# Scilab Textbook Companion for Modern Power System Analysis by D. P. Kothari And I. J. Nagrath[1]

Created by
Brahmesh Jain S D
B.E
Electrical Engineering
Sri Jayachamarajendra College Of Engineering
College Teacher
Prof. R S Anandamurthy
Cross-Checked by
TechPassion

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:** Modern Power System Analysis

**Author:** D. P. Kothari And I. J. Nagrath

**Publisher:** Tata McGraw - Hill Education, New Delhi

**Edition:** 3

**Year:** 2003

**ISBN:** 0070494894

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

# Introduction

**Scilab code Exa 1.1** Example 1

```
//Chapter 1
//Example 1.1
//page 5
clear;clc;
fl=760e3;
pf=0.8;
lsg=0.05;
csg=60;
depre=0.12;
hpw=48;
lv=32;
hv=30;
pkwhr=0.10;

md=fl/pf;
printf('Maximum Demand= %.1f kVA \n\n',md/1000);

//calculation for tariff (b)

printf('Loss in switchgear=%.2f %% \n\n',lsg*100);
input_demand=md/(1-lsg);
```

```
input_demand=input_demand/1000;
cost_sw_ge=input_demand*60;
depreciation=depre*cost_sw_ge;
fixed_charges=hv*input_demand;
running_cost=input_demand*pf*hpw*52*pkwhr;//52 weeks per year
total_b=depreciation + fixed_charges + running_cost;
printf('Input Demand= %.1f kVA \n\n',input_demand);
printf('Cost of switchgear=Rs %d\n\n',cost_sw_ge);
printf('Annual charges on depreciation=Rs %d \n\n',depreciation);
printf('Annual fixed charges due to maximum demand corresponding to triff(b)=Rs %d \n\n',fixed_charges);
printf('Annual running cost due to kWh consumed=Rs %d \n\n',running_cost);
printf('Total charges/annum for tariff(b) = Rs %d\n\n',total_b)

//calculation for tariff (a)
input_demand=md;
input_demand=input_demand/1000;
fixed_charges=lv*input_demand;
running_cost=input_demand*pf*hpw*52*pkwhr;
total_a=fixed_charges + running_cost;
printf('maximum demand corresponding to tariff(a) = %.f kVA \n\n',input_demand);
printf('Annual fixed charges=Rs %d \n\n',fixed_charges);
printf('Annual running charges for kWh consumed = Rs %d \n\n',running_cost);
printf('Total charges/annum for tariff(a) = Rs %d \n\n',total_a);
if(total_a > total_b)
    printf('Therefore, tariff(b) is economical\n\n\n');
else
    printf('Therefore, tariff(a) is economical\n\n\n
');
```

**Scilab code Exa 1.3** Example 3

```
//Chapter 1
//Example 1.3
//page 7
clear;clc;
md=25;
lf=0.6;
pcf=0.5;
puf=0.72;

avg_demand=lf*md;
installed_capacity=avg_demand/pcf;
reserve=installed_capacity-md;
daily_ener=avg_demand*24;
ener_inst_capa=installed_capacity*24;
max_energy=daily_ener/puf;

printf('Average Demand= %.2f MW \n\n',avg_demand);
printf('Installed capacity= %.2f MW \n\n\',installed_capacity);
printf('Reserve capacity of the plant= %.2f MW \n\n',reserve);
printf('Daily energy produced= %d MWh \n\n',daily_ener);
printf('Energy corresponding to installed capacity per day= %d MWh \n\n',ener_inst_capa);
printf('Maximum energy that could be produced = %d MWh/day \n\n',max_energy);
```

**Scilab code Exa 1.4** Example 4

```
//Chapter 1
//Example 1.2
//page 6
clear;clc;
md=20e3;
unit_1=14e3;
unit_2=10e3;
ener_1=1e8;
ener_2=7.5e6;
unit1_time=1;
unit2_time=0.45;

annual_lf_unit1=ener_1/(unit_1*24*365);
md_unit_2=md-unit_1;
annual_lf_unit2=ener_2/(md_unit_2*24*365);
lf_unit_2=ener_2/(md_unit_2*unit2_time*24*365);
unit1_cf=annual_lf_unit1;
unit1_puf=unit1_cf;
unit2_cf=ener_2/(unit_2*24*365);
unit2_puf=unit2_cf/unit2_time;
annual_lf=(ener_1+ener_2)/(md*24*365);


printf('Annual load factor for Unit 1 = %.2f %% \n\n',annual_lf_unit1*100);
printf('The maximum demand on Unit 2 is %d MW \n\n',md_unit_2/1000);
printf('Annual load factor for Unit 2 = %.2f %% \n\n',annual_lf_unit2*100);
printf('Load factor of Unit 2 for the time it takes
the load= %.2f %% \n\n',lf_unit_2*100);
printf('Plant capacity factor of unit 1 = %.2f %% \n\n',unit1_cf*100);
printf('Plant use factor of unit 1 = %.2f %% \n\n',unit1_puf*100);
printf('Annual plant capacity factor of unit 2 = %.2f %% \n\n',unit2_cf*100);
printf('Plant use factor of unit 2 = %.2f %% \n\n',unit2_puf*100);
printf('The annual load factor of the total plant = %.2f %% \n\n',annual_lf*100);
```

**Scilab code Exa 1.5** Example 5

```
//Chapter 1
//Example 1.2
//page 6
clear;clc;

c1_md_6pm=5;      c1_d_7pm=3;     c1_lf=0.2;
c2_md_11am=5;     c2_d_7pm=2;     c2_avg_load=1.2;
c3_md_7pm=3;                      c3_avg_load=1;

md_system=c1_d_7pm + c2_d_7pm + c3_md_7pm;
sum_mds=c1_md_6pm + c2_md_11am + c3_md_7pm;
df=sum_mds/md_system;

printf('Maximum demand of the system is %d kW at 7p.m \n',md_system);
printf('Sum of the individual maximum demands = %d kW \n',sum_mds);
printf('Diversity factor= %.3f \n\n',df);

```

```
c1_avg_load=c1_md_6pm*c1_lf;
c2_lf=c2_avg_load/c2_md_11am;
c3_lf=c3_avg_load/c3_md_7pm;

printf('Consumer1 -->\t Avg_load= %.2f kW \t LF= %.1f %% \n',c1_avg_load,c1_lf*100);
printf('Consumer2 -->\t Avg_load= %.2f kW \t LF= %.1f %% \n',c2_avg_load,c2_lf*100);
printf('Consumer3 -->\t Avg_load= %.2f kW \t LF= %.1f %% \n\n',c3_avg_load,c3_lf*100);

avg_load=c1_avg_load + c2_avg_load + c3_avg_load;
lf=avg_load/md_system;

printf('Combined average load = %.1f kW \n',avg_load);
printf('Combined load factor= %.1f %% \n\n',lf*100);
```

# Chapter 2

# Inductance and Resistance of Transmission Lines

**Scilab code Exa 2.1** self GMD Calculation

```
//Chapter 2
//Example 2.1
//page 56
//To find GMD of the conductor
//From the given the text book,leaving out the factor of "r",we have the seven possible distances
clear;clc;
D1=0.7788*2*2*(2*sqrt(3))*4*(2*sqrt(3))*2;
//since there are 7 identical conductors,the above products remains same dor all D's
D2=D1;
D3=D1;
D4=D1;
D5=D1;
D6=D1;
D7=D1;
Ds=(D1*D2*D3*D4*D5*D6*D7)^(1/(7*7));
printf("\n GMD of the conductor is %0.4fr",Ds);
```

**Scilab code Exa 2.2** Reactance Of ACSR conductors

```
//Chapter 2
//Example 2.2
//page 57
//To find reactance of the conductor
clear;clc;
f=50; //frequency
D=5.04; //diameter of the entire ACSR
d=1.68; //diameter of each conductor
Dsteel=D-2*d; //diameter of steel strand
//As shown in fig
D12=d;
D13=(sqrt(3)*d);
D14=2*d;
D15=D13;
D16=D12;
//neglecting the central sttel conductor,we have the 6 possibilities
D1=(0.7788*d)*D12*D13*D14*D15*D16;
//we have total of 6 conductors,hence
D2=D1;
D3=D1;
D4=D1;
D5=D1;
D6=D1;
Ds=(D1*D2*D3*D4*D5*D6)^(1/(6*6));//GMR;
//since the spacing between lines is 1m=100cm
l=100;
L=0.461*log10(l/Ds); //Inductance of each conductor
Ll=2*L; // loop inductance
Xl=2*%pi*f*Ll*10^(-3);//reactance of the line
```

```
printf("\n\nInductance of each conductor=%0.4f mH/km\n\n",L);
printf("Loop Inductance=%0.4f mH/km\n\n",Ll);
printf("Loop Reactance=%f ohms/km\n\n",Xl);
```

**Scilab code Exa 2.3** Inductance Of Composite Conductor Lines

```
//Chapter 2
//Example 2.3
//page 58
//To find inductance of each side of the line and that of the complete line
clear;clc;
//to find mutual GMD
D14=sqrt(8*8+2*2);
D15=sqrt(8*8+6*6);
D24=sqrt(8*8+2*2);
D25=sqrt(8*8+2*2);
D34=sqrt(8*8+6*6);
D35=sqrt(8*8+2*2);
//sixth root of six mutual distances
Dm=(D14*D15*D24*D25*D34*D35)^(1/6);//mutual GMD between lines

//to find GMR of Side A conductors
D11=0.7788*2.5*10^(-3);
D22=D11;
D33=D11;
D12=4;
D21=D12;
D13=8;
D31=8;
D23=4;
```

```
D32=D23;
//ninth root nine distances in Side A
Da=(D11*D12*D13*D21*D22*D23*D31*D32*D33)^(1/9);

//to find GMR of Side A conductors
D44=0.7788*5*10^(-3);
D45=4;
D54=D45;
D55=D44;
//fourth root of four distances in Side B
Db=(D44*D45*D54*D55)^(1/4);

La=0.461*log10(Dm/Da);//inductance line A
Lb=0.461*log10(Dm/Db);//inductance line B

L=La+Lb; //loop inductance

printf("\n\nMutual GMD between lines = %0.4f m\n\n",Dm);
printf("GMR of Side A conductors = %0.4f m\n\n",Da);
printf("GMR of Side B conductors = %0.4f m\n\n",Db);
printf("Inductance of line A = %0.4f mH/km\n\n",La);
printf("Inductance of line B = %0.4f mH/km\n\n",Lb);
printf("Loop Inductance of the lines = %0.4f mH/km\n\n",L);
```

**Scilab code Exa 2.5** VoltageDrop and FluxLinkage Calculations

```
//Chapter 2
//Example 2.5
//page 63
//To find flux linkages with neutral and voltage induced in neutral
```

```
//To find voltage drop in each of three-phase wires

clear;clc;
Ia=-30+%i*50;
Ib=-25+%i*55;
Ic=-(Ia+Ib);

//(a) to find flux linkages with neutral and voltage induce in it
Dan=4.5;
Dbn=3;  //from figure
Dcn=1.5;
Phi_n=2*10^(-7)*(Ia*log(1/Dan)+Ib*log(1/Dbn)+Ic*log(1/Dcn));
Vn=%i*2*%pi*50*Phi_n*15000; //voltage induced for 15 km long TL
Vn=abs(Vn) ;
printf("\nFlux linkages of the neutral wire = %f Wb-T/m\n\n",Phi_n);
printf("Voltage induced in the neutral = %d\n\n",Vn);

//(b) to find voltage drop in each phase
Phi_a=2*10^(-7)*(Ia*log(1/(0.7788*0.005))+Ib*log(1/1.5)+Ic*log(1/3));
Phi_b=2*10^(-7)*(Ib*log(1/(0.7788*0.005))+Ia*log(1/1.5)+Ic*log(1/1.5));
Phi_c=2*10^(-7)*(Ic*log(1/(0.7788*0.005))+Ib*log(1/1.5)+Ia*log(1/3));

delta_Va=%i*2*%pi*50*Phi_a*15000; //like we did for neutral voltage
delta_Vb=%i*2*%pi*50*Phi_b*15000;
delta_Vc=%i*2*%pi*50*Phi_c*15000;

printf("The Voltage drop of phase a(in volts) =");disp(delta_Va);
printf("\n\nThe Voltage drop of phase b(in volts) ="
);disp(delta_Vb);
printf(”\n\nThe Voltage drop of phase c(in volts) =”);disp(delta_Vc);
```

**Scilab code Exa 2.6** Mutual Inductance Calculation

```
//Chapter 2
//Example 2.6
//page 65
//To find mutual inductance between power line and telephone line and voltage induced in telephone line

clear;clc;
D1=sqrt(1.1*1.1+2*2); //from figure 2.14
D2=sqrt(1.9*1.9+2*2);  //from figure 2.14
Mpt=0.921*log10(D2/D1);  //mutual inductance
Vt=abs(%i*2*%pi*50*Mpt*10^(-3)*100);//when 100A is flowing in the power lines

printf(”\n\nMutual inductance between power line and telephone line = %f mH/km\n\n”,Mpt);
printf(”\n\nVoltage induced in the telephone circuit = %.3f V/km\n\n”,Vt);
```

**Scilab code Exa 2.7** Bundled Conductor Three Phase Line

```
//Chapter 2
//Example 2.7
```

```
//page 69
//To find inductive reactance of for the three phase bundled conductors
clear;clc;
r=0.01725; //radius of each conductor
//from the figure we can declare the distances
d=7;
s=0.4;
//Mutual GMD between bundles of phases a and b
Dab=(d*(d+s)*(d-s)*d)^(1/4);
//Mutual GMD between bundles of phases b and c
Dbc=Dab ; //by symmetry
//Mutual GMD between bundles of phases c and a
Dca=(2*d*(2*d+s)*(2*d-s)*2*d)^(1/4);
//Equivalent GMD is calculated as
Deq=(Dab*Dbc*Dca)^(1/3);
//self GMD is given by
Ds=(0.7788*1.725*10^(-2)*0.4*0.7788*1.725*10^(-2)*0.4)^(1/4);
//Inductive reactance per phase is given by
Xl=2*%pi*50*10^(-3)*0.461*log10(Deq/Ds); //10^(-3) because per km is asked
printf("\n\nMutual GMD between bundles of phases a and b = %0.3fm\n\n",Dab);
printf("Mutual GMD between bundles of phases b and c = %0.3fm\n\n",Dbc);
printf("Mutual GMD between bundles of phases c and a = %0.3fm\n\n",Dca);
printf("Equivalent GMD = %0.3fm\n\n",Deq);
printf("Self GMD of the bundles = %0.3fm\n\n",Ds);
printf("Inductive reactance per phase = %0.3f ohms/km\n\n",Xl);

//now let us compute reactance when center to centerr distances are used
Deq1=(d*d*2*d)^(1/3);
Xl1=2*%pi*50*0.461*10^(-3)*log10(Deq1/Ds);
printf("\n When radius of conductors are neglected
and only distance between conductors are used, we get below results:\n\n");
printf("Equivalent mean distance is = %f\n\n",Deq1);
printf("Inductive reactance per phase = %0.3f ohms/km\n\n",Xl1);

//when bundle of conductors are replaced by an equivalent single conductor
cond_dia=sqrt(2)*1.725*10^(-3); //conductor diameter for same cross-sectional area
Xl2=2*%pi*50*0.461*10^(-3)*log10(Deq1/cond_dia);
printf("\nWhen bundle of conductors are replaced by an equivalent single conductor:\n\n");
printf("Inductive reactance per phase = %0.3f ohms/km\n\n",Xl2) ;
percentage_increase=((Xl2-Xl1)/Xl1)*100;
printf("This is %0.2f higher than corresponding value for a bundled conductor line.",percentage_increase);
```

# Chapter 3

# Capacitance of Transmission Lines

**Scilab code Exa 3.1** Capacitance of a single phase line

```
//Chapter 3
//Example 3.1
//page 87
//To calculate the capacitance to neutral of a single phase line
clear;clc;
r=0.328; //radius of the conductors
D=300; //distance between the conductors
h=750; //height of the conductors

//calculating capacitance neglecting the presence of ground
//using Eq (3.6)
Cn=(0.0242/(log10(D/r)));
printf("\nCapacitance to neutral /km of the given single phase line neglecting presence of the earth (using Eq 3.6) is = %0.5f uF/km\n\n",Cn);

//using Eq (3.7)
```

```
Cn=(0.0242)/log10((D/(2*r))+((D^2)/(4*r^2)-1)^0.5);
printf("Capacitance to neutral /km of the given single phase line neglecting presence of the earth (using Eq 3.7) is = %0.5f uF/km\n\n",Cn);

//Consudering the effect of earth and neglecting the non uniformity of the charge
Cn=(0.0242)/log10(D/(r*(1+((D^2)/(4*h^2)))^0.5));
printf("Capacitance to neutral /km of the given single phase line considering the presence of the earth and neglecting non uniformity of charge distribution (using Eq 3.26b) is = %0.5f uF/km\n\n",Cn);
```

**Scilab code Exa 3.2** Charging current of a threephase line

```
//Chapter 3
//Example 3.2
//page 88
//To claculate the capacitance to neutral and charging current of a three phase transmission line
clear;clc;
d=350;    //distance between adjacent lines
r=1.05/2;     //radius of the conductor
v=110e3;    //line voltage;
f=50;

Deq=(d*d*2*d)^(1/3);    //GMD or equivalent

Cn=(0.0242/log10(Deq/r));

Xn=1/(2*%pi*f*Cn*10^(-6));    // Cn is in uF hence we
```

```
    add 10^6 while printing

Ic=(v/sqrt(3))/Xn;

printf("\nCapacitance to neutral is = %f uF/km\n\n",
    Cn);
printf("Capacitive rectance of the line is = %f ohm/
    km to neutral\n\n",Xn);
printf("Charging Current = %0.2f A/km\n\n",Ic);
```

**Scilab code Exa 3.3** Double circuit three phase transmission line

```
//Chapter 3
//Example 3.3
//page 88
//To claculate the capacitance to neutral and
    charging current of a double circuit three phase
    transmission line
clear;clc;

//After deriving the equation for Cn from the
    textbook and starting calculation from Eq 3.36
    onwards

r=0.865*10^(-2); frequency=50; v=110e3;
h=6; d=8; j=8; //Referring to fig given in the
    textbook

i=((j/2)^2+((d-h)/2)^2)^(1/2);
f=(j^2+h^2)^(1/2);
g=(7^2+4^2)^(1/2);


```

```
Cn=4*%pi*8.85*10^(-12)/(log((((i^2)*(g^2)*j*h)/((r^3)*(f^2*d)))^(1/3)));

Cn=Cn*1000 ; //Cn is in per m.to convert it to per km,we multiply by 1000
WCn=2*%pi*frequency*Cn;

Icp=(v/sqrt(3))*WCn;

Icc=Icp/2;

printf("\nTotal capacitance to neutral for two conductors in parallel = %0.6f uF/km \n\n",Cn*10^(6));
printf("Charging current/phase = %0.3f A/km \n\n",Icp);
printf("Charging current/conductor = %0.4f A/km \n\n",Icc);
```

