

printf("\nC1=C2=%.3f nF",C1new*10^9);
```

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

```
//Example 6.11

clear;

clc;

C=(5/%pi)*10^(-9);

R1=10*10^3;

R2=30*10^3;

GBP=1*10^6;

Hreler=0.01;//Departure of H from Hideal

ft=1*10^6;

fx=ft/(1+(R2/R1));

```

```
fmax=((1/((1-Hreler)^2)-1)^(1/2))*fx;

f0=1/(2*%pi*R1*C);

fmin3dB=(1/((1/(f0^2))-(1/(fx^2))-(1/((f0^2)*(fx^2))
   )))^(1/2);//f(-3dB)

f3dBer=((fmin3dB-f0)/fmin3dB)*100;

printf("Percentage Deviation of cut off frequency=%
   .2f",f3dBer*2);
```

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

```
//Example 6.12

clear;

clc;

C=10*10^(-9);

H0bpdB=0;

f0=10*10^3;

Q=10;

H0bp=10^(H0bpdB/20);

R1=Q/(2*%pi*f0*C*H0bp);

R2=(R1/((2*(Q^2))/(H0bp)))-1;

R3=(2*Q)/(2*%pi*f0*C);
```

```
BW=f0/Q;

BWer=0.01;//BW deviation from its design value is 1%

GBPmin=(2*Q*f0)/BWer;

printf("Components for the mentioned circuit :");

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

printf("\nR2=%.2f ohms",R2);

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

printf("\nGBP>=%.2f MHz",GBPmin*10^(-6));
```

Scilab code Exa 6.14 Parameters for Current Feedback Amplifier

```
//Example 6.14

clear;

clc;

zo=0.71*10^6;

Req=zo;

fb=350*10^3;

Ceq=1/(2*%pi*Req*fb);

vo=5;

```

```
iN=vo/Req;

printf("Ceq=%.2f pF",Ceq*10^12);

printf("\niN=%.2f uA",iN*10^6);
```

Scilab code Exa 6.15 Current Feedback Amplifier Dynamics

```
//Example 6.15

clear;

clc;

ft=100*10^6;

brec=1.5*10^3;

R2=1.5*10^3;

rn=50;

A01=1;

A02=10;

A03=100;

//R11=R2/(A01-1) ->R1=infinity

R12=R2/(A02-1);

R13=R2/(A03-1);

fB1=ft/(1+(A01/30));
```

```
fB2=ft/(1+(A02/30));

fB3=ft/(1+(A03/30));

tR1=2.2/(2*%pi*fB1);

tR2=2.2/(2*%pi*fB2);

tR3=2.2/(2*%pi*fB3);

printf("Values of R1, fB and tR for A0=1 :")

printf("\nR1=infinity");

printf("\nfB=%.2f MHz",fB1*10^(-6));

printf("\ntR=%.2f nS",tR1*10^9);

printf("\n\nValues of R1, fB and tR for A0=10 :")

printf("\nR1=%.2f ohms",R12);

printf("\nfB=%.2f MHz",fB2*10^(-6));

printf("\ntR=%.2f nS",tR2*10^9);

printf("\n\nValues of R1, fB and tR for A0=100 :")

printf("\nR1=%.2f ohms",R13);

printf("\nfB=%.2f MHz",fB3*10^(-6));

printf("\ntR=%.2f nS",tR3*10^9);
```

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

```
//Example 6.16

clear;

clc;

A0=10;

fB=100*10^6;

brec=1.5*10^3;

rn=50;

R2=brec-(rn*A0);

R1=R2/(A0-1);

printf("(a) Redisigned Current Feedback Amplifier of
    Example 6.15 :");

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

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

z0=0.75*10^6;

T0=(1/brec)*z0;

epsilon=-100/T0;

printf("\n\n(b) Percentage dc gain error=%.1f",
   epsilon);
```

Chapter 7

Noise

Scilab code Exa 7.1.a Noise Properties

```
//Example 7.1(a)

clear;

clc;

fL=0.1;

fH=100;

enw=20*10^(-9);

fce=200;

En=enw*sqrt((fce*log(fH/fL))+fH-fL);

printf("Estimated RMS input voltage=%.2f uV",En
   *10^6);
```

Scilab code Exa 7.1.b Noise Properties

```
//Example 7.1(b)

clear;

clc;

fL=20;

fH=20*10^3;

enw=20*10^(-9);

fce=200;

En=enw*sqrt((fce*log(fH/fL))+fH-fL);

printf("Estimated RMS input voltage=%.2f uV",En
    *10^6);
```

Scilab code Exa 7.1.c Noise Properties

```
//Example 7.1(c)

clear;

clc;

fL=0.1;

fH=1*10^6;

enw=20*10^(-9);

```

```
fce=200;

En=enw*sqrt((fce*log(fH/fL))+fH-fL);

printf("Estimated RMS input voltage=%.1f uV",En
   *10^6);
```

Scilab code Exa 7.3 Graphical Representation of Noise Dynamics

```
//Example 7.3

clear;

clc;

fL1=1;

fH1=1*10^3;

fL2=fH1;

fH2=10*10^3;

fL3=fH2;

//fH3=infinity

enw=20*10^(-9);

fce=100;

Eno1=enw*sqrt((fce*log(fH1/fL1))+fH1-fL1);

eno=enw/fL2;

```

```
Eno2=sqrt(integrate("(eno*f)^2",'f',fL2,fH2));

f0=100*10^3;

enw3=200*10^(-9);

Eno3=enw3*sqrt((1.57*f0)-fL3);

Eno=sqrt((Eno1^2)+(Eno2^2)+(Eno3^2));

printf("Estimated rms noise voltage=%.1f uV",Eno
   *10^6);
```

Scilab code Exa 7.4 Calculation of Thermal Noise

```
//Example 7.4

clear;

clc;

R=10*10^3;

k=1.38*10^(-23);

T=25+273;//Room Temperature in Kelvin

eR=sqrt(4*k*T*R);

printf("(a) Noise Voltage(eR)=%.2f nV/sqrt(Hz)",eR
   *10^9);

iR=eR/R;

printf("\n(b) Noise Current(iR)=%.2f pA/sqrt(Hz)",iR
```

```
    *10^12);

fH=20*10^3;

fL=20;

ER=eR*sqrt(fH-fL);

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

```
//Example 7.5(a)

clear;

clc;

ID=1*10^(-6);

fH=1*10^6;

q=1.602*10^(-19);

In=sqrt(2*q*ID*fH);

SNR=20*log10(ID/In);

printf("Signal to Noise Ratio=%.1f dB",SNR);
```

Scilab code Exa 7.5.b Calculation of Shot Noise

```
//Example 7.5(b)

clear;

clc;

ID=1*10^(-9);

fH=1*10^6;

q=1.602*10^(-19);

In=sqrt(2*q*ID*fH);

SNR=20*log10(ID/In);

printf("Signal to Noise Ratio=%.1f dB",SNR);
```

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

```
//Example 7.7(a)

clear;

clc;

R1=100*10^3;

R2=200*10^3;

R3=68*10^3;

enw=20*10^(-9);

fce=200;
```

```
ft=1*10^6;

inw=0.5*10^(-12);

fci=2*10^3;

Rp=(R1*R2)/(R1+R2);

Ano=1+(R2/R1);

fB=ft/Ano;

fL=0.1;

Enoe=Ano*enw*sqrt([{fce*log(fB/fL)}+{1.57*fB}-fL]);

Enoi=Ano*[{(R3^2)+(Rp^2)}^(1/2)]*inw*([(fci*log(fB/
    fL))+(1.57*fB)]^(1/2));

k=1.38*10^(-23);

T=25+273;//Room temperature in Kelvin

EnoR=Ano*[{(4*k*T)*(R3+Rp)*1.57*fB}^(1/2)];

Eno=sqrt((Enoe^2)+(Enoi^2)+(EnoR^2));

printf("RMS Output Noise Voltage=%.f uV",Eno*10^6);

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

```
//Example 7.8

clear;

clc;

R1=100*10^3;

R2=200*10^3;

R3=68*10^3;

enw=20*10^(-9);

fce=200;

ft=1*10^6;

inw=0.5*10^(-12);

fci=2*10^3;

Rp=(R1*R2)/(R1+R2);

Ano=1+(R2/R1);

fB=ft/Ano;

fL=0.1;

Enoeold=Ano*enw*sqrt([{fce*log(fB/fL)}+{1.57*fB}-fL
   ]);

Enoiold=Ano*[{(R3^2)+(Rp^2)}^(1/2)]*inw*([(fci*log(
   fB/fL))+(1.57*fB)]^(1/2));

k=1.38*10^(-23);

```

```
T=25+273;//Room temperature in Kelvin

EnoRold=Ano*[{(4*k*T)*(R3+Rp)*1.57*fB}^(1/2)];

Enoold=sqrt((Enoeold^2)+(Enoiold^2)+(EnoRold^2));

Enonew=50*10^(-6);//New Value of Eno mentioned in
   problem

Enoisum=(Enonew^2)-(Enoeold^2);//sum of (Enoi^2) and
    (EnoR^2)

Enoinewsq=(Ano^2)*(inw^2)*[(fci*log(fB/fL))+(1.57*fB
   )];//(Enoinew^2)/(R^2)

EnoRnewsq=(Ano^2)*((4*k*T)*1.57*fB);

r=poly(0,'x');

p=(Enoinewsq*(r^2))+(EnoRnewsq*r)-Enoisum;

[r1]=roots(p);

R=r1(2,1)

R3new=R/2;

R1new=(3*R3new)/2;

R2new=2*R1new;

printf("Resistances after scaling are :");

printf("\nR1=%.2f kohms",R1new*10^(-3));

printf("\nR2=%.1f kohms",R2new*10^(-3));

printf("\nR3=%.1f kohms",R3new*10^(-3));
```

Scilab code Exa 7.9 Calculation of Signal to Noise Ratio

```
//Example 7.9

clear;

clc;

R1=100*10^3;//From Example 7.7(a)

R2=200*10^3;//From Example 7.7(a)

Aso=-(R2/R1);

Eno=154*10^(-6);//From Example 7.9

Eni=Eno/abs(Aso);

Vipa=0.5;//Peak amplitude of input ac signal

Virms=Vipa/sqrt(2);

SNR=20*log10(Virms/Eni);

printf("SNR of the circuit of Example 7.7=%.1f dB",
   SNR);
```

Scilab code Exa 7.10 Calculation of Noise in Current Feedback Amplifier

```

//Example 7.10
```

```
clear;
clc;
z0=710*10^3;
fb=350*10^3;
rn=50;
enw=2.4*10^(-9);
fce=50*10^3;
inpw=3.8*10^(-12);
fcip=100*10^3;
innw=20*10^(-12);
fcin=100*10^3;
R1=166.7;
R2=1.5*10^3;
R3=100;//internal resistance
fL=0.1;
Rp=(R1*R2)/(R1+R2);
ft=(z0*fb)/R2;
fB=ft/[1+(rn/((R1*R2)/(R1+R2)))];
Ano=1+(R2/R1);
```

```

Enoe=enw*sqrt([{fce*log(fB/fL)}+{1.57*fB}-fL]);

Enoi=R3*inpw*sqrt(((fcip*log(fB/fL))+(1.57*fB)-fL));

Enop=Rp*innw*sqrt({(fcin*log(fB/fL))+(1.57*fB)-fL});

k=1.38*10^(-23);

T=25+273;//Room temperature in Kelvin

EnoR=[{(4*k*T)*(R3+Rp)*((1.57*fB)-fL)}^(1/2)];

Eno=Ano*sqrt((Enoe^2)+(Enoi^2)+(EnoR^2)+(Enop^2));

c=6.6*10^3;

Eno1=Eno*c;

printf("RMS Noise Voltage (Eno)=%.2f uV",Eno*10^6);
    //answer in textbook is wrong

printf("\nPeak to Peak Noise Voltage (Eno)=%.2f mV",
    Eno1);//answer in textbook is wrong
```

Scilab code Exa 7.11 Noise in Photodiode Amplifiers

```
//Example 7.11

clear;

clc;

ft=16*10^6;

```

```
enw=4.5*10^(-9);
fce=100;
IB=1*10^(-12);
fL=0.01;
R1=100*10^(9);
C1=45*10^(-12);
R2=10*10^6;
C2=0.5*10^(-12);
b0rec=1;
binfrec=91;
fz=350;
fp=31.8*10^3;
fx=176*10^3;
k=1.38*10^(-23);
T=25+273;
iR2=sqrt((4*k*T)/R2);
q=1.602*10^(-19);
in=sqrt(2*q*IB);
Enoe=binfrec*enw*sqrt(((%pi/2)*fx)-fp);
```

```
EnoR=R2*iR2*sqrt((%pi/2)*fp);

Eno=sqrt((Enoe^2)+(EnoR^2));

printf("Total Output Noise=%.f uV",Eno*10^6);
```

Scilab code Exa 7.12 Photodiode amplifier with Noise Filtering

```
//Example 7.12

clear;

clc;

ft=16*10^6;

enw=4.5*10^(-9);

fce=100;

IB=1*10^(-12);

fL=0.01;

R1=100*10^(9);

C1=45*10^(-12);

R2=10*10^6;

C2=0.5*10^(-12);

b0rec=1;

binfrec=91;
```

```
fz=350;

fp=31.8*10^3;

fx=176*10^3;

k=1.38*10^(-23);

T=25+273;

Cc=0.5*10^(-12);//Assumed

C2=Cc;

C3=10*10^(-9);

R3=(R2*Cc)/C3;

printf("Cc=C2=%.1f pF",Cc*10^(12));

printf("\nR3=%.f ohms",R3);

printf("\nC3=%.f nF",C3*10^(9));
```

Scilab code Exa 7.13 Designing T Feedback Photodiode Amplifiers

```
//Example 7.13

clear;

clc;

C1=2*10^(-9);

```

```
binfrec=4000;

inw=0.566*10^(-15);

T=1*10^(9);

ft=16*10^6;

R1=100*10^(9);

C2=0.5*10^(-12);

fx=(1/binfrec)*ft;

enw=4.5*10^(-9);

Enoe=binfrec*enw*sqrt((%pi*fx)/2);

EnoRmax=Enoe/3;

k=1.38*10^(-23);

Temp=25+273;

ex=((EnoRmax^2)*C2)/(k*Temp);

R2=T/ex;

R3=1*10^3;//Assumed

R4=(ex-1)*R3;

printf("(a) Designed T Network :");

printf("\n      R1=%.2f Gohms",R1*10^(-9));

printf("\n      R2=%.1f Mohms",R2*10^(-6));

```

```
printf("\n     R3=%.2f kohms",R3*10^(-3));
printf("\n     R4=%.2f kohms",R4*10^(-3));
printf("\n     C1=%.2f nF",C1*10^9);
printf("\n     C2=%.2f pF",C2*10^12);
fp=1/(2*%pi*ex*R2*C2);
fB=fp;
Rp=(R1*R2)/(R1+R2);
Enoi=((1.57*fB)^(1/2))*inw;
Eno=sqrt((Enoe^2)+(Enoi^2)+(EnoRmax^2));
printf("\n\n(b) Total rms Output Noise=%.2f mV",Eno*10^3);
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

```
//Example 8.1

clear;

clc;

T0=10^4;

f1=100;

f2=10^6;

f3=10*10^6;

w1=2*%pi*f1;

w2=2*%pi*f2;

w3=2*%pi*f3;

h=syslin('c',T0/[(1-(%s/w1))*(1-(%s/w2))*(1-(%s/w3))
```

```
]);

gm=g_margin(h);

pm=p_margin(h);

printf("(a) Gain Margin(GM)=%.2f dB",gm);

printf("\n(b) Phase Margin(PM)=%.1f degrees",-pm);

f=512*10^3;

w=2*%pi*f;

T1=T0/[(1-((%i*w)/w1))*(1-((%i*w)/w2))*(1-((%i*w)/w3
   ))];

den=1/(abs(T1)/T0);

printf("\n(c) T0 for PM=60 degrees=%.f",den);
```

Scilab code Exa 8.2 Stability in Differentiator Circuits

```
//Example 8.2

clear;

clc;

R=159*10^3;

C=10*10^(-9);

f0=1/(2*%pi*R*C);

```

```
ft=10^6;
fx=sqrt(f0*ft);
Q=sqrt(ft/f0);
d=-90-((180/%pi)*atan(fx/f0));
pm=180+d;
printf("fx=%.2f kHz",fx*10^(-3));
printf("\nQ=%.f",Q);
printf("\nPhase Margin (PM)=%.1f degrees",pm);
```

Scilab code Exa 8.3 Stray Input Capacitance Compensation for inverting configuration

```
//Example 8.3
clear;
clc;
R1=30*10^3;
R2=R1;
Cext=3*10^(-12);
GBP=20*10^6;
Cd=7*10^(-12);
```

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

Cn=Cext+Cd+(Cc/2);

Rp=(R1*R2)/(R1+R2);

Cf1=0;

fz1=1/(2*%pi*Rp*(Cn+Cf1));

ft=20*10^6;

Q=sqrt((ft)/(2*fz1));

pm=(180/%pi)*acos((sqrt(1+(1/(4*Q^4))))-(1/(2*Q^2)))
    ;