# Chapter 4

# Representation of Power System Components

**Scilab code Exa 4.1** Per Unit Reactance Diagram

```
//Chapter 4
//Example 4.1
//page 103
// to draw the per unit reactance diagram
clear;clc;
mvab=30; kvb=33; //MVA base and KVA base are
    selected

gen1_mva=30; gen1_kv=10.5; gen1_x=1.6;  //Generator
   No.1 details
gen2_mva=15; gen2_kv=6.6; gen2_x=1.2;  //Generator
   No.2 details
gen3_mva=25; gen3_kv=6.6; gen3_x=0.56;  //Generator
   No.3 details

t1_mva=15; t1_hv=33; t1_lv=11; t1_x=15.2; //
    Transformer T1 details
t2_mva=15; t2_hv=33; t2_lv=6.2; t2_x=16; //
    Transformer T1 details
```

```

tl_x=20.5; //Transmission line recatance

//Loads are neglected as said in the problem

tl_pu=(tl_x*mvab)/kvb^2;
t1_pu=(t1_x*mvab)/kvb^2;
t2_pu=(t2_x*mvab)/kvb^2;
gen1_kv_base=t1_lv;
gen1_pu=(gen1_x*mvab)/gen1_kv_base^2;
gen2_kv_base=t2_lv;
gen2_pu=(gen2_x*mvab)/gen2_kv_base^2;
gen3_pu=(gen3_x*mvab)/gen2_kv_base^2;

//diplaying the results on console

printf('Per unit impedance of the components of the given power system are as follows :\n\n');

printf('Transmission line: %0.3f \n\n',tl_pu);

printf('Transformer T1: %0.3f \n\n',t1_pu);

printf('Transformer T2: %0.3f \n\n',t2_pu);

printf('Generator 1: %0.3f \n\n',gen1_pu);

printf('Generator 2: %0.3f \n\n',gen2_pu);

printf('Generator 3: %0.3f \n\n',gen3_pu);
```

**Scilab code Exa 4.2** Per Unit Calculation

```
//Chapter 4
//Example 4.2
//page 104
// To draw the per unit reactance diagram when pu values are specified based on euipment rating
clear;clc;
mvab=30; kvb=11; //MVA base and KVA base are selected in the circuit of generator 1

gen1_mva=30; gen1_kv=10.5; gen1_x=0.435;  //Generator No.1 details
gen2_mva=15; gen2_kv=6.6; gen2_x=0.413;  //Generator No.2 details
gen3_mva=25; gen3_kv=6.6; gen3_x=0.3214;  //Generator No.3 details

t1_mva=15; t1_hv=33; t1_lv=11; t1_x=0.209; //Transformer T1 details
t2_mva=15; t2_hv=33; t2_lv=6.2; t2_x=0.220; //Transformer T1 details

tl_x=20.5; //Transmission line recatance

//Loads are neglected as said in the problem

tl_pu=(tl_x*mvab)/t1_hv^2;
t1_pu=t1_x*(mvab/t1_mva);
t2_pu=t2_x*(mvab/t2_mva);
gen1_pu=gen1_x*(mvab/gen1_mva)*(gen1_kv/kvb)^2;
gen2_kv_base=t2_lv;
gen2_pu=gen2_x*(mvab/gen2_mva)*(gen2_kv/gen2_kv_base)^2;
gen3_kv_base=t2_lv;
gen3_pu=gen3_x*(mvab/gen3_mva)*(gen3_kv/gen3_kv_base)^2;

//diplaying the results on console

```

```
printf('Per unit impedance of the components of the given power system are as follows :\n\n');

printf('Transmission line: %0.3f \n\n',tl_pu);

printf('Transformer T1: %0.3f \n\n',t1_pu);

printf('Transformer T2: %0.3f \n\n',t2_pu);

printf('Generator 1: %0.3f \n\n',gen1_pu);

printf('Generator 2: %0.3f \n\n',gen2_pu);

printf('Generator 3: %0.3f \n\n',gen3_pu);
```

**Scilab code Exa 4.3** Excitation EMF and Reactive Power Calculation

```
//Taking Base value MVA and KVA
clear;clc;
mvab=645; //Base MVA in 3-phase
kvb=24; //Base KV, line-to-line

vl=24/kvb; //Load voltage
xs=1.2;
xs=(xs*mvab)/kvb^2; // xs converted to its pu

//since the generator is operating at full load & 0.9pf
pf_angle=acos(0.9);
Ia=1*(cos(pf_angle)-%i*sin(pf_angle)); //load current
//to find excitation emf
ef=vl+%i*xs*Ia;
```

```
delta=atand(imag(ef)/real(ef));//positive for leading
ef=abs(ef)*kvb; //pu to actual unit conversion
  if(delta>0) then lead_lag='leading';
   else lead_lag='lagging';
  end
printf('Excitation emf= %0.2f kV at an angle %0.3f (%s) \n\n',ef,delta,lead_lag);
//to find reactive power drawn by load
Q=vl*abs(imag(Ia));
Q=Q*mvab; //pu to actual unit conversion
printf('Reactive power drawn by laod= %d MVAR',Q);
```

**Scilab code Exa 4.4** Power Factor And Load Angle Calculation

```
//Taking Base value MVA and KVA
clear;clc;
global mvab
mvab=645; //Base MVA in 3-phase
kvb=24; //Base KV, line-to-line
vt=24/kvb; //Terminal voltage
xs=1.2;
xs=(xs*mvab)/kvb^2; // xs converted to its pu

//since the generator is operating at full load & 0.9pf
pf_angle=acos(0.9);
Ia=1*(cos(pf_angle)-%i*sin(pf_angle)); //load current
//to find excitation emf
ef=vt+%i*xs*Ia;
ef=abs(ef);
P=1*0.9; //at Full load
```

```

/////// writing an inline function /////////////////
function [pf,lead_lag,Q]=excitation_change(P,ef,vt,xs)
sin_delta=(P*xs)/(ef*vt);
delta=asind(sin_delta);
ef0=ef*(cosd(delta)+(%i*sind(delta)));
Ia=(ef0-vt)/(%i*xs);
Ia_mag=abs(Ia);Ia_ang=atand(imag(Ia)/real(Ia)); // Magnitude and angle of Ia
pf=cosd(abs(Ia_ang));
if(Ia_ang>0) then lead_lag='leading';
  elseif (Ia_ang==0) then lead_lag='unity pf'
    else lead_lag='lagging';
  end
Q=vt*Ia_mag*sind(abs(Ia_ang));
Q=abs(Q)*mvab;
endfunction
//////////////////////////////////////////////////////////


// First Case when Ef is increased by 20% at same real load now
 ef1=ef*1.2;
[pf1,lead_lag1,Q1]=excitation_change(P,ef1,vt,xs);
disp("Case (i): When Ef is increased by 20% ");
printf('\n\tPower factor pf= %0.2f %s \n',pf1,lead_lag1);
printf('\tReactive power drawn by the load = %0.1f MVAR \n',Q1);

//Second Case when Ef is decreased by 20% at same real load now
 ef2=ef*0.8;
[pf2,lead_lag2,Q2]=excitation_change(P,ef2,vt,xs);
disp("Case (ii): When Ef is decreased by 20% ");
```

```
printf('\n\tPower factor pf= %0.2f %s \n',pf2,lead_lag2);
printf('\tReactive power drawn by the load = %0.1f MVAR \n',Q2);

disp('The answers given here are exact values. Textbook answers has an approximation of upto 2 decimal places on Xs,Ia,pf. ');
```

# Chapter 5

# Characteristics and Performance of Power Transmission Lines

**Scilab code Exa 5.1** SendingEnd voltage and voltage regulation

```
//Chapter 5
//Example 5.1
//page 132
//To find sending-end voltage and voltage regulation
clc;clear;

load1=5000; //kW
pf=0.707;
Vr=10000; //receiving end voltage
R=0.0195*20;
X=2*%pi*50*0.63*10^-3*20;

//to find sending end voltage and voltage regulation
I=load1*1000/(Vr*pf);
Vs=Vr+I*(R*pf+X*sin(acos(pf)));
voltage_regulation=(Vs-Vr)*100/Vr;
printf('\n\nReceiving current =I=%d A\n',I);
printf('Sending end voltage =Vs=%d V\n',Vs);
printf('Voltage Regulation=%0.2f %%',voltage_regulation);

//to find the value of the capacitor to be connected in parallel to the load
voltage_regulation_desi=voltage_regulation/2;
Vs=(voltage_regulation_desi/100)*Vr+Vr;
//by solving the equations (i) and (ii)
pf=0.911;
Ir=549;
Ic=(Ir*(pf-%i*sin(acos(pf))))-(707*(0.707-%i*0.707));
Xc=(Vr/imag(Ic));
c=(2*%pi*50*Xc)^-1;
printf('\n\nCapacitance to be connected across the load so as to reduce voltage regulation by half of the above voltage regulation is given by :\n C = %d uF\n',c*10^6);

//to find efficiency in both the cases
//case(i)
losses=I*I*R*10^-3;
n=(load1/(load1+losses))*100;
printf('\n Efficiency in : \nCase(i) \t n=%0.1f%%',n);
//caase(ii)
losses=Ir*Ir*R*10^-3;
n=(load1/(load1+losses))*100;
printf('\nCase(ii) \t n=%0.1f%%',n);
```

**Scilab code Exa 5.2** Voltage at the power station end

```
//Chapter 5
//Example 5.2
//page 134
//To find voltage at the bus at the power station end
clc;clear;

base_MVA=5;
base_kV=33;
pf=0.85;
cable_impedance=(8+%i*2.5);
cable_impedance=cable_impedance*base_MVA/(base_kV^2);

transf_imp_star=(0.06+%i*0.36)/3; //equivalent star impedance of winding of the transformer
Zt=(transf_imp_star*5/(6.6^2))+((0.5+%i*3.75)*5/(33^2));
total=cable_impedance+2*Zt;

load_MVA=1;
load_voltage=6/6.6;
load_current=1/load_voltage;

Vs=load_voltage+load_current*(real(total)*pf+imag(total)*sin(acos(pf)));
Vs=Vs*6.6;
printf('\n\nCable impedance= (%0.3f+j%0.4f) pu\n',real(cable_impedance),imag(cable_impedance));
printf('\nEquivalent star impedance of 6.6kV winding of the transformer =(%0.2f+j%0.2f) pu\n',real(transf_imp_star),imag(transf_imp_star));
printf('\nPer unit transformer impedance,Zt=(%0.4f+j%0.3f) pu\n',real(Zt),imag(Zt));
printf('\nTotal series impedance=(%0.3f+j%0.3f) pu\n',real(total),imag(total));
printf('\nSending end Voltage =|Vs|=%0.2fkV (line-to-line)',Vs);
```

**Scilab code Exa 5.3** Problem with mixed end condition

```
//Chapter 5
//Example 5.3
//page 135
//problem with mixed end condition
clc;clear;
Vr=3000; //receiving end voltage
pfs=0.8; //sending end power factor
Ps=2000*10^3; //sending end active power
z=0.4+%i*0.4; //series impedance
Ss=Ps/pfs; //sending end VA
Qs=Ss*sqrt(1-pfs^2); //sending end reacive power

//by substituting all the values to the equation (
    iii)
deff('[y]=fx(I)',"y=(Vr^2)*(I^2)+2*Vr*(I^2)*(real(z)
    *((Ps-real(z)*(I^2))/Vr)+imag(z)*((Qs-imag(z)*(I
    ^2))/Vr))+(abs(z))^2*(I^4)-(Ss^2)");
I=fsolve(100,fx);

pfR=(Ps-real(z)*(I^2))/(Vr*I); //Cos(phi_r)
Pr=Vr*I*pfR;
Vs=(Ps/(I*pfs));

printf('\nLoad Current |I|= %0.2f A',I);
printf('\nLoad Pr=%d W',Pr);
printf('\nReceiving end power factor=%0.2f',pfR);
printf('\nSupply Voltage=%0.2fV',Vs);
```

**Scilab code Exa 5.4** Medium Transmission line system

```
//Chapter 5
//Example 5.4
//page 138
//to find sending end voltage and voltage regulation of a medium transmission line system
clear;clc;
D=300;
r=0.8;
L=0.461*log10(D/(0.7788*r));
C=0.0242/(log10(D/r));
R=0.11*250;
X=2*%pi*50*L*0.001*250;
Z=R+%i*X;
Y=%i*2*%pi*50*C*0.000001*250;
Ir=((25*1000)/(132*sqrt(3)))*(cosd(-36.9)+%i*sind(-36.9));
Vr=(132/sqrt(3));
A=(1+(Y*Z/2));
Vs=A*Vr+Z*Ir*10^(-3);
printf('\n\nVs(per phase)=(%0.2f+%0.2f)kV',real(Vs),imag(Vs));
Vs=abs(Vs)*sqrt(3);
printf('\n\n|Vs|(line)=%d kV',Vs);
Vr0=Vs/abs(A);
printf('\n\n|Vr0|(line no load)=%0.1fkV',Vr0);
Vol_regu=(Vr0-132)/132;
printf('\n\nVoltage Regulation=%0.1f%%\n\n',Vol_regu*100);
```

**Scilab code Exa 5.5** Maximum permissible length and and Frequency

```
//Chapter 5
//Example 5.5
//page 147
//to find maximum permissible length and and frequency
clc;clear;
R=0.125*400;
X=0.4*400;
Y=2.8*(10^-6)*400*%i;
Z=R+X*%i;

//(i) At no-load
A=1+(Y*Z/2);
C=Y*(1+Y*Z/6);
VR_line=220000/abs(A);
Is=abs(C)*VR_line/sqrt(3);
printf('\n\n |VR| line = %d kV',VR_line/1000);
printf('\n |Is| = %d A',Is);

//(ii) to find maximum permissible length
//By solving the equations shown in the book,we get
l=sqrt((1-0.936)/(0.56*10^(-6)));
printf('\n\n Maximum permissible length of the line = %d km',l);

//(iii) to find maximum permissible frequency for the case(i)
//By solving the equations shown in the book,we get
f=sqrt(((1-0.88)*50*50)/(0.5*1.12*10^-3*160));
printf('\n\n Maximum permissible frequency = %0.1f
   Hz\n\n',f);
```

**Scilab code Exa 5.6** Incident and Reflected voltages

```
//Chapter 5
//Example 5.6
//page 149
//to find incident and reflected voltages
clear;clc;

R=0.125;
X=0.4;
y=%i*2.8*10^(-6);
z=R+%i*X;

r=sqrt(y*z); //propogation constant
a=real(r); //attenuation constant
b=imag(r); //phase constant

//(a) At the receiving-end;
Vr=220000;
Inci_vol=Vr/(sqrt(3)*2);
Refl_vol=Vr/(sqrt(3)*2);
printf('\n\nIncident Vvoltage=%0.2f kV',Inci_vol
   /1000);
printf('\nReflected Vvoltage=%0.2f kV',Refl_vol
   /1000);

//(b) At 200km from the receiving-end
x=200;
Inci_vol=Inci_vol*exp(a*x)*exp(%i*b*x);
Refl_vol=Refl_vol*exp(-a*x)*exp(-%i*b*x);
printf('\n\nIncident voltage=%0.2f @ %0.1f deg kV',
abs(Inci_vol)/1000,atand(imag(Inci_vol)/real(Inci_vol)));
printf('\nReflected voltage=%0.2f @ %0.1f deg kV',abs(Refl_vol)/1000,atand(imag(Refl_vol)/real(Refl_vol)));

//(c) Resultant voltage at 200km from the receiving-end
res=Inci_vol+Refl_vol;
printf('\n\nResultant line-to-line voltage at 200km =%0.2f kV',abs(res)*sqrt(3)/1000);
```

**Scilab code Exa 5.7** Tabulate characteristics using different methods

```
//Chapter 5
//Example 5.7
//page 138
//to tabulate characteristics of a system using different methods
clear;clc;

Z=40+125*%i;
Y=%i*10^(-3);
Ir=((50*10^6)/(220000*0.8*sqrt(3)))*(cosd(-36.9)+%i*sind(-36.9));
Vr=220000/sqrt(3);

//(a) Short line approximation
Vs=Vr+Ir*Z;
Vs_line1=Vs*sqrt(3);
Is1=Ir;
pfs1=cos(atan(imag(Vs)/real(Vs))+acos(0.8));
Ps1=sqrt(3)*abs(Vs_line1)*abs(Is1)*pfs1;
```

```

//(b) Nominal pi method
A=1+Y*Z/2;
D=A;
B=Z;
C=Y*(1+Y*Z/4);
Vs=A*Vr+B*Ir;
Is2=C*Vr+D*Ir;
Vs_line2=sqrt(3)*Vs;
pfs2=cos(atan(imag(Is2)/real(Is2))-atan(imag(Vs)/real(Vs)));
Ps2=sqrt(3)*abs(Vs_line2)*abs(Is2)*pfs2;

//(c) Exact transmission line equations
rl=sqrt(Z*Y); //propogation constant
Zc=sqrt(Z/Y); //characteristic impedance
A=cosh(rl);
B=Zc*sinh(rl);
C=sinh(rl)/Zc;
D=cosh(rl);
Vs=A*Vr+B*Ir;
Is3=C*Vr+D*Ir;
Vs_line3=sqrt(3)*Vs;
pfs3=cos(atan(imag(Is3)/real(Is3))-atan(imag(Vs)/real(Vs)));
Ps3=sqrt(3)*abs(Vs_line3)*abs(Is3)*pfs3;

//(d) Approximation
A=(1+Y*Z/2);
B=Z*(1+Y*Z/6);
C=Y*(1+Y*Z/6);
D=A;
Vs=A*Vr+B*Ir;
Is4=C*Vr+D*Ir;
Vs_line4=sqrt(3)*Vs;
pfs4=cos(atan(imag(Is4)/real(Is4))-atan(imag(Vs)/real(Vs)));
Ps4=sqrt(3)*abs(Vs_line4)*abs(Is4)*pfs4;
```

```

//converting all the values to their standard form
   before writing it to table

//voltage to kV
Vs_line1=abs(Vs_line1)/1000;
Vs_line2=abs(Vs_line2)/1000;
Vs_line3=abs(Vs_line3)/1000;
Vs_line4=abs(Vs_line4)/1000;

//Current to kA
Is1=Is1/1000;
Is2=Is2/1000;
Is3=Is3/1000;
Is4=Is4/1000;

//power to MW5
Ps1=Ps1/1000000;
Ps2=Ps2/1000000;
Ps3=Ps3/1000000;
Ps4=Ps4/1000000;

//preparinf table
printf("\n\
   n------------------------------------------------------------------------
   ");
printf('\n          \t\tShort line \t\t        Nominal
   Pi \t\t Exact      \t\t Approximation');
printf("\
   n------------------------------------------------------------------------
   ");
printf('\n|Vs|line\t\t%0.2fkV          \t\t %0.2fkV\t\
   t %0.2fkV \t\t %0.2fkV',Vs_line1,Vs_line2,
   Vs_line3,Vs_line4);
printf('\nIs       \t\t%0.3f@%0.1fdeg kA \t\t%0.2f@%0
   .1fdeg kA\t\t%0.4f@%0.1fdeg kA\t%0.2f@%0.1fdeg kA
   ',abs(Is1),tand(imag(Is1)/real(Is1)),abs(Is2),
   tand(imag(Is2)/real(Is2)),abs(Is3),tand(imag(Is3)
/real(Is3)),abs(Is4),tand(imag(Is4)/real(Is4)));
printf('\npfs      \t\t%0.3f lagging        \t\t%0.3f leading \t\t%0.3f leading \t\t%0.3f leading',pfs1,pfs2,pfs3,pfs4);
printf('\nPs       \t\t%0.2f MW            \t\t%0.2f MW \t\t%0.2f MW \t\t%0.2f MW',Ps1,Ps2,Ps3,Ps4);
printf("\n---------------------------------------------------------------------\n\n\n");
```