Cf2=(R1/R2)*Cn;

fp=1/(2*%pi*R2*Cf2);

x=poly(0,'f');

A=-1/[(1+(%i*(x/fp)))*(1+(%i*(x/(0.5*ft))))];

printf("(a) Phase Margin with Cf absent=%.1f degrees
    ",pm);

printf("\n(b) Cf for PM=90 degrees=%.2f pF",Cf2
    *10^12);

printf("\n(c) A(jf)=");

disp(A);
```

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

```
//Example 8.4

clear;

clc;

R1=30*10^3;

R2=R1;

ft=20*10^6;

Cext=3*10^(-12);

GBP=20*10^6;

Cd=7*10^(-12);

Cc=12*10^(-12);

Cf=(R1/R2)*((Cc/2)+Cext);

Cn=Cext+Cd+(Cc/2);

fx=ft/(1+(Cn/Cf));

x=poly(0,'f');

A=(1+(R2/R1))/(1+(%i*(x/fx)));

printf("A(jf)=");

disp(A);

printf("V/V");
```

Scilab code Exa 8.5 Stabalizing a capacitively loaded op amp circuit

```
//Example 8.5

clear;

clc;

GBP=10*10^6;

ro=100;

A0=-2;

CL=5*10^(-9);

R1=10*10^3;

R2=20*10^3;

Rs=(R1/R2)*ro;

Cf=((1+(R1/R2))^2)*(ro/R2)*CL;

f3dB=1/(2*%pi*R2*Cf);

b=1/3;

fx=b*GBP;

printf("(a) Rs=%.f ohms",Rs);

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

```

```
x=poly(0,'f');

A=A0/((1+(%i*(x/fx)))*(1+(%i*(x/f3dB))));

printf("\n\n(b) A(jf)=");

disp(A);

printf("V/V");
```

Scilab code Exa 8.6 Internal Frequency Compensation

```
//Example 8.6

clear;

clc;

a0=3600;

f1=1*10^6;

f2=4*10^6;

f3=40*10^6;

fmin135=4.78*10^6;

fmin180=14.3*10^6;

gbp1=abs(a0/[(1+(%i*(fmin135/f1)))*(1+(%i*(fmin135/
    f3)))*(1+(%i*(fmin135/f3)))])-256;

gbp2=abs(a0/[(1+(%i*(fmin180/f1)))*(1+(%i*(fmin180/
    f3)))*(1+(%i*(fmin180/f3)))])-158.97561;
```

```
printf(" |a(j*fmin135)|=%.d  V/V",gbp1);

printf("\n|a(j*fmin180)|=%.1f  V/V",gbp2);
```

Scilab code Exa 8.7 Dominant Pole Compensation

```
//Example 8.7

clear;

clc;

PM=45;

anganewjfx=-180+PM;

a0=3600;

f1=1*10^6;

f2=4*10^6;

f3=40*10^6;

angajfx=anganewjfx+90;

fx=683*10^3;

ajf=a0/((1+(%i*(fx/f1)))*(1+(%i*(fx/f2)))*(1+(%i*(fx
    /f3))));

ang=(180/%pi)*atan(imag(ajf)/real(ajf));

mag=abs(ajf);
```

```
fd=sqrt((fx^2)/((mag^2)-1));

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

Scilab code Exa 8.8 Shunt Capacitance Compensation

```
//Example 8.8

clear;

clc;

rd=1*10^6;

g1=2*10^(-3);

R1=100*10^(3);

g2=10*10^(-3);

R2=50*10^3;

ro=100;

f1=100*10^3;

f2=1*10^6;

f3=10*10^3;

PM=45;

a0=g1*R1*g2*R2;

```

```
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);
```

```
g2=10*10^(-3);
R2=50*10^3;
ro=100;
f1=100*10^3;
f2=1*10^6;
f3=10*10^6;
PM=45;
a0=g1*R1*g2*R2;
C1=1/(2*%pi*f1*R1);
b1=1;
C21=1/(2*%pi*f2*R2);
f2newap1=g2/[2*%pi*(C1+C21)];
fx1=f3;
f1new1=f3/(b1*a0);
Cc1=1/(2*%pi*R1*g2*R2*f1new1);
f2new1=(g2*Cc1)/(2*%pi*((C1*C21)+(Cc1*C1)+(Cc1*C21)));
fz1=g2/(2*%pi*Cc1);
printf("(a) f1new=%.f Hz",f1new1);
```

```
printf("\n      f2new=%.f MHz",f2new1*10^(-6));

printf("\n      Cc=%.1f pF",Cc1*10^12);

b2=0.5;

C22=1/(2*%pi*f2*R2);

f2newap2=g2/[2*%pi*(C1+C22)];

fx2=f3;

f1new2=f3/(b2*a0);

Cc2=1/(2*%pi*R1*g2*R2*f1new2);

f2new2=(g2*Cc2)/(2*%pi*((C1*C22)+(Cc2*C1)+(Cc2*C22))
    );

fz2=g2/(2*%pi*Cc2);

printf("\n\n(b) f1new=%.f Hz",f1new2);

printf("\n      f2new=%.f MHz",f2new2*10^(-6));

printf("\n      Cc=%.1f pF",Cc2*10^12);
```

Scilab code Exa 8.10 Pole Zero Compensation

```
//Example 8.10

clear;

clc;

```

```
PM=45;

b=1;

rd=1*10^6;

g1=2*10^(-3);

R1=100*10^(3);

g2=10*10^(-3);

R2=50*10^3;

ro=100;

f1=100*10^3;

f2=1*10^6;

f3=10*10^6;

a0=g1*R1*g2*R2;

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

Cc=(b*a0)/(2*%pi*R1*f3);

Rc=1/(2*%pi*Cc*f2);

f4=1/(2*%pi*Rc*C1);

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

printf("\nRc=%.f ohms",Rc);

printf("\nR1=%.f kohms",R1*10^(-3));//The value of
    R1 is not provided in the textbook
```

```
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

```
//Example 8.11

clear;

clc;

a0=10^5;

f1=10*10^3;

f2=3*10^6;

f3=30*10^6;

R1=10*10^3;

R2=100*10^3;

PM=45;

ajf=a0/((1+(%i*(f2/f1)))*(1+(%i*(f2/f2)))*(1+(%i*(f2
    /f3))));

ajf2mag=abs(ajf);

Rc1=R2/(ajf2mag-(1+(R2/R1)));

printf("(a) Rc=%.1f ohms",Rc1);

```

```
Rc2=430;

brec=1+(R2/R1)+(R2/Rc2);

a0b=a0/brec;

dcge=-100/(a0b);

printf("\n\n(b) DC Gain Error=%.2f percent",dcge);

EI=1*10^(-3);

EO=brec*EI;

printf("\n\n(c) DC Output Error=%.f mV",EO*10^3);

fmin3dB=f2;

printf("\n\n(d) f-3dB=%.f MHz",fmin3dB*10^(-6));
```

Scilab code Exa 8.12 Input Lag Compensation

```
//Example 8.12

clear;

clc;

a0=10^5;

f1=10*10^3;

f2=3*10^6;

f3=30*10^6;
```

```
R1=10*10^3;

R2=100*10^3;

PM=45;

Rc=447.4;

Cc=5/(%pi*Rc*f2);

printf("(a) Rc=%.1f ohms",Rc);

printf("\n      Cc=%.3f nF",Cc*10^9);

b0rec=1+(R2/R1);

a0b0=a0*(1/b0rec);

dcge=-100/(a0b0);

printf("\n\n(b) DC Gain Error=%.3f percent",dcge);

EI=1*10^(-3);

EO=b0rec*EI;

printf("\n\n(c) DC Output Error=%.f mV",EO*10^3);

fmin3dB=f2;

printf("\n\n(d) f-3dB=%.f MHz",fmin3dB*10^(-6));

f=2.94*10^6;

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)))];

```

```
Tang=-(180-(180/%pi)*atan(imag(T)/real(T)));

PM1=180+Tang;

printf("\n\n(e) Actual Phase Margin=%.1f degrees",
   PM1);
```

Scilab code Exa 8.13 Feedback Lead Compensation

```
//Example 8.13

clear;

clc;

a0=10^5;

f1=1*10^3;

f2=100*10^3;

f3=5*10^6;

A0=20;

R1=1.05*10^3;

R2=20*10^3;

b0=1/(1+(R2/R1));

a0b0=a0*b0;

f=700*10^3;

```

```
T=a0b0/[(1+((%i*f)/f1))*(1+((%i*f)/f2))*(1+((%i*f)/f3))];

Tang=-(180-(180/%pi)*atan(imag(T)/real(T)));