**Scilab code Exa 5.8** Torque angle and Station powerfactor

```
//Chapter 5
//Example 5.8
//page 162
//to estimate the torque angle and station powerfactor
clear;clc;
Sd1=15+%i*5;
Sd2=25+%i*15;
//case(a) cable impedance=j0.05pu
r=0;
x=%i*0.05;
PG1=20;
PG2=20;
Ps=5;Pr=5;
V1=1;
V2=1;
d1=asind(Ps*abs(x)/(V1*V2)); //delta1
V1=V1*(cosd(d1)+%i*sind(d1));
Qs=((abs(V1)^2)/abs(x))-((abs(V1)*abs(V2))*cosd(d1)/(abs(x)));
Qr=(((abs(V1)*abs(V2))*cosd(d1)/(abs(x)))-(abs(V1)
   ^2)/abs(x));
Ql=Qs-Qr;
Ss=Ps+%i*Qs;
Sr=Pr+%i*Qr;
Sg1=Sd1+Ss;
Sg2=Sd2-Sr;
pf1=cos(atan(imag(Sg1)/real(Sg1)));
pf2=cos(atan(imag(Sg2)/real(Sg2)));
printf('\n\nCase(a)\nTotal load on station1=%d+j%0.3
   f pu',real(Sg1),imag(Sg1));
printf('\nPower factor of station1=%0.3f pu lagging'
   ,pf1);
printf('\n\Total load on station2=%d+j%0.3f pu',real
   (Sg2),imag(Sg2));
printf('\nPower factor of station2=%0.3f pu lagging'
   ,pf2);
//case(b) cable impedance=0.005+j0.05;
r=0.005;
PG1=20;
V1=1;V2=1;
Ps=5;
//from the eq(i) in the textbook,we can calculate d1
z=r+x;
theta=atand(imag(z)/real(z));
z=abs(z);
d1=acosd(z*(V1^2*cosd(theta)/z-Ps)/(V1*V2))-theta;
Qs=(V1^2*sind(theta)/z)-(V1*V2*sind(theta+d1)/z);
Qg1=5+Qs;
Pr=(V1*V2*cosd(theta-d1)/z)-(V1^2*cosd(theta)/z);
Pg2=25-Pr;
Qr=(V1*V2*sind(theta-d1)/z)-(V1^2*sind(theta)/z);
Qg2=15-Qr;
Ss=Ps+%i*Qs;
Sr=Pr+%i*Qr;
Sg1=Sd1+Ss;
Sg2=Sd2-Sr;
pf1=cos(atan(imag(Sg1)/real(Sg1)));
pf2=cos(atan(imag(Sg2)/real(Sg2)));
```

```
printf('\n\nCase(b)\nTotal load on station1=%d+j%0.3f pu',real(Sg1),imag(Sg1));
printf('\nPower factor of station1=%0.3f pu lagging',pf1);
printf('\n\Total load on station2=%d+j%0.3f pu',real(Sg2),imag(Sg2));
printf('\nPower factor of station2=%0.3f pu lagging\n\n',pf2);
```

**Scilab code Exa 5.9** Power Voltage and Compensating equipment rating

```
//Chapter 5
//Example 5.9
//page 165
//to determine power,voltage,compensating equipment rating
clear;clc;
A=0.85;
B=200;

//case(a)
Vs=275000;
Vr=275000;
a=5;b=75; //alpha and beta
Qr=0;
//from equation 5.62
d=b-asind((B/(Vs*Vr))*(Qr+(A*Vr^2*sind(b-a)/B))); //delta
Pr=(Vs*Vr*cosd(b-d)/B)-(A*Vr^2*cosd(b-a)/B);
printf('\n\ncase(a)\nPower at unity powerfactor that can be received =%0.1f MW',Pr/10^6);

//case(b)
```

```
Pr=150*10^6;
d=b-acosd((B/(Vs*Vr))*(Pr+(A*Vr^2*cosd(b-a)/B))); //delta
Qr=(Vs*Vr*sind(b-d)/B)-(A*Vr^2*sind(b-a)/B);
Qc=-Qr;
printf('\n\ncase(b)\nRating of the compensating equipment = %0.2f MVAR',Qc/10^6);
printf('\ni.e the compensating equipment must feed positive VARs into the line');


//case(c)
Pr=150*10^6;
Vs=275000;
//by solving the two conditions given as (i) and (ii), we get
Vr=244.9*10^3;
printf('\n\ncase(c)\nReceiving end voltage = %0.1f kV',Vr/1000);
```

**Scilab code Exa 5.10** MVA rating of the shunt reactor

```
//Chapter 5
//Example 5.10
//page 170
//To determine the MVA rating of the shunt reactor
clear;clc;
v=275;
l=400;
R=0.035*l;
X=2*%pi*50*1.1*l*10^-3;
Z=R+%i*X;
Y=2*%pi*50*0.012*10^-6*l*%i;
```

```
A=1+(Y*Z/2);
B=Z;
Vs=275;
Vr=275;
r=(Vs*Vr)/abs(B);
Ce=abs(A/B)*Vr^2;
printf('Radius of the receiving-end circle=%0.1f MVA\n\n',r);
printf('Location of the center of receiving-end circle= %0.1f MVA\n\n',Ce);
printf('From the graph, 55 MVA shunt reactor is required\n\n');
theta=180+82.5;
x=-75:0.01:450;
a=Ce*cosd(theta); //to draw the circle
b=Ce*sind(theta);
y=sqrt(r^2-(x-a)^2)+b;
x1=a:0.001:0;
y1=tand(theta)*x1;
plot(x,y,x1,y1);
title('Circle diagram for example 5.10');
xlabel('MW');
ylabel('MVAR');
plot(a,b,'markersize',150);
xgrid(2)
set(gca(),"grid",[0,0])
get("current_axes");
xstring (-75,25,'55 MVAR');
xstring(-75,-25,'83.5 deg');
xstring(-20,-300,'487.6 MVA');
xstring(300,-100,'544.3 MVA');
```

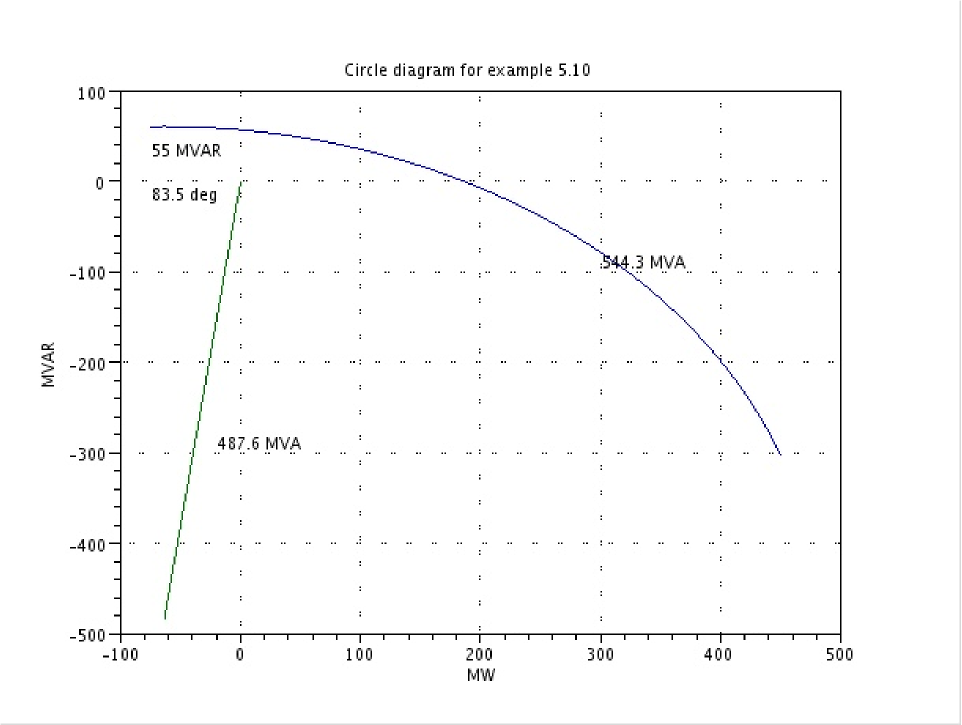


Figure 5.1: MVA rating of the shunt reactor

**Scilab code Exa 5.11** SendingEnd voltage and maximum power delivered

```
//Chapter 5
//Example 5.11
//page 172
//To determine sending-end voltage.maximum power delivered
clear;clc;

A=0.93*(cosd(1.5)+%i*sind(1.5));
B=115*(cosd(77)+%i*sind(77));
Vr=275;
Ce=abs(A/B)*Vr^2;
printf('Centre of the receiving end circle is = %0.1f MVA\n\n',Ce);
CrP=850;Vs=CrP*abs(B)/Vr;
printf('(a) From the diagram,\n\tCrP=%d \n \tSending end voltage|Vs|= %0.1f kV\n\n',CrP,Vs);
Vs=295; //given
r=(Vs*Vr)/abs(B);
Pr_m=556; //from the diagram
printf('(b) Radius of the circle diagram = %0.1f MVA\n\t PR_max=%d MW\n\n',r,Pr_m);
Ps=295; //from the diagram;
printf('(c) Additional MVA to be drawn from the line is = P''S=%d MVAR\n\n',Ps);
```

# Chapter 6

# Load Flow Studies

**Scilab code Exa 6.1** Ybus using singular transformation

```
//Chapter 6
//Example 6.1
//page 195
//To Ybus using singular transformation

clear;clc;
printf('Let us solve this problem by giving values
   given in the table 6.1 instead of keeping it in
   variables');

y10=1;y20=1;y30=1;y40=1;
y34=2-%i*6;y23=0.666-%i*2;
y12=2-%i*6;y24=1-%i*3;
y13=1-%i*3;

Y=[y10 0 0 0 0 0 0 0 0;
   0 y20 0 0 0 0 0 0 0;
   0 0 y30 0 0 0 0 0 0;
   0 0 0 y40 0 0 0 0 0;
   0 0 0 0 y34 0 0 0 0;
   0 0 0 0 0 y23 0 0 0;
   0 0 0 0 0 0 y12 0 0;
   0 0 0 0 0 0 0 y24 0;
   0 0 0 0 0 0 0 0 y13];
A=[1 0 0 0;
   0 1 0 0;
   0 0 1 0;
   0 0 0 1;
   0 0 1 -1;
   0 -1 1 0;
   1 -1 0 0;
   0 -1 0 1;
   -1 0 1 0];
printf('\n\n Ybus matrix using singular transformation for the system of fig.6.2 is \n Ybus= ');
Y=A'*Y*A;
disp(Y);
// for verification let us calculate as given in the text book
printf('\n\n For verification,calculating Ybus substituting as given in the text book\n Ybus(verifiaction)=');
Yveri=[(y10+y12+y13) -y12 -y13 0;-y12 (y20+y12+y23+y24) -y23 -y24;-y13 -y23 (y30+y13+y23+y34) -y34;0 -y24 -y34 (y40+y24+y34)];
disp(Yveri);
```

**Scilab code Exa 6.2** Ybus of a sample system

```
//Chapter 6
//Example 6.2
//page 195
//To Ybus of sample system
```

```
clear;clc;

y10=1;y20=1;y30=1;y40=1;
y34=2-%i*6;y23=0.666-%i*2;
y12=2-%i*6;y24=1-%i*3;
y13=1-%i*3;

//to form Ybus matrix
Y11=y13;Y12=0;Y13=-y13;Y14=0;
Y21=0;Y22=y23+y24;Y23=-y23;Y24=-y24;
Y31=-y13;Y32=-y23;Y33=y13+y23+y34;Y34=-y34;
Y41=0;Y42=-y24;Y43=-y34;Y44=y34+y24;

//case(i) line shown dotted is not connected
Ybus=[Y11 Y12 Y13 Y14;
      Y21 Y22 Y23 Y24;
      Y31 Y32 Y33 Y34;
      Y41 Y42 Y43 Y44];
printf('(i)Assuming that the line shown is not connected \n Ybus= ');disp(Ybus);
//case(ii) line shown dotted is connected
Y12=Y12-y12;Y21=Y12;
Y11=Y11+y12;
Y22=Y22+y12;

Ybus=[Y11 Y12 Y13 Y14;
      Y21 Y22 Y23 Y24;
      Y31 Y32 Y33 Y34;
      Y41 Y42 Y43 Y44];
printf('\n\n(ii)Assuming that the line shown is connected \n Ybus= ');disp(Ybus);
```

**Scilab code Exa 6.3** Approximate load flow solution

```
//Chapter 6
//Example 6.3
//page 201
//To find an approximate load flow solution
clear;clc;

//////////////////////////////////////////////////////////////////////
//Realdemand    Reactive demand    Real generation    Reactive generation    Bus
//////////////////////////////////////////////////////////////////////
 Pd1=1;          Qd1=0.5;          Pg1=0;          Qg1=0;//initialization  1
 Pd2=1;          Qd2=0.4;          Pg2=4;          Qg2=0;//initialization  2
 Pd3=2;          Qd3=1;            Pg3=0;          Qg3=0;//initialization  3
 Pd4=2;          Qd4=1;            Pg4=0;          Qg4=0;//initialization  4

Pg1=Pd1+Pd2+Pd3+Pd4-Pg2;

//Ybus matrix from the network
Ybus=[-21.667*%i 5*%i 6.667*%i 10*%i;
      5*%i -21.667*%i 10*%i 6.667*%i;
      6.667*%i 10*%i -16.667*%i 0;
      10*%i 6.667*%i 0 -16.667*%i];
printf('Ybus matrix of the system is given by \nYbus =');disp(Ybus);
//as given in the text book using approximate load flow equations and simplifying (ii),(iii),(iv)
//delta matrix(x) is of the from A*x=B
A=[-5 21.667 -10 -6.667;
   -6.667 -10 16.667 0;
   -10 -6.667 0 16.667
```

```
    1  0  0  0];

B=[3;  -2;  -2;0];

delta=inv(A)*B; //solving for delta
printf('\nDelta of the system is given by \ndelta(
    rad)=');disp(delta);

Q1=-5*cos(delta(2,1))-6.667*cos(delta(3,1))-10*cos(
    delta(4,1))+21.667;
Q2=-5*cos(delta(2,1))-10*cos(delta(3,1)-delta(2,1))
    -6.667*cos(delta(4,1)-delta(2,1))+21.667;
Q3=-6.667*cos(delta(3,1))-10*cos(delta(3,1)-delta
    (2,1))+16.667;
Q4=-10*cos(delta(4,1))-6.667*cos(delta(4,1)-delta
    (2,1))+16.667;

Q=[Q1;Q2;Q3;Q4];
printf('\nInjected reactive power at the buses is
    given by \nQi(in pu)=');disp(Q);

Qg1=Q1+Qd1;
Qg2=Q2+Qd2;
Qg3=Q3+Qd3;
Qg4=Q4+Qd4;

Qg=[Qg1;Qg2;Qg3;Qg4];
printf('\n Reactive power generation at the four
    buses are \nQgi(in pu)=');disp(Qg);
Qd=[Qd1;Qd2;Qd3;Qd4];
Ql=sum(Qg)-sum(Qd);
printf('\nReactive power losses are QL=%0.5f pu',Ql)
    ;

printf('\n\nLine Flows are given as:\n');
P13=(abs(Ybus(1,3)))*sin(delta(1,1)-delta(3,1));P31
    =-P13;printf('\nP13=-P31=%0.3f pu',P13);
P12=(abs(Ybus(1,2)))*sin(delta(1,1)-delta(2,1));P21
```

```
=-P12;printf('\nP12=-P21=%0.3f pu',P12);
P14=(abs(Ybus(1,4)))*sin(delta(1,1)-delta(4,1));P41=-P14;printf('\nP14=-P41=%0.3f pu',P14);

Q13=abs(Ybus(1,3))-(abs(Ybus(1,3)))*cos(delta(1,1)-delta(3,1));Q31=-Q13;printf('\n\nQ13=-Q31=%0.3f pu',Q13);
Q12=abs(Ybus(1,2))-(abs(Ybus(1,2)))*cos(delta(1,1)-delta(2,1));Q21=-Q12;printf('\nQ12=-Q21=%0.3f pu',Q12);
Q14=abs(Ybus(1,4))-(abs(Ybus(1,4)))*cos(delta(1,1)-delta(4,1));Q41=-Q14;printf('\nQ14=-Q41=%0.3f pu',Q14);
```

**Scilab code Exa 6.4** Bus voltages using GS iterations

```
//Chapter 6
//Example 6.4
//page 209
//To find bus voltages using GS iterations
clear;clc;

//Ybus matrix from the network
Ybus=[3-9*%i -2+6*%i -1+3*%i 0;
      -2+6*%i 3.666-11*%i -0.666+2*%i -1+3*%i
      -1+3*%i -0.666+2*%i 3.666-11*%i -2+6*%i
      0 -1+3*%i -2+6*%i 3-9*%i]

//////////////////////////////////////////////////////

//Pi        Qi        Vi          Remarks         Bus no     //
```

```
P1=0;      Q1=0;     V1=1.04;     //Slack bus     1
P2=0.5;    Q2=-0.2;  V2=1;        //PQbus         2
P3=-1.0;   Q3=0.5;   V3=1;        //PQbus         3
P4=0.3;    Q4=-0.1;  V4=1;        //PQbus         4
/////////////////////////////////////////////////////

n=1;
for i=1:n
    V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-Ybus(2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
    V3=(1/Ybus(3,3))*(((P3-%i*Q3)/conj(V3))-Ybus(3,1)*V1-Ybus(3,2)*V2-Ybus(3,4)*V4);
    V4=(1/Ybus(4,4))*(((P4-%i*Q4)/conj(V4))-Ybus(4,1)*V1-Ybus(4,2)*V2-Ybus(4,3)*V3);
end

printf('\nAt the end of iteration %d the voltages at the buses are:\n\nV1=',n);disp(V1);printf('pu');
printf('\n\n\nV2=');disp(V2);printf('pu');
printf('\n\n\nV3=');disp(V3);printf('pu');
printf('\n\n\nV4=');disp(V4);printf('pu');
```

**Scilab code Exa 6.5** Reactive power injected using GS iterations

```
//Chapter 6
//Example 6.5
//page 210
//To find bus voltages and Reactive power injected using GS iterations
clear;clc;