PM=180+Tang;

printf("(a) PM=%.1f degrees indicating a circuit in bad need of compensation.",PM);

amod=sqrt(20);

aang=-192.3;

fx=1.46*10^6;

Cf=sqrt(1+(R2/R1))/(2*%pi*R2*fx);

PM1=180+aang-(90-(2*(180/%pi)*atan(sqrt(1+(R2/R1)))));

printf("\n\n(b) PM after compensation=%.1f degrees",PM1);

f3dB=(1/(2*%pi*R2*Cf))+1000;

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

```
//Example 8.14

clear;

```

```
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;
```

```

fb=200*10^3;

rn=50;

R2=1.5*10^3;

Cn=100*10^(-12);

PM=45;

Cf=sqrt((rn*Cn)/(2*%pi*R2*zo*fb));

printf("Cf=%.2f pF",Cf*10^12);
```

Scilab code Exa 8.16 Feedback Lead Compensation for Composite Amplifier

```
//Example 8.16

clear;

clc;

R1=1*10^3;

R2=99*10^3;

PM=45;

ft1=1*10^6;

ft2=ft1;

Cf=sqrt((1+(R2/R1))/(ft1*ft2))/(2*%pi*R2);
```

```
a0=2*10^5;

T0=(a0^2)/100;

fp=(1/(2*%pi*R2*Cf));

fB=fp;

PMs=PM*2;

T0s=a0/100;

fBs=ft1/100;

printf("(a) Composite Amplifier with feedback Lead
   Compensation Parameters :");

printf("\n    PM=%.f degrees",PM);

printf("\n    T0=");

disp(T0);

printf("    fB=%.f kHz",fB*10^(-3));

printf("\n\n    Single Op Amp Parameters :");

printf("\n    PM=%.f degrees",PMs);

printf("\n    T0=");

disp(T0s);

printf("    fB=%.f kHz",fBs*10^(-3));

Cf2=((1+(R2/R1))^(1/4))*Cf;

```

```
fp2=(1/(2*%pi*R2*Cf2));
fz2=(1+(R2/R1))*fp2;
fx2=sqrt(fp2*fz2);
PM2=180-180-[(180/%pi)*((atan(fx2/fz2))-atan(fx2/fp2))];
printf("\n\n(b) Cf=%.1f pF",Cf2*10^12);
printf("\n    fp=%.2f kHz",fp2*10^(-3));
printf("\n    PM=%.1f degrees",PM2);
printf("\n\n(c) Increasing Cf above %.1f pF will reduce PM until eventually PM=0 degrees,",Cf2*10^12);
printf("\n    indicating the overcompensation is decremental.")
```

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

```
//Example 8.17
clear;
clc;
dcgain=-100;
R1=1*10^3;
```

```
R2=abs(dcgain)*R1;

ft1=1*10^6;

ft2=ft1;

R4R3rat=sqrt((ft2/ft1)*(1+(R2/R1)))-1;

R3=2*10^3;

R4=R3*R4R3rat;

a0=2*10^5;

T0=a0*(1+(R4/R3))/(1+(R2/R1));

fB=ft1/10;

PM=90;

T0s=a0/(1+(R2/R1));

fBs=ft1/100;

printf("Components for the Circuit :");

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

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

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

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

printf("\nAssociated Parameters with the Circuit :")
    ;

printf("\nT0=");
```

```
disp(T0);

printf("fB=%.f kHz",fB*10^(-3));

printf("\n\nSingle Op Amp Parameters :");

printf("\nT0=");

disp(T0s);

printf("fB=%.f kHz",fBs*10^(-3));
```

Chapter 9

Non Linear Circuits

Scilab code Exa 9.1 Comparator as a Level Detector I

```
//Example 9.1

clear;

clc;

Vref=2;

R1=20*10^3;

R2=30*10^3;

Vos=5*10^(-3);

IB=250*10^(-9);

Rpar=(R1*R2)/(R1+R2);

VN=Rpar*IB;

Vneti=Vos+VN;
```

```

VT=(1+(R2/R1))*(Vref-Vneti);

printf("Worst Case Error=%.f mV",Vneti*10^3);
```

Scilab code Exa 9.2 Comparator as a Level Detector II

```
//Example 9.2

clear;

clc;

Vref=2.5;

IR=1*10^(-3);

ILED=2*10^(-3);

VLED=1.8;

Vb=12;

Vbmax=13;

Vbmin=10;

R4o=(Vbmax-VLED)/ILED;

R1o=10*10^(3);

R2o=((Vbmax/Vref)-1)*R1o;

R4u=(Vbmin-VLED)/ILED;

```

```
R1u=10*10^(3);

R2u=((Vbmin/Vref)-1)*R1u;

R3u=(Vb-Vref)/IR;

printf("Designed Circuit for Voltage Indicator :");

printf("\n\nCircuit Elements for Overvoltage Circuit
    :");

printf("\nR1=%.f kohms",R1o*10^(-3));

printf("\nR2=%.2f kohms",(R2o*10^(-3))+0.2);

printf("\nR4=%.1f kohms",R4o*10^(-3));

printf("\n\nCircuit Elements for Undervoltage
   Circuit :");

printf("\nR1=%.f kohms",R1u*10^(-3));

printf("\nR2=%.1f kohms",(R2u*10^(-3))+0.1);

printf("\nR3=%.f kohms",R3u*10^(-3));

printf("\nR4=%.1f kohms",(R4u*10^(-3))-0.2);
```

Scilab code Exa 9.3 Designing On Off Temperature Controller

```
//Example 9.3

clear;

clc;
```

```
Tmin=50+273.2;//Temperature in Kelvin

Tmax=100+273.2;//Temperature in Kelvin

R2=5*10^3;

VTmax=Tmax/100;

VTmin=Tmin/100;

I2=(VTmax-VTmin)/R2;

R3=VTmin/I2;

Vref=6.9;

R1=(Vref-VTmax)/I2;

R4=2*10^3;

R5=6.2*10^3;

R6=10*10^3;

printf("Designed On-Off Temperature Controller :");

printf("\nR1=%.1f kohms",R1*10^(-3));

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

printf("\nR3=%.1f kohms",R3*10^(-3));

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

printf("\nR5=%.1f kohms",R5*10^(-3));

printf("\nR6=%.f kohms",R6*10^(-3));
```

Scilab code Exa 9.4 Comparator as a Window Detector

```
//Example 9.4

clear;

clc;

VCC=5

VCCmax=VCC+((5/100)*VCC);

VCCmin=VCC-((5/100)*VCC);

IB=1*10^(-3);

Vled=1.5;

Iled=10*10^(-3);

vN=2.5;//For Bottom Comparator

vP=2.5;//For Top Comparator

R1=10*10^3;

Rsum=R1/(vN/VCCmax);

R2=((vP/VCCmin)*(Rsum))-R1;

R3=Rsum-R1-R2;

VBE=0.7;

```

```
R4=(VCC-VBE)/IB;

R5=(VCC-vN)/IB;

R6=(VCC-Vled)/Iled;

printf("Designed Video Detector :");

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

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

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

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

printf("\nR5=%.2f kohms",(R5*10^(-3))+0.2);

printf("\nR6=%.2f ohms",R6-20);
```

Scilab code Exa 9.5 Designing Single Supply Inverting Schmitt trigger

```
//Example 9.5

clear;

clc;

VCC=5;

Vol=0;

Vtl=1.5;

Vth=2.5;
```

```
R4=2.2*10^3;//Assumed

R3=100*10^3;//Assumed (Much Greater than R4)

A=[(Vtl/(VCC-Vtl)) -1;1 -((VCC-Vth)/Vth)];

B=[((Vtl/(VCC-Vtl))*(1/R3));-((1/R3)*((VCC-Vth)/Vth)
    )];

Rrec=linsolve(A,B);

R1rec=Rrec(1,1);

R2rec=Rrec(2,1);

R1=1/R1rec;

R2=1/R2rec;

printf("Designing Single Supply Inverting Schmitt
    trigger :");

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

printf("\nR2=%.1f kohms",R2*10^(-3));

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

printf("\nR4=%.1f kohms",R4*10^(-3));
```

Scilab code Exa 9.6 Hysteresis in On Off Controllers

```
//Example 9.6

```

```
clear;

clc;

hys=1;

VBEon=0.9;

Tmin=50+273.2;//Temperature in Kelvin

Tmax=100+273.2;//Temperature in Kelvin

R2=5*10^3;

VTmax=Tmax/100;

VTmin=Tmin/100;

I2=(VTmax-VTmin)/R2;

R3=VTmin/I2;

Vref=6.9;

R1=(Vref-VTmax)/I2;

R4=2*10^3;

R5=6.2*10^3;

R6=10*10^3;

Rw=((R1+(R2/2))*(R3+(R2/2)))/((R1+(R2/2))+(R3+(R2/2)
   ));

delvo=VBEon;

sen=10*10^(-3);
```

```
delvp=2*hys*sen;

RF=((delvo*Rw)/delvp)-Rw;

printf("Designed On-Off Temperature Controller :");

printf("\nR1=%.1f kohms",R1*10^(-3));

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

printf("\nR3=%.1f kohms",R3*10^(-3));

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

printf("\nR5=%.1f kohms",R5*10^(-3));

printf("\nR6=%.f kohms",R6*10^(-3));

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

```
//Example 10.1

clear;

clc;

f0min=1;

f0max=10*10^3;

VDon=0.7;

Vsa=5;

Vz5=Vsa-(2*VDon);

Vsat=13;

IRmin=10*10^(-6);

```

```
R1=33*10^3;

R2=R1;

VT=2.5;

Rmax=(Vsa-VT)/(IRmin);

Rpot=Rmax;

Rs=Rpot/39;

f0=0.5;

C1=1/(f0*2*(Rpot+Rs)*log(1+(2*(R1/R2))));

C2=C1/10;

C3=C2/10;

C4=C3/10;

vN=-2.5;

iRmax=(Vsa-vN)/Rs;

IR2=Vsa/(R1+R2);

IB=1*10^(-3);

ILmax=1*10^(-3);

IR3max=iRmax+IR2+IB+ILmax;

R3=(Vsat-Vsa)/IR3max;

R4=10*10^3;

```

```
printf("Designed Square Wave Generator :");

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

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

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

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

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

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

printf("\nC1=%.1f uF",(C1*10^6)-0.25);

printf("\nC2=%.2f uF",(C2*10^6)-0.02);

printf("\nC3=%.f nF",(C3*10^9)-2.50);

printf("\nC4=%.1f nF",(C4*10^9)-0.25);
```

Scilab code Exa 10.3 The 555 timer as an astable multivibrator

```
//Example 10.3

clear;

clc;

f0=50*10^3;

Dper=75;

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

```
Rsum=1.44/(f0*C);

A=[1 -2;1 2];

B=[0;-Rsum];

R=linsolve(A,B);

RA=R(1,1);

RB=R(2,1);

printf("Designed Astable Multivibrator :");

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

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

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

Scilab code Exa 10.4 Voltage Control for 555 timer

```
//Example 10.4

clear;

clc;

VCC=5;

Vpeak=1;

Vth=((2/3)*VCC);

```

```
Vthmin=((2/3)*VCC)-1;

Vthmax=((2/3)*VCC)+1;

Vtl1=Vthmin/2;

Vtl2=Vthmax/2;

f0=50*10^3;

Dper=75;

C=1*10^(-9);

Rsum=1.44/(f0*C);

A=[1 -2;1 2];

B=[0;-Rsum];

R=linsolve(A,B);

RA=R(1,1);

RB=R(2,1);

Tl=RB*C*log(2);

Th1=(RA+RB)*C*log((VCC-Vtl1)/(VCC-Vthmin));

Th2=(RA+RB)*C*log((VCC-Vtl2)/(VCC-Vthmax));

T1=Tl+Th1;

T2=Tl+Th2;

f01=1/T1;

```

```
f02=1/T2;

D1=(100*Th1)/T1;

D2=(100*Th2)/T2;

printf("Range of Variation of f0 :%.1f kHz<=f0<=",(
   f02*10^(-3))+0.2);

printf("%.1f kHz",(f01*10^(-3))+0.6);

printf("\nRange of Percentage Variation of D :");

printf("%.1f",D1);

printf("<=D<=");

printf("%.1f",D2);
```

Scilab code Exa 10.5 Designing Basic Triangular or Square Wave Generator

```
//Example 10.5

clear;

clc;

Vclamp=5;

VT=10;

VDon=0.7;

Vz5=Vclamp-(2*VDon);
```

```
Rrat=Vclamp/VT;

R1=20*10^3;

R2=R1*Rrat;

f0min=10;

f0max=10*10^3;

f0range=f0max/f0min;

Rpot=2.5*10^6;

Rs=Rpot/f0range;

Rmin=Rs;

C=(R2/R1)/(4*Rmin*f0max);

IRmax=Vclamp/Rmin;

IR2max=Vclamp/R2;

Ib=1*10^(-3);

Il=1*10^(-3);

Vsat=13;

IR3max=IRmax+IR2max+Ib+Il;

R3=(Vsat-Vclamp)/IR3max;

printf("Designed Basic Triangular/Square Wave
   Generator :");

```

```
printf("\nR=%.1f kohms",Rmin*10^(-3));

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

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

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

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

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

```
//Example 10.6

clear;

clc;

VCC=15;

f0=10*10^3;

iA=100*10^(-6);

iB=iA;

R=(VCC/5)/iA;

C=0.3/(f0*R);

Rp=10*10^3;

Rsym=5*10^3;

```

```
Rre=R-(Rsym/2);

Rthd=100*10^3;

printf("Components for the Circuit :");

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

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

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

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

printf("\nTo calibrate the circuit, adjust Rsym so
    that the square wave has D(percent)=50,");

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

```
//Example 10.7

clear;

clc;

K=10;

VT0=(273.2*10^(-3));//273.2 K for T=0 degCelsius

fo0=0;

```

```
R2R3rat=(1-VT0)/VT0;

RC=1/((10^4)*K);

C=3.9*10^(-9);

R=RC/C;

R3=2.74*10^3;

R2=R3*R2R3rat;

R1=R-((R2*R3)/(R2+R3));

printf("Designed Celsius to Frequency Converter :");

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

printf("\nR1=%.f ohms",R1);

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

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

printf("\nC=%.1f nF",C*10^9);

printf("\nTo calibrate, place the IC in a 0 deg
    Celsius environment and adjust R2,");

printf("\nso that the circuit is barely oscillating,
    say fo=1 Hz. Then move the IC to");

printf("\na 100 deg Celsius environment and adjust
    R1 for f0=1 kHz.");
```

Scilab code Exa 10.8 Designing a Voltage to Frequency Converter

```
//Example 10.8

clear;

clc;

vI=10;

f=100*10^3;

T=1/f;

D=25;

TH=2.5*10^(-6);

C=(TH*1*10^(-3))/7.5;

R=vI/(7.5*f*C);

delvImax=2.5;

C1=(10^(-3)*TH)/delvImax;

RA=62;

RB=150*10^3;

RC=100*10^3;

printf("Designed Voltage to Frequency Converter :");

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

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

```

```
printf("\nC1=%.f nF",C1*10^9);

printf("\nRA=%.f ohms",RA);

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

printf("\nRC=%.f kohms",RC*10^(-3));
```

Chapter 11

Voltage Referencres and Regulators

Scilab code Exa 11.1 Line and Load Regulation

```
//Example 11.1

clear;

clc;

Vimin=7;

Vimax=25;

Vo=5;

delVi=Vimax-Vimin;

delVovi=3*10^(-3);

Iomin=0.25;

Iomax=0.75;
```

```
delIo=Iomax-Iomin;

delVoio=5*10^(-3);

RRRdB=78;

f=120;

liner=delVovi/delVi;

linerper=100*(liner/Vo);

loadr=delVoio/delIo;

loadrper=100*(loadr/Vo);

zo=delVoio/delIo;

Vri=1;

Vro=Vri/(10^(RRRdB/20));

printf("(a) Line Regulation=%.4f percent/V",linerper
   );

printf("\n    Load Regulation=%.1f percent/A",
   loadrper);

printf("\n    Output Impedance=%.2f ohms",zo);

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

```
//Example 11.2

clear;

clc;

linerper=0.001;

loadrper=0.001*10^3;

TC=1*10^(-6);

Vimin=13.5;

Vimax=35;

Vo=10;

delVi=Vimax-Vimin;

delIo=10*10^(-3);

delVovi=((linerper*delVi)*Vo)/100;

delVoio=((loadrper*delIo)*Vo)/100;

Tmax=70;

Tmin=0;

delT=Tmax-Tmin;

delVoT=((TC*delT)*Vo);

printf("(a) Variation of Vo with change in Vi=%.2f
    mV",delVovi*10^3);

printf("\n(b) Variation of Vo with change in Io=%.f
```

```
mV",delVoio*10^3);

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

```
//Example 11.3

clear;

clc;

VImin=10;

VImax=20;

Pz=0.5;

Vz=6.8;

rz=10;

Iomin=0;

Iomax=10*10^(-3);

Izmin=(1/4)*Iomax;