```

```
//Ybus matrix from the network
Ybus=[3-9*%i -2+6*%i -1+3*%i 0;
      -2+6*%i 3.666-11*%i -0.666+2*%i -1+3*%i
      -1+3*%i -0.666+2*%i 3.666-11*%i -2+6*%i
      0 -1+3*%i -2+6*%i 3-9*%i]

//Case(i)

//////////////////////////////////////////////////////
//Pi        Qi        Vi          Remarks         Bus no  //
P1=0;       Q1=0;     V1=1.04;    //Slack bus     1
P2=0.5;     Q2=0.2;  V2=1.04;     //PVbus         2
P3=-1.0;    Q3=0.5;   V3=1;       //PQbus         3
P4=0.3;     Q4=-0.1; V4=1;        //PQbus         4
/////////////////////////////////////////////////////
printf('\nCase(i) When 0.2<Q2<1 pu and running for 1 iteration ,we get \n\n');
Q2min=0.2;Q2max=1;
n=1;

for i=1:n
    if Q2<Q2min then
        Q2=Q2min;
        V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-Ybus(2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
     elseif Q2>Q2max then
         Q2=Q2max;
         V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-Ybus(2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
      else
          Q2=-imag(conj(V2)*Ybus(2,1)*V1+conj(V2)*(Ybus(2,2)*V2+Ybus(2,3)*V3+Ybus(2,4)*V4));
          [mag,delta2]=polar((1/Ybus(2,2))*(((P2-%i*Q2)/(conj(V2)))-Ybus(2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4));
          V2=abs(V2)*(cos(delta2)+%i*sin(delta2));
     end
    V3=(1/Ybus(3,3))*(((P3-%i*Q3)/conj(V3))-Ybus(3,1)*V1-Ybus(3,2)*V2-Ybus(3,4)*V4);
    V4=(1/Ybus(4,4))*(((P4-%i*Q4)/conj(V4))-Ybus(4,1)*V1-Ybus(4,2)*V2-Ybus(4,3)*V3);
end

printf('Q2=');disp(Q2);printf('pu');
printf('\n\n\ndelta2=');disp(abs(delta2));printf('rad');
printf('\n\n\nV1=');disp(V1);printf('pu');
printf('\n\n\nV2=');disp(V2);printf('pu');
printf('\n\n\nV3=');disp(V3);printf('pu');
printf('\n\n\nV4=');disp(V4);printf('pu');


// case(ii)

printf('\n\n\nCase(ii) When 0.25<Q2<1 pu and running for 1 iteration,we get \n\n');

////////////////////////////////////////////////////////
//Pi         Qi         Vi          Remarks          Bus no      //
P1=0;      Q1=0;     V1=1.04;      //Slack bus      1
P2=0.5;              V2=1.04;      //PVbus          2
P3=-1.0;   Q3=0.5;   V3=1;         //PQbus          3
P4=0.3;    Q4=-0.1;  V4=1;         //PQbus          4
///////////////////////////////////////////////////////

```

```
Q2min=0.25;Q2max=1;
n=1;

for i=1:n
    if Q2<Q2min then
        Q2=Q2min;
        V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-Ybus(2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
     elseif Q2>Q2max then
         Q2=Q2max;
         V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-Ybus(2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
      else
          Q2=-imag(conj(V2)*Ybus(2,1)*V1+conj(V2)*(Ybus(2,2)*V2+Ybus(2,3)*V3+Ybus(2,4)*V4));
          [mag,delta2]=polar((1/Ybus(2,2))*(((P2-%i*Q2)/(conj(V2)))-Ybus(2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4));
          V2=abs(V2)*(cos(delta2)+%i*sin(delta2));
     end
    V3=(1/Ybus(3,3))*(((P3-%i*Q3)/conj(V3))-Ybus(3,1)*V1-Ybus(3,2)*V2-Ybus(3,4)*V4);
    V4=(1/Ybus(4,4))*(((P4-%i*Q4)/conj(V4))-Ybus(4,1)*V1-Ybus(4,2)*V2-Ybus(4,3)*V3);
end

printf('Q2=');disp(Q2);printf('pu');
printf('\n\n\nV1=');disp(V1);printf('pu');
printf('\n\n\nV2=');disp(V2);printf('pu');
printf('\n\n\nV3=');disp(V3);printf('pu');
printf('\n\n\nV4=');disp(V4);printf('pu');
```

**Scilab code Exa 6.6** Load flow solution using the NR method

```
//Chapter 6
//Example 6.6
//page 218
//To find load flow solution using the NR method
clear;clc;

/////////////////////////////////////////////////////////////////////
//Pd        Qd          Pg         Qg         V           Bus      Type/////
/////////////////////////////////////////////////////////////////////
Pd1=2.0;    Qd1=1.0;    Pg1=0;     Qg1=0;     V1=1.04;        //1     slack bus
Pd2=0;      Qd2=0;      Pg2=0.5;   Qg2=1;     V2=1;           //2     PQ bus
Pd3=1.5;    Qd3=0.6;    Pg3=0.0;   Qg3=0;     V3=1.04;        //3     PV bus
/////////////////////////////////////////////////////////////////////
[V1_mag,V1_ang]=polar(V1);
[V2_mag,V2_ang]=polar(V2);
[V3_mag,V3_ang]=polar(V3);
y_series=1/(0.02+%i*0.08);
y_self=2*y_series;
y_off=-1*y_series;
Ybus=[y_self y_off y_off;y_off y_self y_off;y_off y_off y_self];

[y_bus_mag_21,y_bus_ang_21]=polar(Ybus(2,1));
[y_bus_mag_22,y_bus_ang_22]=polar(Ybus(2,2));
[y_bus_mag_23,y_bus_ang_23]=polar(Ybus(2,3));
[y_bus_mag_31,y_bus_ang_31]=polar(Ybus(3,1));
```

```
[y_bus_mag_32,y_bus_ang_32]=polar(Ybus(3,2));
[y_bus_mag_33,y_bus_ang_33]=polar(Ybus(3,3));
[y_bus_mag_11,y_bus_ang_11]=polar(Ybus(1,1));

//direct computer solution has been found as below
   by running for 3 iterations

n=3;
for i=1:n
//from eq.6.27 and 6.28
P2=V2_mag*V1_mag*y_bus_mag_21*cos(y_bus_ang_21+
   V1_ang-V2_ang)+(V2_mag^2)*y_bus_mag_22*cos(
   y_bus_ang_22)+V2_mag*V3_mag*y_bus_mag_23*cos(
   y_bus_ang_23+V3_ang-V2_ang);

P3=V3_mag*V1_mag*y_bus_mag_31*cos(y_bus_ang_31+
   V1_ang-V3_ang)+(V3_mag^2)*y_bus_mag_33*cos(
   y_bus_ang_33)+V2_mag*V3_mag*y_bus_mag_32*cos(
   y_bus_ang_32+V2_ang-V3_ang);

Q2=-V2_mag*V1_mag*y_bus_mag_21*sin(y_bus_ang_21+
   V1_ang-V2_ang)-(V2_mag^2)*y_bus_mag_22*sin(
   y_bus_ang_22)-V2_mag*V3_mag*y_bus_mag_23*sin(
   y_bus_ang_23+V3_ang-V2_ang);

P2=real(P2);
P3=real(P3);
Q2=real(Q2);

delta_P2=(Pg2-Pd2)-(P2);
delta_P3=(Pg3-Pd3)-(P3);
delta_P2=(Pg2-Pd2)-(P2);
delta_Q2=(Qg2-Qd2)-(Q2);

//forming jacobian matrix by differentiating
   expressions of P2,P3,Q2
j11=V2_mag*V1_mag*y_bus_mag_21*sin(y_bus_ang_21+
   V1_ang-V2_ang)+V2_mag*V3_mag*y_bus_mag_23*sin(
y_bus_ang_23+V3_ang-V2_ang);
j12=-V2_mag*V3_mag*y_bus_mag_23*sin(y_bus_ang_23+V3_ang-V2_ang);
j13=V1_mag*y_bus_mag_21*cos(y_bus_ang_21+V1_ang-V2_ang)+(V2_mag*2)*y_bus_mag_22*cos(y_bus_ang_22)+V3_mag*y_bus_mag_23*cos(y_bus_ang_23+V3_ang-V2_ang);

j21=-V2_mag*V3_mag*y_bus_mag_32*sin(y_bus_ang_32+V2_ang-V3_ang);
j22=V3_mag*V1_mag*y_bus_mag_31*sin(y_bus_ang_31+V1_ang-V3_ang)+V2_mag*V3_mag*y_bus_mag_32*sin(y_bus_ang_32+V2_ang-V3_ang);
j23=V3_mag*y_bus_mag_32*cos(y_bus_ang_32+V2_ang-V3_ang);

j31=V2_mag*V1_mag*y_bus_mag_21*cos(y_bus_ang_21+V1_ang-V2_ang)+V2_mag*V3_mag*y_bus_mag_23*cos(y_bus_ang_23+V3_ang-V2_ang);
j32=-V2_mag*V3_mag*y_bus_mag_23*cos(y_bus_ang_23+V3_ang-V2_ang);
j33=-V1_mag*y_bus_mag_21*sin(y_bus_ang_21+V1_ang-V2_ang)-(V2_mag*2)*y_bus_mag_22*sin(y_bus_ang_22)-V3_mag*y_bus_mag_23*sin(y_bus_ang_23+V3_ang-V2_ang);

J=[j11 j12 j13;j21 j22 j23;j31 j32 j33];
J=real(J);

//power residuals
PR=[delta_P2;delta_P3;delta_Q2];

//changes in variables
ch_var=inv(J)*PR;

V2_ang=V2_ang+ch_var(1,1);
V3_ang=V3_ang+ch_var(2,1);
V2_mag=V2_mag+ch_var(3,1);
```

```

P1=(V1_mag^2)*y_bus_mag_11*cos(y_bus_ang_11)+V1_mag*V2_mag*y_bus_mag_21*cos(y_bus_ang_21+V2_ang-V1_ang)+V1_mag*V3_mag*y_bus_mag_31*cos(y_bus_ang_31+V3_ang-V1_ang);
Q1=-V1_mag^2*y_bus_mag_11*sin(y_bus_ang_11)-V1_mag*V2_mag*y_bus_mag_21*sin(y_bus_ang_21+V2_ang-V1_ang)-V1_mag*V3_mag*y_bus_mag_31*sin(y_bus_ang_31+V3_ang-V1_ang);

Q3=-V3_mag*V1_mag*y_bus_mag_31*sin(y_bus_ang_31+V1_ang-V3_ang)-(V3_mag^2)*y_bus_mag_33*sin(y_bus_ang_33)-V2_mag*V3_mag*y_bus_mag_32*sin(y_bus_ang_32+V2_ang-V3_ang);
Qg3=Q3+Qd3;

end

S1=real(P1)+%i*real(Q1);
S2=P2+%i*Q2;
S3=P3+%i*Q3;

printf('\nThe final results are given below:\n');
printf('V2=%0.3f @ %0.3f rad\n',V2_mag,V2_ang);
printf('V3=%0.3f @ %0.3f rad\n',V3_mag,V3_ang);
printf('Qg3=%0.2f pu(with in limits)\n',Qg3);
printf('\nS1=');disp(S1);printf('pu');
printf('\n\nS2=');disp(S2);printf("pu");
printf('\n\nS3=');disp(S3);printf("pu");
printf('\n\nTransmission losses=%0.3f pu\n',(real(P1)+P2+P3));

//Line Flows

//V_mag=[V1_mag V2_mag V3_mag];
//V_ang=[V1_ang V2_ang V3_ang];
v1=V1_mag*(cos(V1_ang)+%i*sin(V1_ang));
v2=V2_mag*(cos(V2_ang)+%i*sin(V2_ang));
```

```
v3=V3_mag*(cos(V3_ang)+%i*sin(V3_ang));
V=[v1 v2 v3];
for i=1:3
    for j=1:3
        s(i,j)=conj(V(i))*(V(i)-V(j))*(2.941-%i*11.764)+conj(V(i))*V(i)*(%i*0.01);
        s(j,i)=conj(V(j))*(V(j)-V(i))*(2.941-%i*11.764)+conj(V(j))*V(j)*(%i*0.01);
     end
end
P=real(s);
Q=-imag(s);
printf('\nLine Flows\nThe following matrix shows the real part of line flows(in pu)');disp(P);
printf('\nThe following matrix shows the imaginary part of line flows(in pu)');disp(Q);
```

**Scilab code Exa 6.7** Ybus after including regulating transformer

```
//Chapter 6
//Example 6.7
//page 234
//To find modified Ybus after including regulating transformer
clear;clc;

y34=2-%i*6;y23=0.666-%i*2;
y12=2-%i*6;y24=1-%i*3;
y13=1-%i*3;

//case(i) when a=1/1.04;
a=1/1.04;
//to form Ybus matrix
```

```
Y11=y13+y12;Y12=-y12;Y13=-y13;Y14=0;
Y21=-y12;Y22=y12+y23+y24;Y23=-y23;Y24=-y24;
Y31=-y13;Y32=-y23;Y33=(a^2)*y34+y23+y13;Y34=-(a')*y34;
Y41=0;Y42=-y24;Y43=-a'*y34;Y44=y34+y24;


Ybus=[Y11 Y12 Y13 Y14;
      Y21 Y22 Y23 Y24;
      Y31 Y32 Y33 Y34;
      Y41 Y42 Y43 Y44];
printf('Case(i) When a=1/1.04');
printf('\nYbus=');disp(Ybus);
printf('\nObserve the changes in elements between bus 3&4 when compared with the result of example_6.2');

//case(ii) when a=e^(-j3)

a=cosd(3)-%i*sind(3);
//to form Ybus matrix
Y11=y13+y12;Y12=-y12;Y13=-y13;Y14=0;
Y21=-y12;Y22=y12+y23+y24;Y23=-y23;Y24=-y24;
Y31=-y13;Y32=-y23;Y33=(abs(a)^2)*y34+y23+y13;Y34=(a')*(-y34);
Y41=0;Y42=-y24;Y43=a*(-y34);Y44=y34+y24;


Ybus=[Y11 Y12 Y13 Y14;
      Y21 Y22 Y23 Y24;
      Y31 Y32 Y33 Y34;
      Y41 Y42 Y43 Y44];
printf('\n\nCase(ii) When a=e^(-j3)');
printf('\nYbus=');disp(Ybus);
printf('\nObserve the changes in elements between bus 3&4 when compared with the result of example_6.2');
```

**Scilab code Exa 6.8** Decoupled NR method and FDLF method

```
//Chapter 6
//Example 6.8
//page 226
//To find load flow solution using the decoupled NR method and FDLF method
clear;clc;

////////////////////////////////////////////////////////////////////
//Pd          Qd             Pg          Qg          V                Bus     Type/////
////////////////////////////////////////////////////////////////////
Pd1=2.0;    Qd1=1.0;    Pg1=0;    Qg1=0;    V1=1.04;        //1     slack bus
Pd2=0;      Qd2=0;      Pg2=0.5;  Qg2=1;    V2=1;           //2     PQ bus
Pd3=1.5;    Qd3=0.6;    Pg3=0.0;  Qg3=0;    V3=1.04;        //3     PV bus
////////////////////////////////////////////////////////////////////
[V1_mag,V1_ang]=polar(V1);
[V2_mag,V2_ang]=polar(V2);
[V3_mag,V3_ang]=polar(V3);
y_series=1/(0.02+%i*0.08);
y_self=2*y_series;
y_off=-1*y_series;
```

```
Ybus=[y_self y_off y_off;y_off y_self y_off;y_off y_off y_self];

[y_bus_mag_21,y_bus_ang_21]=polar(Ybus(2,1));
[y_bus_mag_22,y_bus_ang_22]=polar(Ybus(2,2));
[y_bus_mag_23,y_bus_ang_23]=polar(Ybus(2,3));
[y_bus_mag_31,y_bus_ang_31]=polar(Ybus(3,1));
[y_bus_mag_32,y_bus_ang_32]=polar(Ybus(3,2));
[y_bus_mag_33,y_bus_ang_33]=polar(Ybus(3,3));
[y_bus_mag_11,y_bus_ang_11]=polar(Ybus(1,1));

//case(a) Decoupled NR method :
printf('\ncase(a) Decoupled NR method :\n') ;

H22=0.96+23.508;
H23=-1.04*11.764;
H33=25.89;
L22=1+23.508;
H=[H22 H23;H23 H33];
delta_P=[0.73;-1.62];

delta_V_ang=inv(H)*delta_P;
delta_V2_ang=delta_V_ang(1,1);
delta_V3_ang=delta_V_ang(2,1);
printf('\ndelta_Angle_V2=');disp(real(delta_V2_ang));
printf('\ndelta_Angle_V3=');disp(real(delta_V3_ang));
V2_ang=V2_ang-delta_V2_ang;
V3_ang=V3_ang-delta_V3_ang;

Q2=-V2_mag*V1_mag*y_bus_mag_21*sin(y_bus_ang_21+V1_ang-V2_ang)-(V2_mag^2)*y_bus_mag_22*sin(y_bus_ang_22)-V2_mag*V3_mag*y_bus_mag_23*sin(y_bus_ang_23-V3_ang+V2_ang);

printf('\nQ2=');disp(real(Q2));
delta_Q2=(Qg2-Qd2)-(Q2);
```

```
printf('\ndelta_Q2=');disp(real(delta_Q2));
L=[L22];
delta_v=inv(L)*delta_Q2;
delta_V2=delta_v*V2_mag;

printf('\ndelta_V2=%0.3f',delta_V2);
V2_mag=V2_mag+delta_V2;
printf('\n\nV2=%0.3f pu',V2_mag);

Q3=-V3_mag*V1_mag*y_bus_mag_31*sin(y_bus_ang_31+
   V1_ang-V3_ang)-(V3_mag^2)*y_bus_mag_33*sin(
   y_bus_ang_33)-V2_mag*V3_mag*y_bus_mag_32*sin(
   y_bus_ang_32+V2_ang-V3_ang);

printf('\n\nQ3=');disp(real(Q3));

//case(b) FDLF method:

printf('\n\n\ncase(b) FDLF method :\n') ;

//
   /////////////////////////////////////////////////////////////////

//Pd          Qd           Pg          Qg          V
                Bus      Type/////
//
   /////////////////////////////////////////////////////////////////

Pd1=2.0;    Qd1=1.0;    Pg1=0;     Qg1=0;     V1=1.04;
          //1     slack bus
Pd2=0;      Qd2=0;      Pg2=0.5;   Qg2=1;     V2=1;
             //2     PQ bus
Pd3=1.5;    Qd3=0.6;    Pg3=0.0;   Qg3=0;     V3=1.04;
          //3     PV bus
//
   /////////////////////////////////////////////////////////////////

[V1_mag,V1_ang]=polar(V1);
```

```
[V2_mag,V2_ang]=polar(V2);
[V3_mag,V3_ang]=polar(V3);
y_series=1/(0.02+%i*0.08);
y_self=2*y_series;
y_off=-1*y_series;
Ybus=[y_self y_off y_off;y_off y_self y_off;y_off y_off y_self];

[y_bus_mag_21,y_bus_ang_21]=polar(Ybus(2,1));
[y_bus_mag_22,y_bus_ang_22]=polar(Ybus(2,2));
[y_bus_mag_23,y_bus_ang_23]=polar(Ybus(2,3));
[y_bus_mag_31,y_bus_ang_31]=polar(Ybus(3,1));
[y_bus_mag_32,y_bus_ang_32]=polar(Ybus(3,2));
[y_bus_mag_33,y_bus_ang_33]=polar(Ybus(3,3));
[y_bus_mag_11,y_bus_ang_11]=polar(Ybus(1,1));

B22=-23.508;
B23=11.764;
B32=B23;
B33=B22;

B=[-B22 -B23;-B32 -B33];

delta_P=[0.73;-1.557];

delta_V_ang=inv(B)*delta_P;
delta_V2_ang=delta_V_ang(1,1);
delta_V3_ang=delta_V_ang(2,1);
printf('\ndelta_Angle_V2=');disp(real(delta_V2_ang));
printf('\ndelta_Angle_V3=');disp(real(delta_V3_ang));
V2_ang=V2_ang-delta_V2_ang;
V3_ang=V3_ang-delta_V3_ang;

Q2=-V2_mag*V1_mag*y_bus_mag_21*sin(y_bus_ang_21+V1_ang-V2_ang)-(V2_mag^2)*y_bus_mag_22*sin(y_bus_ang_22)-V2_mag*V3_mag*y_bus_mag_23*sin(
   y_bus_ang_23-V3_ang+V2_ang);

delta_Q2=(Qg2-Qd2)-(Q2);

delta_v=inv([-B22])*delta_Q2;
delta_V2=delta_v*V2_mag;

printf('\ndelta_V2=%0.3f',delta_V2);
V2_mag=V2_mag+delta_V2;
printf('\n\nV2=%0.3f pu',V2_mag);

Q3=-V3_mag*V1_mag*y_bus_mag_31*sin(y_bus_ang_31+
   V1_ang-V3_ang)-(V3_mag^2)*y_bus_mag_33*sin(
   y_bus_ang_33)-V2_mag*V3_mag*y_bus_mag_32*sin(
   y_bus_ang_32+V2_ang-V3_ang);

printf('\n\nQ3=');disp(real(Q3));
```