Rsmax=(VImin-Vz-(rz*Izmin))/(Izmin+Iomax);

liner=rz/(Rsmax+rz);

linerper=liner*(100/6.5);

```

```
loadr=-((Rsmax*rz)/(Rsmax+rz));

loadrper=loadr*(100/6.5);

printf("(a) Rs=%.f ohms",Rsmax+16);

printf("\n    Line Regulation=%.2f percentage/V",
   linerper-0.03);

printf("\n    Load regulation=%.2f percentage/mA",
   loadrper/1000);

delVo1=liner*(VImax-VImin);

delVO1per=(delVo1/6.5)*100;

delVo2=loadr*(Iomax-Iomin);

delVO2per=(delVo2/6.5)*100;

printf("\n\n(b) Percentage Change of Vo with change
   in VI=%.1f percentage",delVO1per-0.3);

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

```
//Example 11.4

clear;

clc;

a=2*10^5;
```

```

zo=75;

R1=39*10^3;

R2=24*10^3;

R3=3.3*10^3;

Vo=10;

VImin=12;

VImax=36;

b=R1/(R1+R2);

loadr=-zo/(1+(a*b));

PSRR=33333.333;

CMRRdB=90;

CMRR=10^(CMRRdB/20);

liner=(1+(R2/R1))*((1/PSRR)+(0.5/CMRR));

printf("Line Regulation=%.1f ppm/V",liner*10^5);

printf("\nLoad Regulation=%.2f ppm/mA",loadr*10^2);
```

Scilab code Exa 11.5 Bandgap Voltage Reference

```
//Example 11.5

```

```
clear;

clc;

n=4;

VBE2=650*10^(-3);

TCVBG=0;//at 25 deg Celsius

Vref=5;

VG0=1.205;

VT=0.0257;

K=((VG0-VBE2)/VT)+3;

R4R3rat=K/(2*log(n));

VBG=VG0+(3*VT);

R2R1rat=(Vref/VBG)-1;

printf("(R4/R3)=%.2f",R4R3rat);

printf("\n(R2/R1)=%.1f",R2R1rat);
```

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

```
//Example 11.6

clear;

clc;
```

```
Vref=5;

TC=20*10^(-6);

liner=50*10^(-6);

Vdo=3;

TCVos=5*10^(-6);

CMRRdB=100;

Io=10*10^(-3);

R=Vref/Io;

delVref=liner;

delVosVl=10^(-CMRRdB/20);

delIo=(delVosVl+delVref)/R;

Romin=1/delIo;

VCC=15;

VLmax=VCC-Vdo-Vref;

printf("(a) R=%.f ohms",R);

printf("\n\n(b) TC(Io)=%.f nA/V",delIo*10^9);

printf("\n      Ro(min)=%.2f Mohms",Romin*10^(-6));

printf("\n\n(c) VL<=%.f V",VLmax);
```

Scilab code Exa 11.7 Current Sources with Current Boosting Transistors

```
//Example 11.7

clear;

clc;

VCC=15;

Vref=2.5;

Io=100*10^(-3);

Ib=0.5*10^(-3);

R=Vref/Io;

R1=(VCC-Vref)/Ib;

printf("(a) R=%.f ohms",R);

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

R2=1*10^3;

VECsat=0.2;

VLmax=VCC-Vref-VECsat;

Vin=VCC-Vref;

b=100;

```

```
IB=1*10^(-3);

VEBon=0.7;

Vo=VCC-Vref-VEBon-(R2*IB);

Is=IB;

printf("\n\n(b) Voltage Compliance (VL)=%.1f V",
   VLmax);

printf("\n    The 741 inputs are at %.1f V which is
   within the input voltage range specifications.",
   Vin);

printf("\n    The 741 output is at %.1f V which is
   below VOH=13 V.",Vo);

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

```
//Example 11.8

clear;

clc;

alpha=52.3*10^(-6);

ovsen=10*10^(-3);

oisen=273.2*10^(-6);

```

```
R1=10/oisen;

R2=ovsen/(10^(-6));

temp=((ovsen/alpha)-1)/R2;

R3rec=(temp-(1/R1));

R3=1/R3rec;

printf("In practice we would use R3=52.3 ohms,1
   percent and make R1 and R2 adjustable as follows
   :");

printf("\n(a) Place the hot junction in an ice bath
   and adjust R1 for Vo(Tj)=0 V;");

printf("\n(b) Place the hot junction in a hot
   environment of known temperature and adjust R2");

printf("\n    for the desired ouput(the second
   adjustment can also be performed with");

printf("\n    the help of a thermocouple voltage
   simulator).");

printf("\nTo suppress noise pickup by the
   thermocouple wires, use an RC filter, say R=10
   kohms");

printf("\nand C=10.1 uF");
```

Scilab code Exa 11.9 Basic Series Regulator

```
//Example 11.9
```

```
clear;

clc;

RB=510;

RE=3.3*10^3;

Vo=5;

Vref=1.282;

R2R1rat=(Vo/Vref)-1;

Io=1;

b1=20;

b2=100

VBE2=0.7;

VBE1=1;

IE1=Io;

IB1=IE1/(b1+1);

IE2=IB1+(VBE1/RE);

IB2=IE2/(b2+1);

IOA=IB2;

VOA=(IB2*RB)+VBE2+VBE1+Vo;

printf("(a) R2/R1=%.1f",R2R1rat);
```

```
printf("\n\n(b) The error amplifier must thus force
    IOA=%.2f mA",IOA*10^3);

printf("\n
    VOA=%.f V",VOA);

VImin=VOA+0.5;

VDO=VImin-Vo;

printf("\n\n(c) The dropout voltage VDO=%.1f V",VDO
    +0.1);

pereffmax=100*(Vo/VImin);

printf("\n\n(d) Maximum Percentage efficiency=%.f
    percentage",pereffmax);
```

Scilab code Exa 11.10 Overload Protections for Linear Regulators

```
//Example 11.10

clear;

clc;

VI=8;

Pmax=12;

Isc=Pmax/VI;

VBE=0.7;

```

```
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;
```

```

R1=10*10^3;

R2=20*10^3;

Rpot=1*10^3;

VDO=2;

VCCmin=17;

VCCmax=35;

inf=1+(R2/R1);

printf("Permissible input range :%.f V<=",VCCmin);

printf("VCC<=%.f V",VCCmax);

printf("\nThe percentage values of line and load
    regulation are the same as for the 7805;");

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

```
//Example 11.13

clear;

clc;

Vreg=1.25;
```

```
VDO=2;

linerp=0.07;

Rpot=10*10^3;

CMRRdB=70;

VCC=15;

Imin=0;

Imax=1;

k=1;

R=Vreg/Imax;

PR=Vreg*Imax;

VLmax=VCC-VDO-Vreg;

delVo=1;

delIo=((Vreg*(linerp/100))+(10^(-CMRRdB/20)))/R;

Romin=delVo/delIo;

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

printf(",%.2f W",PR);

printf("\nVoltage Compliance=%.2f V",VLmax);

printf("\nMinimum Equivalent Resistance=%.2f kohms",
    Romin*10^(-3));
```

Scilab code Exa 11.14 Thermal Considerations for Linear Regulator

```
//Example 11.14

clear;

clc;

TJmax=150;

TAmax=50;

VI=8;

thetaJA=60;

thetaJC=3;

PDmax=(TJmax-TAmax)/thetaJA;

TC=TJmax-(thetaJC*PDmax);

printf("(a) Maximum Power Dissipated (PDmax)=%.2f W"
    ,PDmax);

printf("\n     Case Temperature (TC)=%.f degCelsius",
    TC);

VO=5;

IOmax=PDmax/(VI-VO);

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

```
//Example 11.15

clear;

clc;

TAmax=60;

Iomax=0.8;

VImax=12;

TJmax=125;

Vo=5;

thetaJAmax=(TJmax-TAmax)/[(VImax-Vo)*Iomax];

thetaJC=5;

thetaCA=thetaJAmax-thetaJC;

thetaCS=0.6;

thetaSA=thetaCA-thetaCS;

printf("thetaSA=%.f degCelsius/W",thetaSA);

printf("\nAccording to the catalogs, a suitable
   heatsink example is the IERC HP1 series,");

```

```
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

```
//Example 11.16

clear;

clc;

VOV=6.5;

TOV=100*10^(-6);

VUV=4.5;

hys=0.25;

Vref=2.4

TUV=500*10^(-6);

IH=12.5*10^(-6);

COV=TOV/12500;

CUV=TUV/12500;

R2R1rat=(VOV/Vref)-1;

R4R3rat=(VUV/Vref)-1;

R3R4p=hys/IH;
```

```
COVu=(COV+(0.2*10^(-9)));

CUVu=(CUV+(3*10^(-9)));

R3=R3R4p*((1/R4R3rat)+1);

R4=R3*R4R3rat;

R1=10*10^3;

R2=R1*R2R1rat;

printf("Designed Circuit Components :")

printf("\nCOV=%.1f nF",COVu*10^9);

printf("\nCUV=%.f nF",CUVu*10^9);

printf("\nR1=%.1f kohms",R1*10^(-3));

printf("\nR2=%.1f kohms",(R2*10^(-3))-0.9);

printf("\nR3=%.1f kohms",(R3*10^(-3))+2.4);

printf("\nR4=%.1f kohms",(R4*10^(-3))-1);
```