# Chapter 7

# Optimal System Operation

**Scilab code Exa 7.1** Incremental cost and load sharing

```
//Chapter 7
//Example 7.1
//page 246
//To find incremental cost and load sharing
clear;clc;

///Let us use the program given in the Appendix G in
    the textbook to write
//a function that returns the value of lamda and
   Loading of each generator
//when the total load on the plant is sent to the
   function

function [lamdaprev,Pg]=optimum(Pd)
        n=2; //number of generators
        Alpha=[0.2 0.25];
        Beta=[40 30];
        lamda=35; //initial guess for lambda
        lamdaprev=lamda;
        eps=1; //tolerance
        deltalamda=0.25; //increment in lamda
        Pgmax=[125 125];
        Pgmin=[20 20];
        Pg=100*ones(n,1);
        while abs(sum(Pg)-Pd)>eps
            for i=1:n
                Pg(i)=(lamda-Beta(i))/Alpha(i);
                if Pg(i)>Pgmax(i) then
                    Pg(i)=Pgmax(i);
                end
                if Pg(i)<Pgmin(i) then
                    Pg(i)=Pgmin(i);
                end
           end
           if  (sum(Pg)-Pd)<0 then
               lamdaprev=lamda;
               lamda=lamda+deltalamda;
            else
               lamdaprev=lamda;
               lamda=lamda-deltalamda;
           end
           end
endfunction


//to draw the table 7.1
printf('Table 7.1 Output of each unit and plant output for various values of lamda\n')
printf('---------------------------------------------------------------------------\n');
printf('Plant Lamda,         Unit 1         Unit 2         Plant Output  \n');
printf('Rs/MWh               Pg1,MW         Pg2,MW         (Pg1+Pg2),MW  \n');
printf('---------------------------------------------------------------------------\n');

```

```
Pd_matrix=[40 76 130 150 175 220 231.25 250];
for i=1:8
    [lamda,Pg]=optimum(Pd_matrix(i));
    printf('%0.2f                    %0.2f            %0.2f                %0.2f\n',lamda,Pg(1),Pg(2),Pg(1)+Pg(2));
end
printf('------------------------------------------------------------------------\n');

//To draw the Graphs 7.3 and 7.4

Pd_test=40:3.75:250;
[Pd_ro,Pd_co]=size(Pd_test)
for i=1:Pd_co
   [lamda,Pg]=optimum(Pd_test(i));
   lamda_test(i)=lamda;
   Pg1_test(i)=Pg(1);
   Pg2_test(i)=Pg(2);
end
Pg1_test=Pg1_test.'; //transposing without conjugating
Pg2_test=Pg2_test.';
lamda_test=lamda_test.';

subplot(211)
plot(Pd_test,lamda_test);
title('Incremental Fuel cost versus plant output');
xlabel('Plant output,MW');
ylabel('Incremental fuel cost,Rs/MWh');
set(gca(),"grid",[0,0])
get("current_axes");

subplot(212)
plot(Pd_test,Pg1_test,Pd_test,Pg2_test);
title('Output of each unit versus plant output');
xlabel('Plant output,MW');
```

```
ylabel('Unit output,MW');
legend(["Unit 1";"Unit 2"],[2]);
set(gca(),"grid",[0,0])
get("current_axes");
```

**Scilab code Exa 7.2** Savings by optimal scheduling

```
//Chapter 7
//Example 7.2
//page 248
//To find the saving in fuel cost by optimal scheduling
clear;clc;

//Example reveals that for optimal load sharing units 1&2 has to take up 50MW and 80MW respectively
//If each unit supplies 65MW,increase in cost for units 1&2 are

Increase1=integrate('0.2*Pg1+40','Pg1',50,65);
Increase2=integrate('0.25*Pg2+30','Pg2',80,65);
printf('\nIncrease in cost for unit 1 is = %0.1f Rs/hr',Increase1);
printf('\n\nIncrease in cost for unit 2 is = %0.3f Rs/hr',Increase2);
printf('\n\nNet saving caused by optimum scheduling is = %0.3f Rs/hr',Increase1+Increase2);
printf('\n\nTotal yearly saving assuming continuous operation= Rs %d',(Increase1+Increase2)*24*365);
```

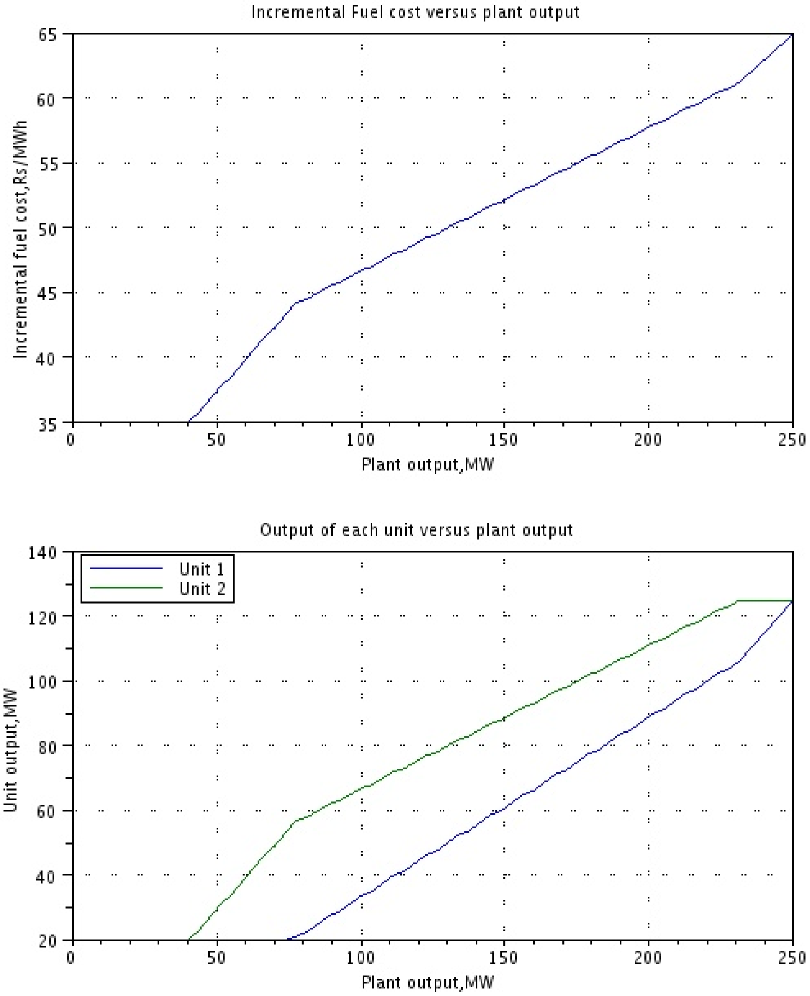


Figure 7.1: Incremental cost and load sharing

**Scilab code Exa 7.3** Economical operation

```
//Chapter 7
//Example 7.3
//page 249
//To find the economical operation
clear;clc;

//from the table we got as the output in the example_7_1
//for optimum operation of load 220MW, unit 1&2 must be loaded 100MW and 120MW respwctively
//and for a load of 76MW, unit 1&2 must be loaded 20 MW and 56MW respwctively
start_up=400;
//case(i)
printf('\nCase(i)');
//total fuel cost for the load of 220MW during 6AM to 6PM
Pg1=100;
Pg2=120;
C1=0.1*Pg1^2+40*Pg1+120;
C2=0.125*Pg2^2+30*Pg2+100;
total1=(C1+C2)*12;
printf('\nTotal fuel cost for the load of 220MW during 6AM to 6PM = Rs. %d',total1);

//total fuel cost for the load of 76MW during 6PM to 6AM
Pg1=20;
Pg2=56;
C1=0.1*Pg1^2+40*Pg1+120;
```

```
C2=0.125*Pg2^2+30*Pg2+100;
total2=(C1+C2)*12;
printf('\nTotal fuel cost for the load of 76MW during 6PM to 6AM if both the units run = Rs. %d',total2);

total=total1+total2; //total fuel cost for 24hrs

printf('\nTotal fuel cost for the load during 24hrs if both the units run = Rs. %d',total);

//case(ii)
printf('\n\nCase(ii)');
//If during light load condition unit2 is On and Unit1 is Off then
Pg2=76;
C2=0.125*Pg2^2+30*Pg2+100;
total2=C2*12;
total_case2=total1+total2+start_up;

printf('\nTotal fuel cost for the 24hrs laod if only unit 2 run during light loads is = Rs. %d',total_case2);
```

**Scilab code Exa 7.4** Generation and losses incurred

```
//Chapter 7
//Example 7.4
//page 263
//To find required generation for each plant and losses incurred
clear;clc;

```

```
///Let us use the program given in the Appendix G in the textbook which includes penalty factor also to write
//a function that returns the value of lamda,Loading of each generator and losses
//when the total load on the plant is sent to the function

function [lamda,Pg,PL]=optimum2(Pd)
n=2; //no of generators
Alpha=[0.02 0.04];
Beta=[16 20];
lamda=20; //initial value of lamda
lamdaprev=lamda;
eps=1; //tolerance
deltalamda=0.1;
Pgmax=[200 200];
Pgmin=[0 0];
B=[0.001 0;0 0];
Pg=zeros(n,1);
noofiter=0;
PL=0;
Pg=zeros(n,1);
while abs(sum(Pg)-Pd-PL)>eps
    for i=1:n
        sigma=B(i,:)*Pg-B(i,i)*Pg(i);
        Pg(i)=(1-(Beta(i)/lamda)-(2*sigma))/(Alpha(i)/lamda+2*B(i,i));
        PL=Pg.'*B*Pg;
        if Pg(i)>Pgmax(i) then
            Pg(i)=Pgmax(i);
        end
        if Pg(i)<Pgmin(i) then
            Pg(i)=Pgmin(i);
        end
    end
    PL=Pg.'*B*Pg;
    if(sum(Pg)-Pd-PL)<0 then
```

```
        lamdaprev=lamda;
        lamda=lamda+deltalamda;
    else
        lamdaprev=lamda;
        lamda=lamda-deltalamda;
    end
    noofiter=noofiter+1;
    Pg;
end
endfunction

//In this example let us take the answer .i.e load(Pd)=237.04MW and calculate
//lamda so that we can use the algorithm used in the textbook
Pd=237.04
[lamda_test,Pg_test,PL_test]=optimum2(Pd);
printf('\nLagrange''s multiplier (lamda) is\n Lamda=%0.1f',lamda_test);
printf('\n\nRequired generation for optimum loading are \n Pg1=%0.2f MW \n Pg2=%d MW\n',Pg_test(1),Pg_test(2));
printf('\nThe transmission power loss is\n PL=%0.2f MW',PL_test);
printf('\n\nThe load is \n Pd=%0.2f MW',Pd);
```

**Scilab code Exa 7.5** Savings on coordination of losses

```
//Chapter 7
//Example 7.5
//page 264
//To find savings when losses are coordinated
clear;clc;
```

```

function [lamdaprev,Pg]=optimum(Pd)
        n=2; //number of generators
        Alpha=[0.02 0.04];
        Beta=[16 20];
        lamda=20; //initial guess for lambda
        lamdaprev=lamda;
        eps=1; //tolerance
        deltalamda=0.25; //increment in lamda
        Pgmax=[200 200];
        Pgmin=[0 0];
        Pg=100*ones(n,1);
        while abs(sum(Pg)-Pd)>eps
            for i=1:n
                Pg(i)=(lamda-Beta(i))/Alpha(i);
                if Pg(i)>Pgmax(i) then
                    Pg(i)=Pgmax(i);
                end
                if Pg(i)<Pgmin(i) then
                    Pg(i)=Pgmin(i);
                end
           end
           if  (sum(Pg)-Pd)<0 then
               lamdaprev=lamda;
               lamda=lamda+deltalamda;
            else
               lamdaprev=lamda;
               lamda=lamda-deltalamda;
           end
           end
endfunction

//the above function "optimum" doesn't coordinate losses

//case(i) when the losses are included but not coordinated
[lamda_case1,Pg_case1]=optimum(237.04);
```

```
//since Pg2 does not supply transmission losses and the losses are supplied only by Pg1
Pg2_1=Pg_case1(2);
//to get Pg1 we will solve Pg1+Pg2=0.001*Pg1^2+237.04
//the above equation can be wriiten as (0.001*Pg1^2) - Pg1 +(237.04-Pg2) =0
p=poly([0.001 -1 (237.04+Pg2_1)],"Pg1");
Pg1_1=roots(p);
Pg1_1=Pg1_1(1);

printf('\ncase(i) when the losses are included but not coordinated');
printf('\nPg1=%0.2f MW    Pg2=%0.2f MW',Pg1_1,Pg2_1);

//case(ii) when the losses are also coordinated
//we have the solution for case(ii) from example_7_4
Pg1_2=128.57; Pg2_2=125; //case(ii)

printf('\n\ncase(ii) when the losses are coordinated');
printf('\nPg1=%0.2f MW    Pg2=%0.2f MW',Pg1_2,Pg2_2);

//saving at plant 1 is
saving1=integrate('0.02*Pg1+16','Pg1',Pg1_2,Pg1_1);
printf('\n\nSaving at plant 1 due to loss coordination is = Rs %0.2f/hr',saving1);

//saving at plant 2 is
saving2=integrate('0.04*Pg2+20','Pg2',Pg2_2,Pg2_1);
printf('\n\nSaving at plant 2 due to loss coordination is = Rs %0.2f/hr',saving2);

//net savings achieved
printf('\n\nThe net saving achieved by coordinating losses while scheduling the recieved load of
    237.04MW is Rs %0.2f/hr',saving1+saving2);
```

**Scilab code Exa 7.6** Loss formula coefficients calculation

```
//Chapter 7
//Example 7.6
//page 268
//To calculate the loss formula coefficients of the
   system
clear;clc;

Ia=2-%i*0.5;     Ic=1-%i*0.25;
Ib=1.6-%i*0.4;     Id=3.6-%i*0.9;
Za=0.015+%i*0.06;     Zc=0.01+%i*0.04;
Zb=0.015+%i*0.06;     Zd=0.01+%i*0.04;

ID=Id+Ic ;//total load current

//calculation of current distribution factors
printf('\nCurrent distribution factors are :\n')
Ma1=(ID/ID);
Ma2=(0/ID);
Mb1=(-Ic/ID);
Mb2=(Id/ID);
Mc1=(Ic/ID);
Mc2=(Ic/ID);
Md1=(Id/ID);
Md2=(Id/ID);
printf('Ma1=%d\tMb1=%0.4f\tMc1=%0.4f\tMd1=%0.4f\nMa2
   =%d\tMb2=%0.4f\tMc2=%0.4f\tMd2=%0.4f',Ma1,Mb1,Mc1
   ,Md1,Ma2,Mb2,Mc2,Md2);

//bus voltage calcultion
```

```
[V1_mag,V1_ang]=polar(1.0+Ia*Za);
[V2_mag,V2_ang]=polar(1+Ib*Zb);
V1_ang=real(V1_ang)*180/%pi;
V2_ang=real(V2_ang)*180/%pi;
printf('\n\nBus voltages are given by \nV1=%0.3f @ %0.2fdeg PU\tV2=%0.3f @ %0.2fdeg PU',V1_mag,V1_ang,V2_mag,V2_ang);

//current phase angles at the plants
sigma1=atand(imag(Ia)/real(Ia));
sigma2=atand(imag(Ib+Ic)/real(Ib+Ic));
printf('\n\nCurrent phase angles at the plants\nSigma1=%ddeg\tSigma2=%ddeg',sigma1,sigma2);

//plant power factors
pf1=cosd(V1_ang-sigma1);
pf2=cosd(V2_ang-sigma2);
printf('\n\nThe plant power factors are\npf1=%0.4f\tpf2=%0.4f',pf1,pf2);

//calculation of loss coefficients
B11=(Ma1*Ma1*real(Za)+Mb1*Mb1*real(Zb)+Mc1*Mc1*real(Zc)+Md1*Md1*real(Zd))/(V1_mag*V1_mag*pf1*pf1);
B22=(Ma2*Ma2*real(Za)+Mb2*Mb2*real(Zb)+Mc2*Mc2*real(Zc)+Md2*Md2*real(Zd))/(V2_mag*V2_mag*pf2*pf2);
B12=(Ma1*Ma2*real(Za)+Mb1*Mb2*real(Zb)+Mc1*Mc2*real(Zc)+Md1*Md2*real(Zd))/(V1_mag*V2_mag*pf1*pf2);
printf('\n\nThe Loss coefficients in PU are \nB11=%0.5f pu\nB22=%0.5f pu\nB12=%0.5f pu',B11,B22,B12);
printf('\n\nThe Loss coefficients in reciprocal megawatts are \nB11=%0.8f MW^-1\nB22=%0.8f MW^-1\nB12=%0.8f MW^-1',B11/100,B22/100,B12/100);
```

**Scilab code Exa 7.7** Optimal generation schedule for hydrothermal system

```
//Chapter 7
//Example 7.7
//page 281
//To find the optimal generation schedule for a
   typical day of the fundamental hydrothermal
   system
clear;clc;

h_b=20;//basic head of the water
e=0.005; //head correction factor
r=2; //non-effective water discharge
Pd_1=7; Pd_2=10; Pd_3=5; //load at three intervals
   of time during a day
alpha=0.5;//positive scalar
X_0=100; //initial water storage in the reservoir
X_3=60; //final water storage in the reservoir
//let us assume the initial values of the control
   variables
q_2=15;
q_3=15;
i=0; //iteration count
grad_2=1;grad_3=1; //inital value for iterations

while ((grad_2>0.1)|(grad_3>0.1))

//water discharge in the first interval
q_1=X_0-X_3-(q_2+q_3);

//water level after the first intervals are
X_1=X_0-q_1;
X_2=X_1-q_2;

//hydro generations in the subintervals
Pgh_1=9.81*(10^-3)*20*(1+0.5*e*(X_1+X_0))*(q_1-r);
Pgh_2=9.81*(10^-3)*20*(1+0.5*e*(X_2+X_1))*(q_2-r);
```

```
Pgh_3=9.81*(10^-3)*20*(1+0.5*e*(X_3+X_2))*(q_3-r);

//thermal generation in the three intervals
Pgt_1=Pd_1-Pgh_1;
Pgt_2=Pd_2-Pgh_2;
Pgt_3=Pd_3-Pgh_3;

//calculating lamda_1 for three subintervals
lamda_1_1=Pgt_1+25;
lamda_1_2=Pgt_2+25;
lamda_1_3=Pgt_3+25;

//since we are considering lossless case
lamda_3_1=lamda_1_1;
lamda_3_2=lamda_1_2;
lamda_3_3=lamda_1_3;

//for calculating lamda_2 for three intervals
lamda_2_1=lamda_3_1*9.81*(10^-3)*20*(1+0.5*e*(2*X_0-2*q_1+r));
lamda_2_2=lamda_2_1-lamda_3_1*(0.5*9.81*(10^-3)*20*e*(q_1-r))-lamda_3_2*(0.5*9.81*(10^-3)*20*e*(q_2-r));
lamda_2_3=lamda_2_2-lamda_3_2*(0.5*9.81*(10^-3)*20*e*(q_2-r))-lamda_3_3*(0.5*9.81*(10^-3)*20*e*(q_3-r));

//calculation of gradient vector
grad_2=lamda_2_2-lamda_3_2*9.81*(10^-3)*20*(1+0.5*e*(2*X_1-2*q_2+r));
grad_3=lamda_2_3-lamda_3_3*9.81*(10^-3)*20*(1+0.5*e*(2*X_2-2*q_3+r));

q_2=q_2-alpha*grad_2; //updating value of q and reiterating
q_3=q_3-alpha*grad_3;
i=i+1;
end
```

```

//Hydel and thermal generation for the three sub
    interavals are given in tabular format
printf('\nResults for Optimal Loading of
    Hydrothermal stations at the end of %d iterations
    ',i);
printf('\n
    ----------------------------------------------------------------------------------
    n');
printf('Interval\t\tLoad\t\tHydro\t\tThermal\t\
    tWater discharge\n');
printf('          \t\tMW\t\tMW\t\tMW\t\tm^3/s\n');
printf('
    ----------------------------------------------------------------------------------
    n');
printf('   1     \t\t%d\t\t%0.4f\t\t%0.4f\t\t%0.2f\n'
    ,Pd_1,Pgh_1,Pgt_1,q_1);
printf('   2     \t\t%d\t\t%0.4f\t\t%0.4f\t\t%0.2f\n'
    ,Pd_2,Pgh_2,Pgt_2,q_2);
printf('   3     \t\t%d\t\t%0.4f\t\t%0.4f\t\t%0.2f\n'
    ,Pd_3,Pgh_3,Pgt_3,q_3);
printf('
    ----------------------------------------------------------------------------------
    n');
```

# Chapter 8

# Automatic Generation and Voltage Control

**Scilab code Exa 8.1** Frequency change Calculation

```
//Chapter 8
//Example 8.1
//page 300
//To determine the change in the frequency
clear;clc;
f=50;
H=5e3;
KE=H*100*1000; //K.E stored in the generator
PI=50e6; //power input to generator before the stem valve is closed
EE=PI*0.4 ; //Excess energy input to the rotating parts
fnew=f*((KE+EE)/KE)^0.5; //frequency at the end of the 0.4sec
printf('\nKinetic Energy stored in the rotating parts of generator and turbine = %d kW-sec',KE/1000);
printf('\nExcess power input to generator before the stem valve begins to close=%d MW',PI/1000000);
printf('\nExcess energy input to rotating parts in 0.4sec=%d kW-sec',EE/1000);
printf('\nFrequency at the end of 0.4sec=%0.2f Hz\n\n',fnew);
```

**Scilab code Exa 8.2** Load sharing and System Frequency

```
//Chapter 8
//Example 8.2
//page 301
//To determine determine load sharing and system frequency
clear;clc;
f=50; // system frequency
x=200; //load on first generator(value is assumed first)
delta_f=0.01*x; //from the first equation given in the book
x=(3.75)/(0.01+0.00625); //by substituting (i) in (ii)
delta_f=0.01*x; //recalculating the value
x2=600-x;
printf('\nLoad shared by the generator:\n\tGenerator1=%0.2f MW\n\tGenerator2=%0.2f MW\n',x,x2);
printf('\nSystem Frequency=%0.2f Hz\n\n',f-delta_f);
```

# Chapter 9

# Symmetrical Fault Analysis

**Scilab code Exa 9.1** Fault Current Calculation

```
//Chapter 9
//Example 9.1
//page 335
//To calculate fault current
clear;clc;
//selecting base KVA and MVA
mvab=100;
Gmva=10;
T1mva=10; T2mva=5;
Gkvb=11; //generator kV base
OHLkvb=33; //overhead line kV base
Ckvb=6.6;// cable kB base
xg1=%i*0.15; xg2=%i*0.125; xt1=%i*0.10; xt2=%i*0.08;
xOHL=0.27+%i*0.36 ; xcab= 0.135+%i*0.08;

//clculating PU impedances

xg1=(xg1*mvab)/Gmva;
xg2=(xg2*mvab)/Gmva;
xt1=(xt1*mvab)/T1mva;
xt2=(xt2*mvab)/T2mva;
```

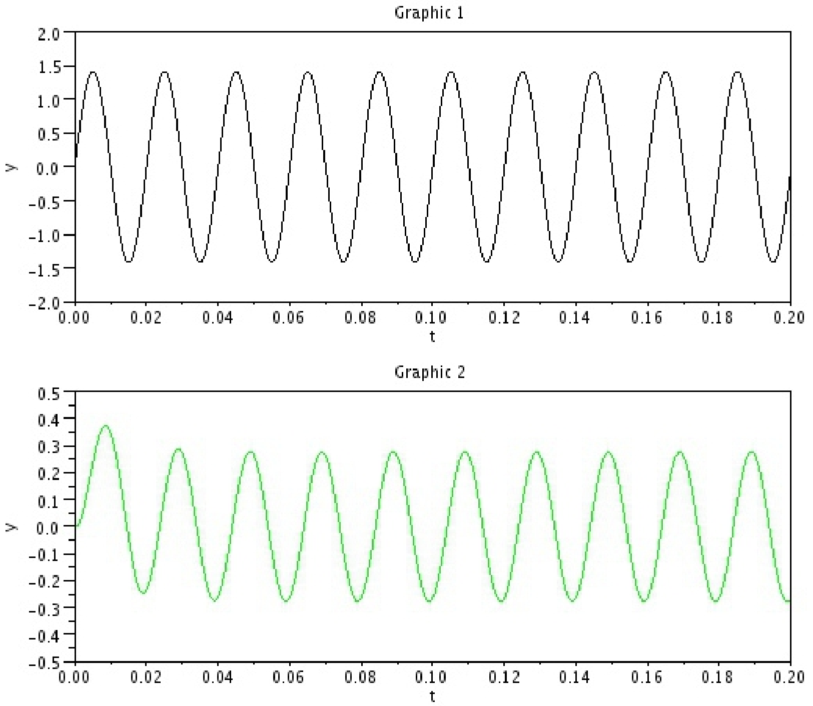
```
xOHL=(30*xOHL*mvab)/(OHLkvb^2);
xcab=(3*xcab*mvab)/(Ckvb^2);
//displaying results
printf('\n Reactance of G1= j%0.1f pu \n',abs(imag(xg1)));
printf(' Reactance of G2= j%0.1f pu\n',abs(imag(xg2)));
printf(' Reactance of T1= j%0.1f pu\n',abs(imag(xt1)));
printf(' Reactance of T2= j%0.1f pu\n',abs(imag(xt2)));
printf(' Overhead line impedance=(%0.3f + j%0.3f) pu\n',real(xOHL),abs(imag(xOHL)));
printf(' Cable impedance= (%0.3f + j%0.3f) pu\n',real(xcab),abs(imag(xcab)));

// Impedance diagram is as shown in the figure9.7 in the textbook
// A XCOS simulation for this proble is done to explain the subtransient,transient and steady state periods of a symmetrical short circuit
xtotal=((xg1*xg2)/(xg1+xg2)+xt1+xt2+xOHL+xcab);
Isc_pu=(1/xtotal);
Ibase=(mvab/(sqrt(3)*Ckvb))*1000;
Isc=Isc_pu*Ibase;
x_F_to_bus=(xt1+xt2+xOHL+xcab);
v_11b=x_F_to_bus*Isc_pu*11;
//displaying results
printf('\nTotal impedance= %0.1f < %0.2f deg pu \n',abs(xtotal),atand(imag(xtotal)/real(xtotal)));
printf('Short circuit current= %d A\n',abs(Isc));
printf('Voltage at 11kV bus=%0.2f kV\n',abs(v_11b));
```

This code can be downloaded from the website wwww.scilab.in

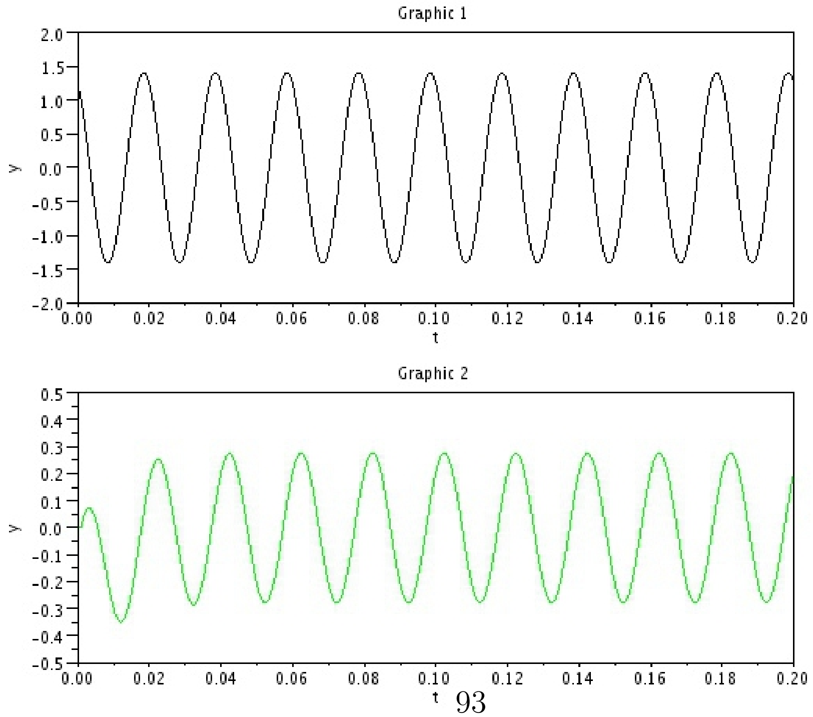
**Scilab code Exa 9.2** Subtransient and Momentary current Calculation

```
//Chapter 9
//Example 9.2
//page 337
//To calculate subtransient and momentary current
clear;clc;
mvab=25;
Gmva=25;
T1mva=25; T2mva=25;
Gkvb=11; //generator kV base
OHLkvb=66; //overhead line kV base
Mkvb=6.6; //motor kV base
Mmva=5; //motor mva

XdG=%i*0.2; //Generator's subtransient reactance
XdM=%i*0.25; //Motor's subtransient reactance
XdM2=%i*0.3; //Motor's transient reactance
Xt1=%i*0.1; // step up transformer's reactance
Xt2=%i*0.1;//step down transformer's reactance
Xtl=%i*0.15 ;//trnasmission line's reactance

//per unit calculation
XdM=(XdM*mvab)/Mmva ;//perunit impedance of each motor
printf('\nSubtransient reactance of each motor = j%0.2f pu\n',abs(XdM));
```

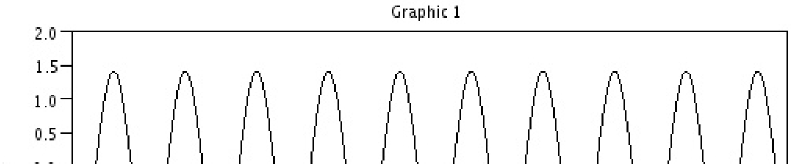
R-Phase Voltage and Fault Current



Y-Phase Voltage and Fault Current



B-Phase Voltage and Fault Current



```

//(a)subtransient current in the fault
Isc=(3*(1/XdM))+(1/(XdG+Xt1+Xt2+Xtl));
Ibase=(mvab*1000)/(sqrt(3)*Mkvb);
Isc=Isc*Ibase;
printf('\nSubtransient current in the fault =%0.1fA\n',abs(Isc));

//(b)subtransient current in the breaker B
IscB=(2*(1/XdM))+(1/(XdG+Xt1+Xt2+Xtl));
IscB=IscB*Ibase;
printf('\nSubtransient current in breaker B=%0.1fA\n',abs(IscB));

//(c) to find the momentary current through breaker B
ImomB=1.6*IscB;
printf('\nMomentary current through the breaker B=%dA\n',abs(ImomB));

//(d) to compute current to be interrupted by breaker in 5 cycles
XdM2=(XdM2*mvab)/Mmva ;//perunit transient impedance of each motor
IscB=(2*(1/XdM2))+(1/(XdG+Xt1+Xt2+Xtl));
IscB=IscB*Ibase;
ImomB=1.1*IscB;
printf('\nCurrent to be interrupted by breaker B in five cycles=%dA\n',abs(ImomB));
```

This code can be downloaded from the website wwww.scilab.in

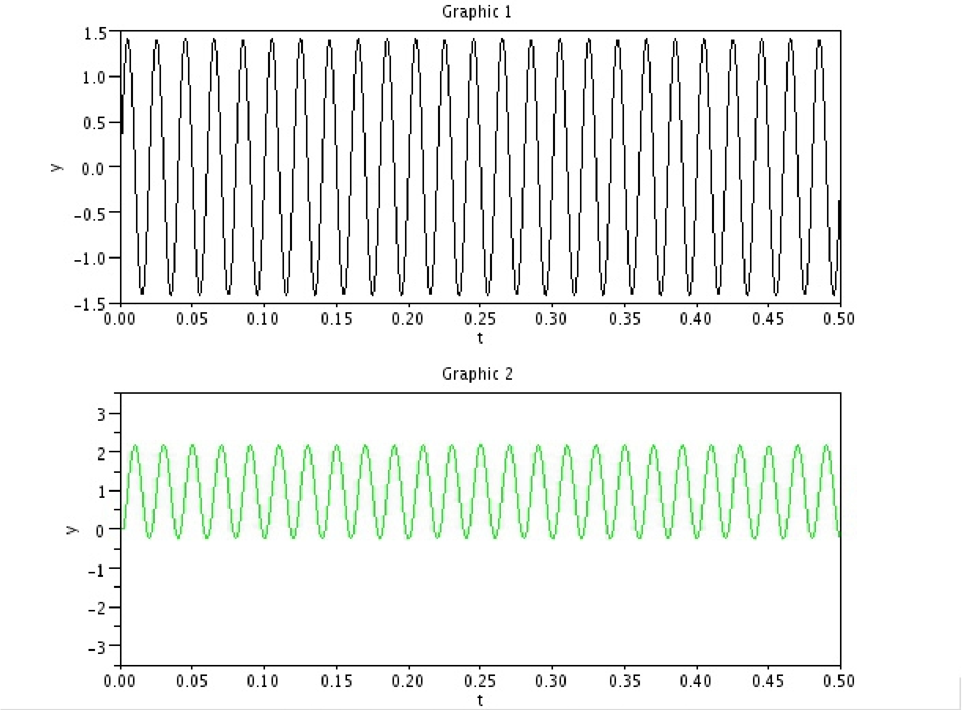


Fig 9.2 : Generator Voltage and Fault current

Figure 9.2: Subtransient and Momentary current Calculation

**Scilab code Exa 9.3** Subtransient Current Calculation

```
//Chapter 9
//Example 9.3
//page 340
//To calculate subtransient current in Generator, Motor and fault
clear;clc;
mvab=25;
kvb=11;
Vo=10.6/kvb; //PU Prefault voltage
printf('\nPrefault Voltage = %0.4fpu\n',Vo);

Load=15/mvab; //load PU with 0.8pf leading
Io=(Load/(Vo*0.8))*(cosd(36.9)+%i*sind(36.9)); //Prefault current
printf('\nPrefault current = %0.4f at %0.1f deg  PU',abs(Io),atand(imag(Io)/real(Io)));

Eg=Vo+(%i*0.45*Io); //voltage behind subtransient reactance(generator)
printf('\n\nVoltage behind subtransient reactance(Generator) = %0.4f+j%0.2f pu\n''',real(Eg),imag(Eg));

Em=Vo-(%i*0.15*Io); //voltage behind subtransient reactance(motor)
printf('\nVoltage behind subtransient reactance(Motor) = %0.4f-j%0.4f pu',real(Em),abs(imag(Em)));

Ig=Eg/(%i*0.45); //under fault condition
Im=Em/(%i*0.15); //under fault condition
printf('\n\nUnder Faulted condition \n Ig””=%0.4f-
   j%0.4f pu',real(Ig),abs(imag(Ig)));
printf('\n Im""=%0.4f-j%0.4f pu',real(Im),abs(imag(
   Im)));
If=Ig+Im; //Current in fault
printf('\n\nCurrent in fault= -j%0.4f pu',abs(imag(
   If)));

Ib=(mvab*1000/(sqrt(3)*11)); //Base current
//Actual Currents
printf("\n\nNow");
Ig=Ig*Ib
Im=Im*Ib
If=If*Ib
printf('\nIg""= %0.1f-j%0.1f A',real(Ig),abs(imag(Ig
   )));
printf('\nIm""= %0.1f-j%0.1f A',real(Im),abs(imag(Im
   )));
printf('\nIf= -j%d A',abs(imag(If)));
```

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 9.4** Maximum MVA Calculation

```
//Chapter 9
//Example 9.4
//page 345
//To calculate maximum MVA
clear;clc;
mvab=50;
kvb=6.6;
```