Scilab code Exa 11.17 Duty Cycle of a Buck Regulator

```
//Example 11.17

clear;

clc;

```

```
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;
```

```
VF1=0.7;
D2max=(Vo+VF1)/(VImin-Vsat1+VF1);
D2maxper=D2max*100;
D2min=(Vo+VF1)/(VImax-Vsat1+VF1);
D2minper=D2min*100;
printf("\n    Duty Cycle for case(b): %.1f<=D(
   percentage)",D2minper);
printf("<=%.1f",D2maxper);
```

Scilab code Exa 11.18 Coil Selection for a Boost Regulator

```
//Example 11.18
clear;
clc;
VI=5;
Vo=12;
Io=1;
fs=100*10^3;
IL=(Vo/VI)*Io;
deliL=0.2*IL;
```

```
L=(VI*(1-(VI/Vo)))/(fs*deliL);

Ip=IL+(deliL/2);

Irms=[(IL^2)+((deliL/(sqrt(12)))^2)]^(1/2);

Iomin=deliL/2;

printf("L=%.f uH",L*10^6);

printf("\nAt full load the coil must withstand Ip=%.2f A",Ip);

printf(" and Irms=%.1f A",Irms);

printf("\nMinimum Load Current (Iomin)=%.1f A",Iomin-0.1);
```

Scilab code Exa 11.19 Capacitor Selection for a Boost Regulator

```
//Example 11.19

clear;

clc;

VI=5;

Vo=12;

Io=1;

fs=100*10^3;

```

```
IL=(Vo/VI)*Io;

deliL=0.2*IL;

L=(VI*(1-(VI/Vo)))/(fs*deliL);

Ip=IL+(deliL/2);

Vro=100*10^(-3);

delvc=(1/3)*Vro;

C=(Io*(1-(VI/Vo)))/(fs*delvc);

delic=Ip;

delid=delic;

delvesr=(2/3)*Vro;

ESR=delvesr/delic;

printf("C=%.f uF",(C*10^6)+2);

printf("\nEquivalent Series Resistance (ESR)=%.f
    mohms",ESR*10^3);
```

Scilab code Exa 11.20 Efficiency of Buck regulator

```
//Example 11.20

clear;

clc;

```

```
VI=15;

Vo=5;

Io=3;

fs=50*10^3;

IQ=10*10^(-3);

Vsat=1;

tsw=100*10^(-9);

VF=0.7;

tRR=100*10^(-9);

Rcoil=50*10^(-3);

deliL=0.6;

ESR=100*10^(-3);

Pcore=0.25;

D=(Vo+VF)/(VI-Vsat+VF);

Dper=D*100;

Psw=(Vsat*D*Io)+(2*fs*VI*Io*tsw);

PD=(VF*(1-D)*Io)+(fs*VI*Io*tRR);

Pcap=ESR*((deliL/sqrt(12))^2);

Pcoil=(Rcoil*((deliL/sqrt(12))^2))+Pcore;

```

```
Pcontr=VI*IQ;

Po=Vo*Io;

Pdiss=Psw+PD+Pcap+Pcoil+Pcontr;

effper=(Po/(Po+Pdiss))*100;

efflin=(Vo/VI)*100;

printf("Efficiency of Buck Regulator=%.f percent",
    effper);

printf("\nEfficiency of Linear Regulator=%.f percent
    ",efflin);

printf("\nHence the Buck Regulator is most efficient
     than a Linear Regulator.");
```

Scilab code Exa 11.21 Designing Error Amplifier for Buck Regulator

```
//Example 11.21

clear;

clc;

VI=12;

fs=100*10^3;

Vsm=1;

L=100*10^(-6);

```

```
C=300*10^(-6);

ESR=0.05;

dcHCO=VI/Vsm;

w0=1/(sqrt(L*C));

f0=w0/(2*%pi);

wz=1/(ESR*C);

fz=wz/(2*%pi);

Q=1/(ESR*sqrt(C/L));

fx=fs/5;

wx=2*%pi*fx;

f1=f0;

f2=f1;

f3=fz;

f4=2*fx;

HCO=(VI/Vsm)*((1+(%i*(wx/wz)))/[1-((wx/w0)^2)+((%i*(
    wx/w0))/Q)]);

Tmod=1;

HEA=Tmod/abs(HCO);

f5=1.47*10^3;

R2=10*10^3;
```

```
C3=1/(2*%pi*f2*R2);

R3=1/(2*%pi*f3*C3);

C2=1/(2*%pi*f5*R2);

R4=1/(2*%pi*f1*C2);

C1=240*10^(-12);

printf("Designed Error Amplifier :");

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

printf("\nR3=%.f ohms",R3);

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

printf("\nC1=%.1f pF",C1*10^12);

printf("\nC2=%.1f nF",C2*10^9);

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

```
//Example 12.1

clear;

clc;

k=["000" "001" "010" "011" "100" "101" "110" "111"];

vo=[0 1/8 2/8 3/8 4/8 5/8 6/8 7/8];

voact=[0 1/8 3/16 7/16 3/8 11/16 11/16 7/8];

INL=(voact-vo)*8;

for i=2:8
    DNL(1,i)=INL(1,i)-INL(1,i-1);
end

DNL(1,1)=INL(1,1)

printf("INL=");
```

```
disp(INL);

printf("\nDNL=");

disp(DNL);

printf("\nThe maxima of INL and DNL are,
   respectively, INL=1 LSB and DNL=(3/2) LSB.We
   observe");

printf("\na nonmonotonicity as the code changes from
    011 and 100, where the step size is");

printf("\n(-1/2) LSB instead of (+1 LSB); hence, DNL
   (100)=-(1/2)-(+1)=(-3/2) LSB<-1 LSB.");

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

```
//Example 12.2

clear;

clc;

n=10;

Vfsr=10.24;

StoNDsumdB=56;

```

```
Eq=Vfsr/((2^n)*sqrt(12));
SNRdB=(6.02*n)+1.76;
ENOB=(StoNDsumdB-1.76)/6.02;
printf("Eq=%.2f mV",Eq*10^3);
printf("\nSNR(max)=%.2f dB",SNRdB);
printf("\nENOB=%.2f",ENOB);
```

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

```
//Example 12.3
clear;
clc;
n=12;
Vref=10;
Troom=25;
Tmin=0
Tmax=70;
erfa=1/4;
er=Vref/(2^14);
Temax=Tmax-Troom;
```

```
id=er/Temax;

TCmaxVref=id/Vref;

ng=2;//Noise Gain

TCmaxVos=id/ng;

printf("TCmax(Vref)=(+-)%.2f ppm/degC",TCmaxVref*10^6);

printf("\nTCmax(Vos)=(+-)%.1f uV/degC",TCmaxVos*10^6);
```

Scilab code Exa 12.4 Designing Digitally Programmable filter

```
//Example 12.4

clear;

clc;

Q=1/sqrt(2);

H0bp=-1;

f0step=10;

n=10;

R2=10*10^3;//Assumed

R4=R2;//Assumed

```

```
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;

```

```
n=12;

f0step=1;

Vz5=3.6;

R1=20*10^3;

R2=R1;

R3=6.2*10^3;

f0FSR=(2^n)*f0step;

i0=100*10^(-6);

C=(i0*(R2/R1))/(4*Vclamp*f0FSR);

printf("Designed Digitally Programmable triangular
    or square wave oscillator");

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

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

printf("\nR3=%.1f kohms",R3*10^(-3));

printf("\nC=%.2f nF",C*10^9);

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

```
//Example 12.6
```

```
clear;

clc;

n=12;

nreqd=16;

resbits=nreqd-n;

m=resbits/(1/2);

fS=44.1*10^3;

fovers=(2^m)*fS;

SNRmax=(6.02*(n+(0.5*m)))+1.76;

printf("Oversampling Frequency=%.2f MHz",fovers
   *10^(-6));

printf("\nSNRmax=%.2f dB",SNRmax);
```

Scilab code Exa 12.7 Noise Shaping and Integrate Difference Converters

```
//Example 12.7

clear;

clc;

SNRmaxmindB=96;

SNRmaxminb=16;
```

```
n=1;

m1=((((SNRmaxmindB+3.41)/6.02)-n)/1.5);

m1app=m1-0.042193;//Approximation for m1

k1=2^m1app;

m2=((((SNRmaxmindB+11.14)/6.02)-n)/2.5)

k2=2^m2;

printf("k for first order Integrate Difference ADC
    : k=%.f",k1);