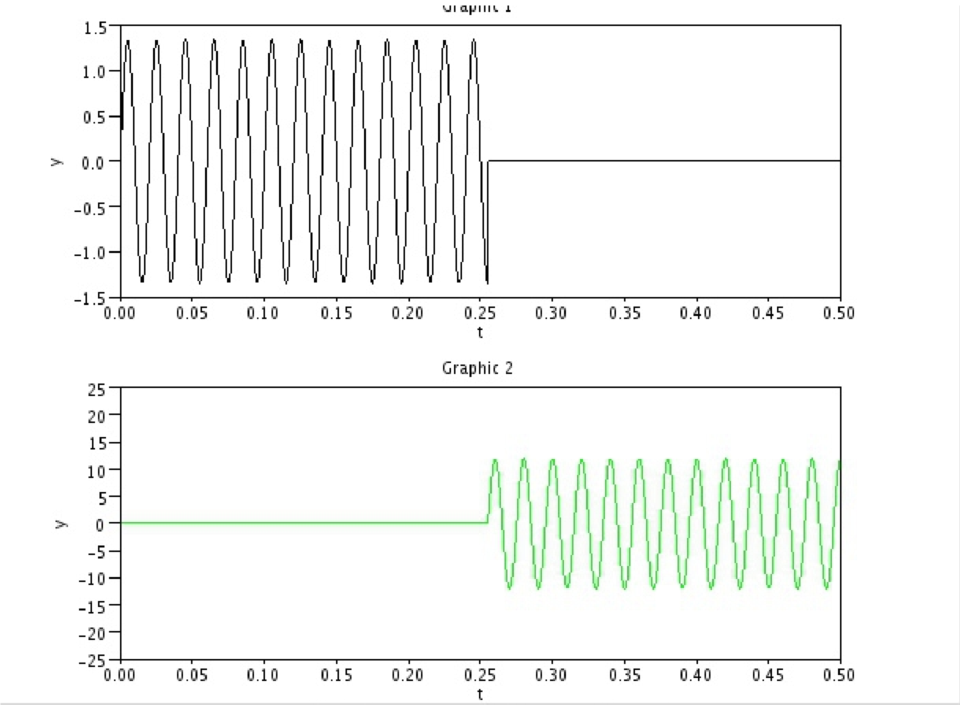


Fig 9.3 : Voltage and Current at the fault

Figure 9.3: Subtransient Current Calculation

```
mvaA=40;
mvaB=50;
mvaC=25;
feeder_impedance=((0.06+%i*0.12)*mvab)/(kvb^2)

Gen_A_reactance=(%i*0.1*mvab/mvaA);
Gen_B_reactance=(%i*0.1*mvab/mvaB);
Gen_C_reactance=(%i*0.1*mvab/mvaC);

printf('\nGenerator A reactance = j%0.3f pu',abs(Gen_A_reactance));
printf('\nGenerator B reactance = j%0.3f pu',abs(Gen_B_reactance));
printf('\nGenerator C reactance = j%0.3f pu',abs(Gen_C_reactance));

Reactor_A_reactance=(%i*0.12*mvab/mvaA);
Reactor_B_reactance=(%i*0.12*mvab/mvaB);
Reactor_C_reactance=(%i*0.12*mvab/mvaC);

printf('\nReactor A reactance = j%0.3f pu',abs(Reactor_A_reactance));
printf('\nReactor B reactance = j%0.3f pu',abs(Reactor_B_reactance));
printf('\nReactor C reactance = j%0.3f pu',abs(Reactor_C_reactance));

function resistance=parallel(r1,r2)
resistance=(r1*r2/(r1+r2));
endfunction

Z=(feeder_impedance)+parallel(%i*0.125,(%i*0.15 + parallel(%i*0.22,%i*0.44)));
scmva=(1/abs(Z))*mvab;
printf("\n\nSC MVA = %d MVA",scmva);
```

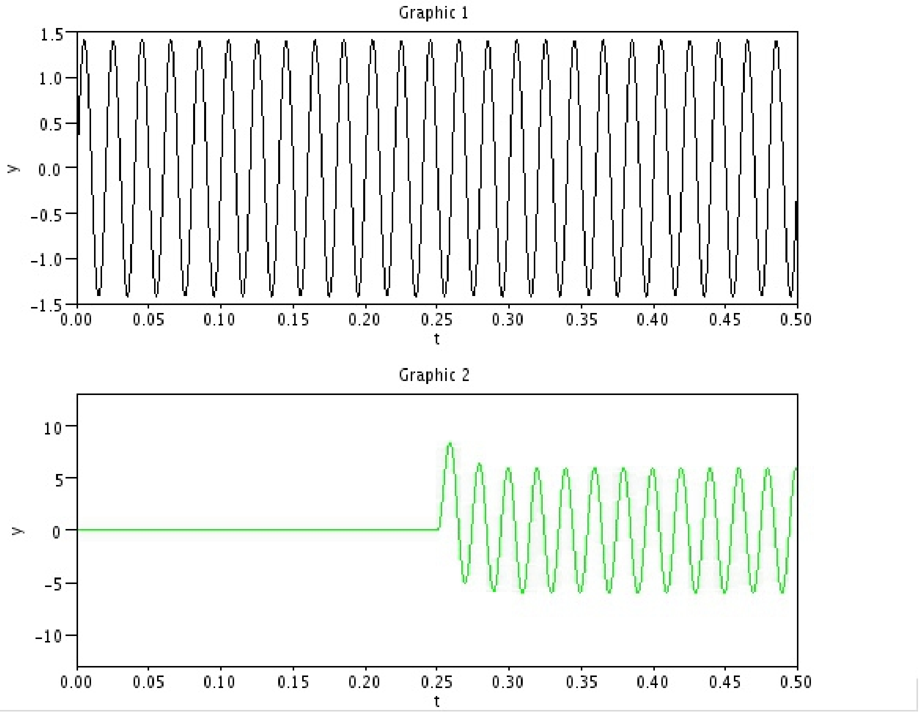


Fig 9.4: Generator Voltage and Fault Current Waveform

Figure 9.4: Maximum MVA Calculation

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 9.5** Short Circuit Solution

```
//Chapter 9
//Example 9.5
//page 347
```

```
//To calculate short circuit solution
clear;clc;
//referring to figures 9.19 in the text book,we get directly the fault current
V4o=1.0;
Zf=%i*0.13560;
If=V4o/Zf;
printf('\nIf= -j%0.5f pu\n\n',abs(If));

//From Fig9.19d
I1=If*((%i*0.19583)/(%i*0.37638));
I2=If*((%i*0.18055)/(%i*0.37638));
printf('I1 = -j%0.5f pu \n\nI2 = -j%0.5f pu\n\n',abs(I1),abs(I2));

//voltage changes for bus 1,2 and 3
deltaV1=0-(%i*0.15)*I1;
deltaV2=0-(%i*0.15)*I2;
printf('DeltaV1=%0.5f pu\n\nDeltaV2=%0.5f pu\n\n',deltaV1,deltaV2);

//reffering to book
V1f=1+deltaV1;
V2f=1+deltaV2;
printf('V1f= %0.5f pu\n\nV2f=%0.5f pu\n\n',V1f,V2f);
I13=(V1f-V2f)/(%i*0.15+%i*0.1);
printf('I13=j%0.5f pu\n\n',abs(I13));
deltaV3=0-((%i*0.15)*(I1)+(%i*0.15)*(I13));
Vf3=1+deltaV3;
printf('DeltaV3=%0.5f pu\n\n',deltaV3);
printf('Vf3=%0.5f pu\n\n',Vf3);
Vf4=0;
printf('Vf4=%d\n\n',Vf4);
//short circuit MVA at bus 4
SC_MVA_4=abs(If)*100;
printf('Short circuit MVA at bus4 =%0.3f MVA',SC_MVA_4);
```

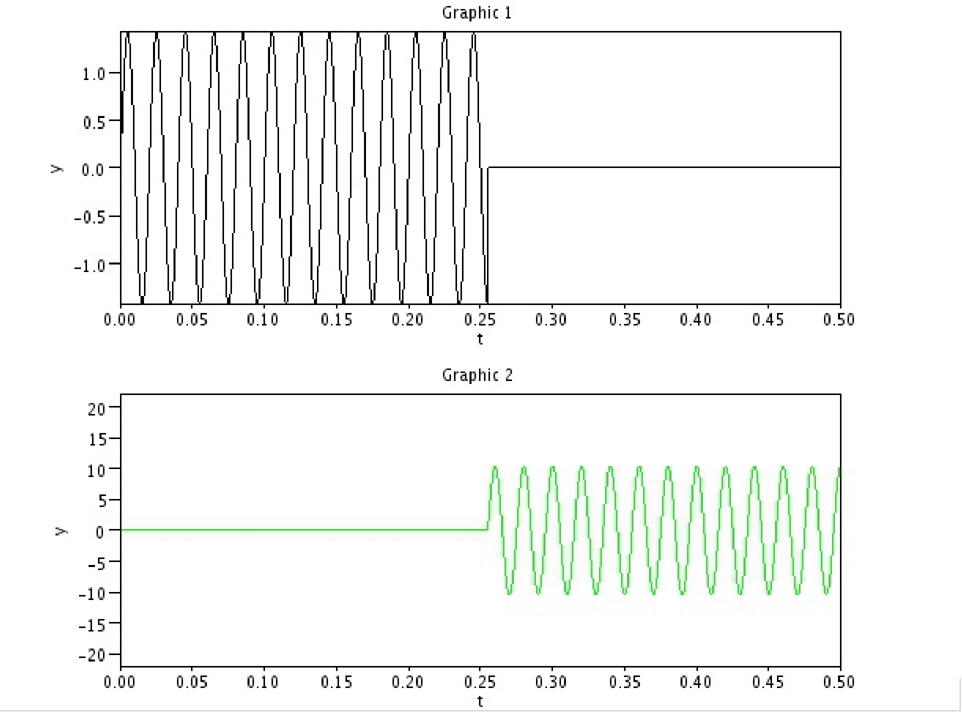


Fig 9.5: Voltage and Current at the Fault

Figure 9.5: Short Circuit Solution

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 9.6** Short Circuit Solution using Algorithm

```
//Chapter 9
//Example 9.6
//page 352
```

```
//To calculate short circuit solution using algorithm for short circuit studies
clear;clc;

Y11=1/(0.15*%i)+1/(0.15*%i)+1/(0.1*%i)+1/(0.2*%i);
Y12=-1/(0.2*%i);
Y21=Y12;
Y13=-1/(0.15*%i);
Y31=Y13;
Y14=-1/(0.1*%i);
Y41=Y14;
Y22=1/(0.15*%i)+1/(0.15*%i)+1/(0.1*%i)+1/(0.2*%i);
Y23=-1/(0.1*%i);
Y32=Y23;
Y24=-1/(0.15*%i);
Y42=Y24;
Y33=1/(0.15*%i)+1/(0.1*%i);
Y34=0;
Y43=Y34;
Y44=1/(0.15*%i)+1/(0.1*%i);

//Ybus matrix can be written as

Ybus=[Y11 Y12 Y13 Y14;Y21 Y22 Y23 Y24;Y31 Y32 Y33 Y34;Y41 Y42 Y43 Y44];

Zbus=inv(Ybus);

//preault voltages
V10=1;V20=1;V30=1;V40=1;

//post fault voltages
V1f=V10-(Zbus(1,4)/Zbus(4,4))*V40;
V2f=V20-(Zbus(2,4)/Zbus(4,4))*V40;
V3f=V30-(Zbus(3,4)/Zbus(4,4))*V40;
V4f=V40-(Zbus(4,4)/Zbus(4,4))*V40;

//to calculate fault current through Zf=0
```

```
If=V40/(Zbus(4,4)+0);

//short circuit current in lines 1-3,1-2,1-4,2-4 and 2-3

I13f=(V1f-V3f)/(0.15*%i);
I12f=(V1f-V2f)/(0.2*%i);
I14f=(V1f-V4f)/(0.1*%i);
I24f=(V2f-V4f)/(0.15*%i);
I23f=(V2f-V3f)/(0.1*%i);

//If at all fault occurs on bus1 or bus2
If12=1/Zbus(1,1);

//displaying the results
printf('\n Ybus=');
disp(Ybus);

printf('\n Zbus=');
disp(Zbus);

printf('\nV1f= %0.4f pu',V1f);
printf('\nV2f= %0.4f pu',V2f);
printf('\nV3f= %0.4f pu',V3f);
printf('\nV4f= %0.1f pu\n',V4f);

printf('\nFault current=-j%0.5f pu\n',abs(If));

printf('\nI13f=j%0.3f pu',abs(I13f));
printf('\nI12f=j%0.3f pu',abs(I12f));
printf('\nI14f=-j%0.3f pu',abs(I14f));
printf('\nI24f=-j%0.3f pu',abs(I24f));
printf('\nI23f=-j%0.3f pu\n',abs(I23f));

printf('\n Fault current for a fault on bus 1 (or bus 2)\n If=-j%0.6f pu\n\n',abs(If12));
```

**Scilab code Exa 9.7** Current Injection Method

```
//Chapter 9
//Example 9.7
//page 355
//To evaluate Zbus using Current Injection method
clear;clc;

disp(”We can approach this problem using XCOS simulation”)
disp(”In this simulation”);
disp(”1)For injecting unit current at bus1 keeping bus2 open circuit,we use a current source of 1 unit which is switched on from t=0 to t=2. During this period we can observe the voltage waveforms of V1 and V2 and compare with the results given in the textbook”);
disp(”2)For injecting unit current at bus2 keeping bus1 open circuit,we use a current source of 1 unit which is switched on from t=4 to t=6. During this period we can observe the voltage waveforms of V1 and V2 and compare with the results given in the textbook”);

Z11=7;
Z21=4;
Z12=Z21;
Z22=6;

Zbus=[Z11 Z12;Z21 Z22]
```

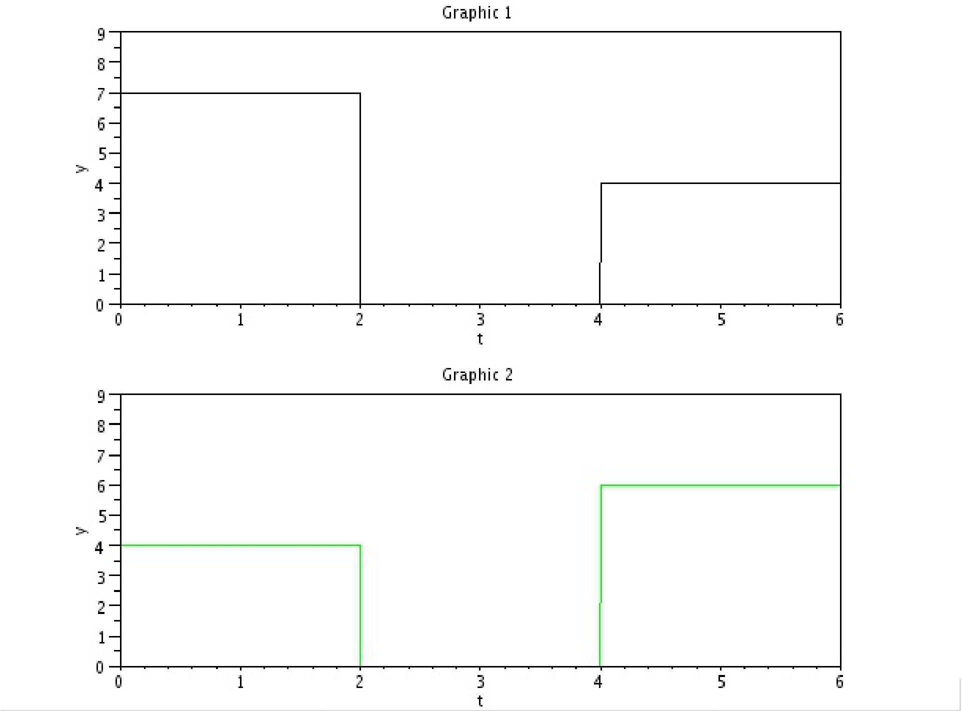


Fig 9.7: Waveform of voltages V1 and V2

Figure 9.6: Current Injection Method

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 9.8** Zbus matrix building using Algorithm

```
//Chapter 9
//Example 9.8
//page 360
```

```
//To build Zbus matrix using Zbus building algorithm
clear;clc;

disp("Let us go on modifying Zbus by including nodes and the elements connected to it one by one as given in the textbook")

//step-1 type1 modification
Zbus=[0.25];
printf('\nstep-1 type1 modification\nZbus=');disp(Zbus);

//step-2 type2 modification
Zbus=[Zbus,0.25;0.25,0.25+0.1];
printf('\nstep-2 type2 modification\nZbus=');disp(Zbus);

//step-3 type2 modification
Zbus=[Zbus [0.25;0.25]; 0.25 0.25 0.35];
printf('\nstep-3 type2 modification\nZbus=');disp(Zbus);

//step-4 type3 modification
Zbus=Zbus-(1/(Zbus(3,3)+0.25))*[Zbus(1:3,2:2)]*[Zbus(2:2,1:3)];
printf('\nstep-4 type3 modification\nZbus=');disp(Zbus);

printf('This is the final Zbus matrix after including all the elements\n');
//step-5 type4 modification
Zbus=Zbus-(1/(0.1+Zbus(2,2)+Zbus(3,3)-2*Zbus(2,3)))*[Zbus(1:3,2:2)-Zbus(1:3,3:3)]*[Zbus(2:2,1:3)-Zbus(3:3,1:3)];
printf('\nstep-5 type4 modification\nZbus=');disp(Zbus);

disp("opening a line between 2-3 is equivalent to
connecting (-0.1) between bus3 bus2")
Zbus=Zbus-(1/(-0.1+Zbus(2,2)+Zbus(3,3)-2*Zbus(2,3)))*[Zbus(1:3,2:2)-Zbus(1:3,3:3)]*[Zbus(2:2,1:3)-Zbus(3:3,1:3)];
printf('Zbus=');disp(Zbus);printf('(same as in step 4)');
```

**Scilab code Exa 9.9** PostFault Currents and Voltages Calculation

```
//Chapter 9
//Example 9.9
//page 362
//To find postfault currents and voltages
clear;clc;

disp("The Thevenin passive network for this system is drawn in Example_9_8 (or fig 9.28 in the textbook)");
disp("Using the Zbus matrix from the results of example_9_8,we can calculate post fault currents and voltages");
Zbus=%i*[0.1397059 0.1102941 0.125;0.1102941 0.1397059 0.125;0.125 0.125 0.175]

//to find fault current
V30=1;V10=1;V20=1;
If=(V30/(Zbus(3,3)+0));
printf('\nIf=-j%0.2f pu\n',abs(If));


//to find postfault voltages
V1f=V10-(Zbus(1,3)/Zbus(3,3));
V2f=V20-(Zbus(2,3)/Zbus(3,3));
```

```
printf('\nV1f=%0.3f',V1f);
printf('\nV2f=%0.3f',V2f);

//to find fault currents in the TL
I12f=(V1f-V2f)/(%i*0.1);
I13f=(V1f-0)/(%i*0.1);
I23f=(V2f-0)/(%i*0.1);
printf('\n\nI12f=%d',I12f);
printf('\nI13f=-j%0.2f',abs(I13f));
printf('\nI23f=-j%0.2f',abs(I23f));

//to find generator currents during faults
Eg1=1;Eg2=1;
Ig1f=(Eg1-V1f)/(0.2*%i+0.05*%i);
Ig2f=(Eg2-V2f)/(0.2*%i+0.05*%i);
printf('\n\nIg1f=-j%0.2f',abs(Ig1f));
printf('\nIg2f=-j%0.2f\n\n',abs(Ig2f));
```

# Chapter 10

# Symmetrical Components

**Scilab code Exa 10.1** Symmetrical components of line currents Calculation

```
//Chapter 10
//Example 10.1
//page 374
//To calculate symmetrical components of line currents
clear;clc;
Ia=10*(cosd(30)+%i*sind(30));
Ib=15*(cosd(-60)+%i*sind(-60));
// from KCL Ia+Ib+Ic=0
Ic=-(Ia+Ib);
//defining alpha(a)
a=cosd(120)+(%i*sind(120));
Ip=[Ia;Ib;Ic];
A=[1 1 1;a^2 a 1;a a^2 1];
IA=inv(A)*Ip;
IB=diag([a^2,a,1])*IA;
IC=diag([a,a^2,1])*IA;

function [r,theta]=phasorform(x)
    r=abs(x);
    theta=atand(imag(x),real(x));
endfunction

[IAr,IAth]=phasorform(IA);
[IBr,IBth]=phasorform(IB);
[ICr,ICth]=phasorform(IC);

//to display the results of symettrical components of line currents

printf('\n\nIA1=%0.2f @ %d deg A',IAr(1,1),IAth(1,1));
printf('\nIA2=%0.2f @ %d deg A',IAr(2,1),IAth(2,1));
printf('\nIA0=%0.2f A',IAr(3,1));


printf('\n\nIB1=%0.2f @ %d deg A',IBr(1,1),IBth(1,1));
printf('\nIB2=%0.2f @ %d deg A',IBr(2,1),IBth(2,1));
printf('\nIB0=%0.2f A',IBr(3,1));


printf('\n\nIC1=%0.2f @ %d deg A',ICr(1,1),ICth(1,1));
printf('\nIC2=%0.2f @ %d deg A',ICr(2,1),ICth(2,1));
printf('\nIC0=%0.2f A',ICr(3,1));

//to calculate Delta currents

IAB=(Ia-Ib)/3;
IBC=(Ib-Ic)/3;
ICA=(Ic-Ia)/3;

//to get the results in phasor notation
[IABr,IABth]=phasorform(IAB);
[IBCr,IBCth]=phasorform(IBC);
[ICAr,ICAth]=phasorform(ICA);