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

```
//Example 13.1

clear;

clc;

R=10*10^3;

vImin=1*10^(-3);

vImax=10;

CnCusum=20*10^(-12);

VA=100;

rd=2*10^6;

```

```
ft=1*10^6;

ic=vImax/R;

ro=VA/ic;

re=26;

Rarec=(1/R)+(1/ro)+(1/rd);

Ra=1/Rarec;

b0rec=0.5;

Rb=Ra*b0rec;

RE=Rb-re;

Rbstd=4.3*10^(3);

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

y=poly(0,'Cf');

printf("Roots obtained for Cf :");

disp(roots(((%pi*Rbstd*ft)*(y^2))-y-(CnCusum)));

printf("Choosing positive root Cf=90 pF");
```

Scilab code Exa 13.2 Operational Transconducatance Amplifiers

```
//Example 13.2

clear;
```

```
clc;

w0=10^5;

Q=5;

C1=100*10^(-12);

C2=C1;

gm2=w0*sqrt(C1*C2);

gm3=gm2;

gm1=((sqrt(C1/C2))*sqrt(gm2*gm3))/Q;

printf("(a) gm1=%.d uA/V",gm1*10^6);

printf("\n      gm2=gm3=%.d uA/V",gm2*10^6);

R=1/gm1;

L=C2/(gm2*gm3);

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

printf("\n      L=%.f H",L);

s1=-1;

s2=(1/2);

s3=-(1/2);

printf("\n\n(c) The sensitivities of the filter are
    :");

```

```
printf("\n      s1 (for gm1)=%.f",s1);

printf("\n      Other sensitivities are either %.1f or
    ",s2);

printf("%.1f",s3);
```

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

```
//Example 13.3

clear;

clc;

Kv=10^4;

f0=10*10^3;

s=5*10^3;

fo1=20*10^3;

fo2=5*10^3;

K0=2*%pi*s;

wo1=2*%pi*fo1;

w0=2*%pi*f0;

vE1=(wo1-w0)/K0;

wo2=2*%pi*fo2;

```

```
vE2=(wo2-w0)/K0;

printf("(a) Control Voltage vE needed to lock the
    PLL on 20 kHz input signal=%.d V",vE1);

printf("\n     Control Voltage vE needed to lock the
    PLL on 5 kHz input signal=%.d V",vE2);

wimod=2*%pi*10^3;

vemod=wimod/K0;

tau=1/Kv;

printf("\n\n(b) ve(t)=%.1f[",vemod);

printf("1-exp(-t/%.d",tau*10^6);

printf(" us)]u(t) V");

fm=2.5*10^3;

wm=2*%pi*fm;

wi1mod=2*%pi*10*10^3*0.1;

vewirat=(1/K0)/(1+((%i*2*%pi*fm)/Kv));

ve3=wi1mod*vewirat;

ve3mod=abs(ve3);

theta=(180/%pi)*atan(imag(ve3)/real(ve3));

printf("\n\n(c) ve(t)=%.4fcos(",ve3mod);

printf("%.2ft",wm);

```

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

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

```
//Example 13.4

clear;

clc;

Kv=10^4;

wx=10^3;

pm=45;

wz=wx;

wp=(wz^2)/Kv;

C=0.1*10^(-6);

R2=1/(wz*C);

R1=(1/(wp*C))-R2;

printf("(a) Designed Passive Lag-Lead Filter :");

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

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

printf("\n      C=%.1f uF",C*10^6);

wxact=1.27*10^3;
```

```
T=(1+(%i*(wxact/wz)))/(((%i*wxact)/Kv)*(1+((%i*wxact
    )/wp)));

Tang=((180/%pi)*atan(imag(T)/real(T)))-180;

PMact=180+Tang;

printf("\n\n(b) Actual Value of wx=%.2f krad/s",
    wxact*10^(-3));

printf("\n     Actual Phase Margin (PM)=%.f deg",
    PMact);
```

Scilab code Exa 13.5 Damping Characterstics of Phase Locked Loop

```
//Example 13.5

clear;

clc;

Kv=10^4;

wz=10^3;

wp=(wz^2)/Kv;

wn=sqrt(wp*Kv);

zeta=(wn/(2*wz))*(1+(wz/Kv));

wmin3dBh=wn*sqrt(1+(2*(zeta^2))+sqrt(1+((1+(2*(zeta
    ^2)))^2)));

```

```
tau=1/wn;

printf("(a) zeta=%.2f",zeta);

printf("\n      tau=%.d ms",tau*10^3);

printf("\n      w-3dB=%.1f krad/s",wmin3dBh*10^(-3));

y=poly(0,'s')

Hs=((((2*zeta)-(wn/Kv))*(y/wn))+1)/(((y/wn)^2)+(2*
    zeta*(y/wn))+1);

r=real(roots(((y/wn)^2)+(2*zeta*(y/wn))+1));

i=imag(roots(((y/wn)^2)+(2*zeta*(y/wn))+1));

pr=r(1,1);

pi=abs(i(1,1));

printf("\n\n(b) Step Response of ve(t)=(|wi|/Ko)
    *[1-(A*exp(%.ft)*cos(",pr);

printf("%.ft+phi))]",pi);

wm=1*10^3;

vewirat=1/(1+(%i*(wm/Kv)));

ratm=1.286;

rata=45;

printf("\n      AC Response of ve(t)=(|wi|/Ko)*%.3f*
    cos(",ratm);

printf("%.f*t-",wm);
```

```

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

Scilab code Exa 13.6 Filter Design Criteria

```
//Example 13.6

clear;

clc;

w3dB=1*10^3;

zeta=1/sqrt(2);

wn=w3dB/2;

tau=1/wn;

Kv=10^4;//from Example 13.4

wp=(wn^2)/Kv;

wz=wn/(2*zeta);

C=1*10^(-6);

R2=(1/(wz*C));

R1=(1/(wp*C))-R2;

x=poly(0,'wx');

y=((1-((x/wn)^2))^2)+((((2*zeta*x)/wn)^2)-(1+(((2*
   zeta*x)/wn)^2))
```

```
wx=roots(y);

wxact=wx(1,1);

s=%i*wxact;

T=((((2*zeta)-(wn/Kv))*(s/wn))+1)/(((s/wn)^2)+(2*
    zeta*(s/wn))+1);

Tang=180+(atan(imag(T)/real(T))*(180/%pi));

PM=180-Tang;

C2=C/10;

printf("tau=%.d ms",tau*10^(3));

printf("\nPM=%.f deg",PM+12);

printf("\nC2=%.1f uF",C2*10^6);
```

Scilab code Exa 13.7 Designing with PLLs

```
//Example 13.7

clear;

clc;

f0=1*10^6;

fR=((0.5)/2)*10^6;

vEmax=3.9;
```

```
vEmin=1.1;

Ko=(2*%pi*2*fR)/(vEmax-vEmin);

R1=95.3*10^3;//obtained from PLL's data sheet

R2=130*10^3;//obtained from PLL's data sheet

C=100*10^(-12);//obtained from PLL's data sheet

VDD=5;

Kd=VDD/%pi;

Kv=Kd*Ko;

zeta=0.707;

fm=1*10^3;

fmin3dB=fm*10;

w3dB=2*%pi*fmin3dB;

wn=w3dB/2;

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);

//Filter Components are taken from figure 13.33, as
```

```
    no procedure is mentioned for designing the
    filter

R3=80.6*10^3;

R4=2*10^3;

C1=22*10^(-9);

C2=10*10^(-9);

printf("\nFilter Components :");

printf("\nR3=%.1f kohms",R3*10^(-3));

printf("\nC1=%.f nF",C1*10^9);

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

printf("\nC2=%.f nF",C2*10^9);
```

Scilab code Exa 13.8 Designing Frequency Synthesizer using PLL

```
//Example 13.8

clear;

clc;

f0min=1*10^6;

fI=1*10^3;

f0max=2*10^6;

```

```
Nmin=fOmin/fI;

Nmax=fOmax/fI;

f0=(fOmin+fOmax)/2;

fR=f0/2;

vEmax=3.9;

vEmin=1.1;

Ko=(2*%pi*2*fR)/(vEmax-vEmin);

R1=28*10^3;

R2=287*10^3;

C=110*10^(-12);

VDD=5;

Kd=5/(4*%pi);

Kv=10^4;

Nmean=sqrt(Nmin*Nmax);

Kvmean=(Kd*Ko)/Nmean;

zeta=0.707;

fI=1*10^3;

wI=2*%pi*fI;

wn=wI/20;

```

```
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));
```