```

```
printf('\n\nIAB=%0.2f @ %d deg A',IABr,IABth);
printf('\nIBC=%0.2f @ %d deg A',IBCr,IBCth);
printf('\nICA=%0.2f @ %d deg A',ICAr,ICAth);

//to calculte the symmetrical components of delta currents by reusing the variable Ip
Ip=[IAB;IBC;ICA];
IAB=inv(A)*Ip;
IBC=diag([a^2,a,1])*IAB;
ICA=diag([a,a^2,1])*IAB;

[IABr,IABth]=phasorform(IAB);
[IBCr,IBCth]=phasorform(IBC);
[ICAr,ICAth]=phasorform(ICA);

//to display the results of symmetrical components of Delta currents

printf('\n\nIAB1=%0.2f @ %d deg A',IABr(1,1),IABth(1,1));
printf('\nIAB2=%0.2f @ %d deg A',IABr(2,1),IABth(2,1));
printf('\nIAB0=%0.2f A',IABr(3,1));


printf('\n\nIBC1=%0.2f @ %d deg A',IBCr(1,1),IBCth(1,1));
printf('\nIBC2=%0.2f @ %d deg A',IBCr(2,1),IBCth(2,1));
printf('\nIBC0=%0.2f A',IBCr(3,1));


printf('\n\nICA1=%0.2f @ %d deg A',ICAr(1,1),ICAth(1,1));
printf('\nICA2=%0.2f @ %d deg A',ICAr(2,1),ICAth(2,1));
printf('\nICA0=%0.2f A\n\n',ICAr(3,1));
```

**Scilab code Exa 10.2** Sequence Network of the System

```
//Chapter 10
//Example 10.2
//page no 390
//To draw sequence networks of the system
clear;clc;

//selecting generator rating as base in generator circuit

mvab=25;
kvGb=11;  //base voltage for generator
kvTLb=kvGb*(121/10.8); //base voltage for TL
kvMb=kvTLb*(10.8/121); //base voltage for motors

xG=%i*0.2;
xT=%i*0.1;
xTL=100;
xM=%i*0.25;

mvaG=25;
mvaT=30;
mvaM1=15;
mvaM2=7.5;

kvM=10;

//converting all the reactances to PUs

xT=xT*(mvab/mvaT)*(10.8/kvGb)^2;
xTL=xTL*(mvab/(kvTLb)^2);
```

```
xM1=xM*(mvab/mvaM1)*(kvM/kvMb)^2;
xM2=xM*(mvab/mvaM2)*(kvM/kvMb)^2;

//displaying the results

printf('\n\nTransmission line voltage base = %0.1f kV',kvTLb);
printf('\n\Motor voltage base = %d kV',kvMb);
printf('\n\nTransformer reactance = %0.4f pu',abs(imag(xT)));
printf('\nLine reactance = %0.3f pu',abs(xTL));
printf('\nReactance of motor 1 = %0.3f pu',abs(imag(xM1)));
printf('\nReactance of motor 2 = %0.3f pu\n\n',abs(imag(xM2)));

disp('Positive and Negative sequence diagram has been drawn using XCOS, simulation has not been done as it is not being asked in the problem');
```

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 10.3** Zero sequence Network

```
//Chapter 10
//Example 10.3
//page no 392
//To draw the zero sequence networks of the system
clear;clc;

disp('Zero sequence diagram has been drawn using XCOS, simulation has not been done as it is not
being asked in the problem');
```

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 10.4** Zero Sequence Network

```
//Chapter 10
//Example 10.4
//page no 392
//To draw the zero sequence networks of the system given in example 10.2
clear;clc;

//selecting generator rating as base in generator circuit

mvab=25;
kvGb=11;  //base voltage for generator
kvTLb=kvGb*(121/10.8); //base voltage for TL
kvMb=kvTLb*(10.8/121); //base voltage for motors

//Calculation of zero sequence reactance

xT0=0.0805; //zero sequence reactance of transformer
xG0=0.06; //zero sequence reactance of generator

//zero sequence reactanc eof motors
xM1_0=0.06*(mvab/15)*(10/kvMb)^2;
xM2_0=0.06*(mvab/7.5)*(10/kvMb)^2;

x_clr_0=3*2.5*(mvab/kvGb^2); // Reactance of current limiting reactors to be icluded in the zero
```

```
sequence network
x_TL_0=300*(mvab/kvTLb^2); //Zero sequence reactance of TL

printf('\n\nTransformer zero sequence reactance = %0.4f pu',xT0);
printf('\nGenerator zero sequence reactances = %0.2f pu',xG0);
printf('\nZero sequence reactance of motor 1 = %0.3f pu',xM1_0);
printf('\nZero sequence reactance of motor 2 = %0.3f pu',xM2_0);
printf('\nReactance of current limiting reactors = %0.3f pu',x_clr_0);
printf('\nZero sequence reactance of transmission line = %0.3f pu\n\n',x_TL_0);

disp('Zero sequence diagram has been drawn using XCOS,simulation has not been done as it is not being asked in the problem');
```

This code can be downloaded from the website wwww.scilab.in

# Chapter 11

# Unsymmetrical Fault Analysis

**Scilab code Exa 11.1** LG and 3Phase faults Comparision

```
//Chapter 11
//Example 11.1
//page 406
//To draw sequence networks of generator and to compare LG fault current will be greater than three-phase fault current when neutral is solidly grounded
clear;clc;

disp("Sequence networks of synchronous generator grounded through neutral impedance has been drawn using XCOS ");

disp("Since the derivation can not be done here, let us do this problem by taking a suitable values for the sequence reactances of the generator");

disp("X1=j0.18, X2=j0.15, X0=j0.10 pu and Ea=1");

disp("From the figs 11.13 and 11.14 in the textbook, we can find Ilg and I3L");
```

```

Ea=1;X1=0.18*%i;X2=0.15*%i;X0=0.10*%i;

IaLG=3*Ea/(2*X1+X0)
Ia3L=3*Ea/(3*X1)

disp("Same values of sequence impedance have been used in XCOS simulation also to varify the result");
```

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 11.2** Grounding Resistor voltage and Fault Current

```
//Chapter 11
//Example 11.2
//page 408
//To find fault current and voltage across the grounding resistor
clear;clc;

X1eq=(%i*0.18)/2;
X2eq=(%i*0.15)/2;
Z0eq=(%i*0.10)+3*(2*20/(11^2));

Ea=1;

//calculation of fault current
printf('\nFault current is given by ');
If=(3*Ea)/(X1eq+X2eq+Z0eq)
```

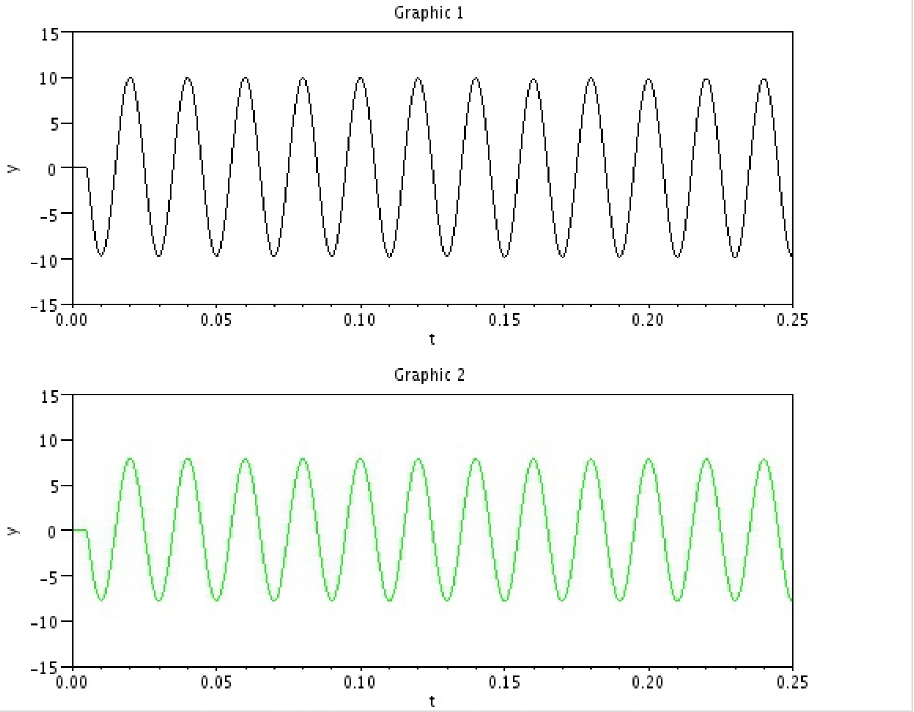


Fig 11.1: Fault current in LG and 3phase Faults

Figure 11.1: LG and 3Phase faults Comparision

```

//current in grounding resistor
Ifg=abs(If)*(20/(11*sqrt(3)));
printf('\n\nCurrent through grounding resistor Ifg=%0.2fkA',Ifg);

//voltage across grounding resistor
Vgr=abs(If*(2*20/(11^2))*(11/sqrt(3)));
printf('\n\nVoltage across grounding resistor Vgr=%0.2fkV\n\n',Vgr);
```

---

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 11.3** Fault and subtransient currents of the system

```
//Chapter 11
//Example 11.3
//page 409
//To find fault current and subtransient current in all parts of the system
clear;clc;

a=-0.5+(sqrt(3)/2)*%i;

//neglecting prefault currents
Vf0=10/11;
Eg=Vf0; Em1=Vf0 ;Em2=Vf0;

//positive sequence network when it is replaced by its thevenin's equvivalent as shown in fig11.18
```

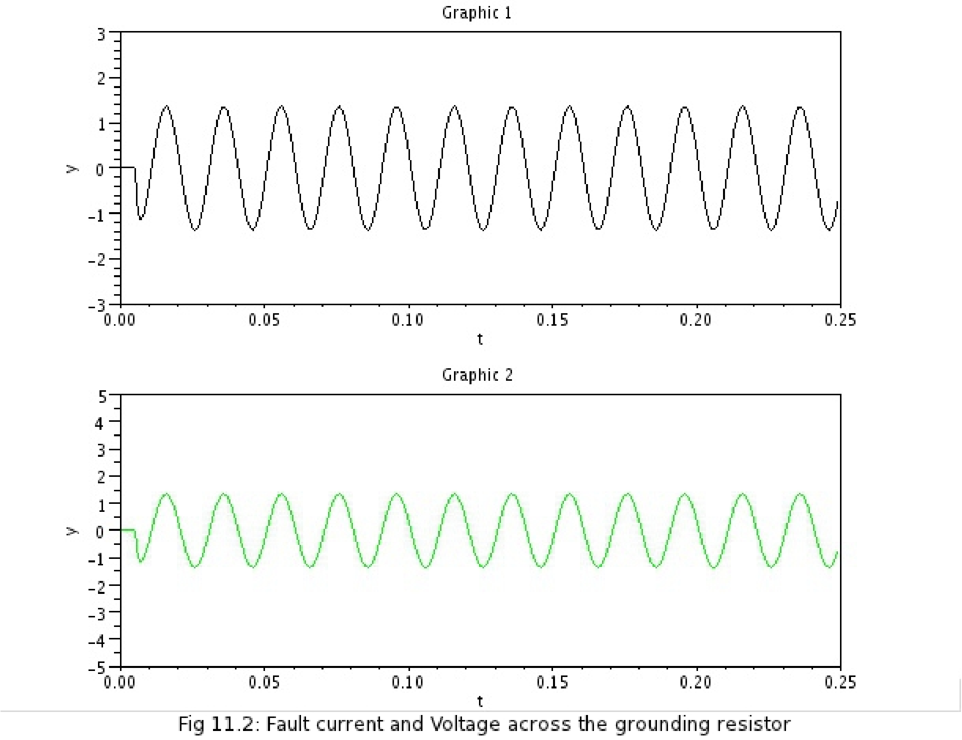


Figure 11.2: Grounding Resistor voltage and Fault Current

```
printf('\nsequence impedances are given by \n');
Z1=(%i*0.525*%i*0.23)/(%i*0.755);
Z2=Z1;
Z0=%i*1.712;
printf('Z1=j%0.4f \nZ2=j%0.4f \nZ0=j%0.4f',abs(imag(Z1)),abs(imag(Z2)),abs(imag(Z0)));
//to find sequence current
Ia1=Vf0/(Z1+Z2+Z0);
Ia2=Ia1;
Ia0=Ia1;

//to find fault current
If=3*Ia0;
printf('\n\nFault Current= -j%0.4f',abs(imag(If)));


//component current flowing from generator and motor
printf('\n\nComponents currents flowing from Generator and motor are \n')
Ig1=Ia1*(0.23/0.755) ;
Ig2=Ig1;
Ig0=0;
printf('Ig1= -j%0.4f \nIg2= -j%0.4f \nIg0=%d',abs(Ig1),abs(Ig2),abs(Ig0));
printf('\n');
Im1=Ia1*(0.525/0.755);
Im2=Im1;
Im0=Ia0;
printf('\nIm1= -j%0.4f \nIm2= -j%0.4f \nIm0= -j%0.4f',abs(Im1),abs(Im2),abs(Im0));

//fault currents from the generator and motor towards g are
printf('\n\nFault current from the generator towards g are ');
Ig=[1 1 1;a^2 a 1;a a^2 1]*[Ig1;Ig2;Ig0];
disp(Ig);
printf('and to g from motors are');
```

```
Im=[1 1 1;a^2 a 1;a a^2 1]*[Im1;Im2;Im0];
disp(Im);

printf('\nPositive sequence current =%0.3f pu',(-%i*Ig1));
printf('\nNegative sequence current =%0.3f pu',(%i*Ig2));
printf('\nZero sequence current=%d\n',Ig0);

//under loaded condition,PU motor currents are
Im1o=(15/(25*0.909*0.8))*(0.800103636+%i*0.5998617938);
Im2o=(7.5/(25*0.909*0.8))*(0.800103636+%i*0.5998617938);
printf('\nThe per unit motor currents are:\n');
printf('Motor1:%0.2f +j%0.3f pu',real(Im1o),imag(Im1o));
printf('\nMotor2:%0.2f +j%0.3f pu',real(Im2o),imag(Im2o));

//the voltages behind subtransient reactances are calculated below
printf('\n\nVoltage behind subtransient reactances:\n');
printf('Motor1:');
Em1=Em1-(%i*0.345*Im1o);
printf('Em1= %0.4f-j%0.4f',real(Em1),abs(imag(Em1)));

printf('\nMotor2:');
Em2=Em2-(%i*0.69*Im2o);
printf('Em2= %0.4f-j%0.4f',real(Em2),abs(imag(Em2)));

printf('\nGenerator:');
Eg=Eg+(%i*0.525*(Im2o+Im1o));
printf('Eg= %0.4fj+%0.4f',real(Eg),abs(imag(Eg)));

```

```
//actual value of positive sequence current from generator and motor
printf('\n\nThe actual value of positive sequence current from the generator towards fault is = %0.2f+j%0.3f',real(Im1o+Im2o+Ig1),imag(Im1o+Im2o+Ig1));
printf('\nThe actual value of positive sequence current from the motors towards fault is = %0.2f-j%0.3f',real(-Im1o-Im2o+Im1),abs(imag(-Im1o-Im2o+Im1)));
```

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 11.4** LL Fault Current

```
//Chapter 11
//Example 11.4
//page 412
//To find L-L fault current and voltage of healthy phase
clc;clear;
X1eq=0.09*%i;
X2eq=0.075*%i;
Z0=0.99+(%i*0.1);
Ea=1;Ia0=0;

//to calculate Ia1
Ia1=Ea/(X1eq+X2eq);

//to calculate fault current
```

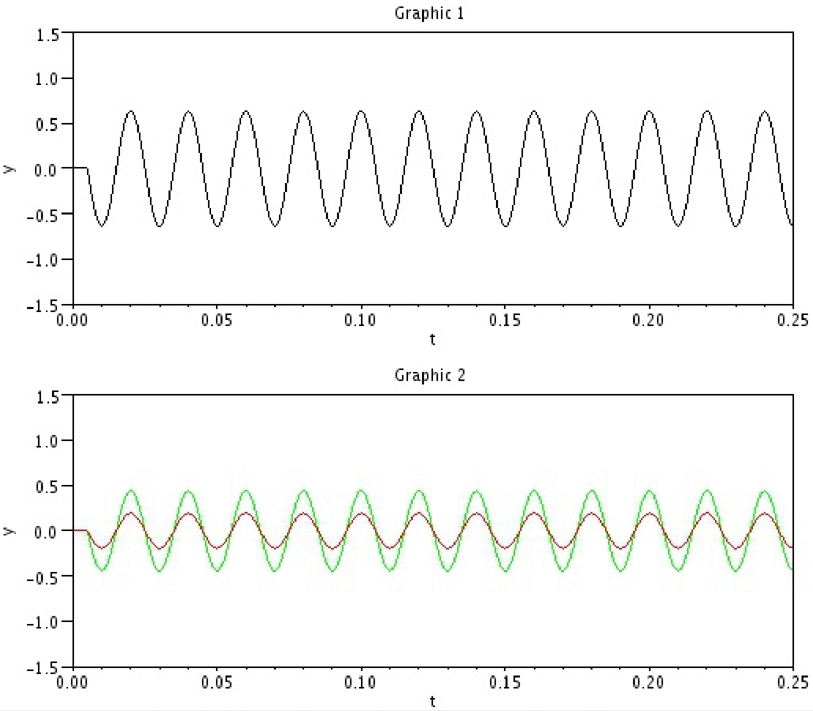


Fig 11.3: Fault Current.Generator and Motor side components

Figure 11.3: Fault and subtransient currents of the system

```
If=(-%i*sqrt(3))*(-%i*6.06);
Va1=Ea-(Ia1*X1eq);
Va0=(-Ia0*Z0);
Va2=Va1;

//voltage in healthy phase
Va=Va1+Va2+Va0;

//displaying the result
printf('\nIa1=-j%0.2f',abs(Ia1));
printf('\nIf=%0.3f',If);
printf('\nVa1=Va2=%0.3f',Va1);
printf('\nVa0=%d',Va0);
printf('\nVa=Va1+Va2+Va0=%0.2f\n\n',Va);
```

This code can be downloaded from the website wwww.scilab.in

**Scilab code Exa 11.5** Double line to ground Fault

```
//Chapter 11
//Example 11.5
//page 413
//To find Double line to ground fault current and voltage of healthy phase
clc;clear;

Z1eq=0.09*%i;
Z2eq=0.075*%i;
Z0=(%i*0.1);
Ea=1;
```

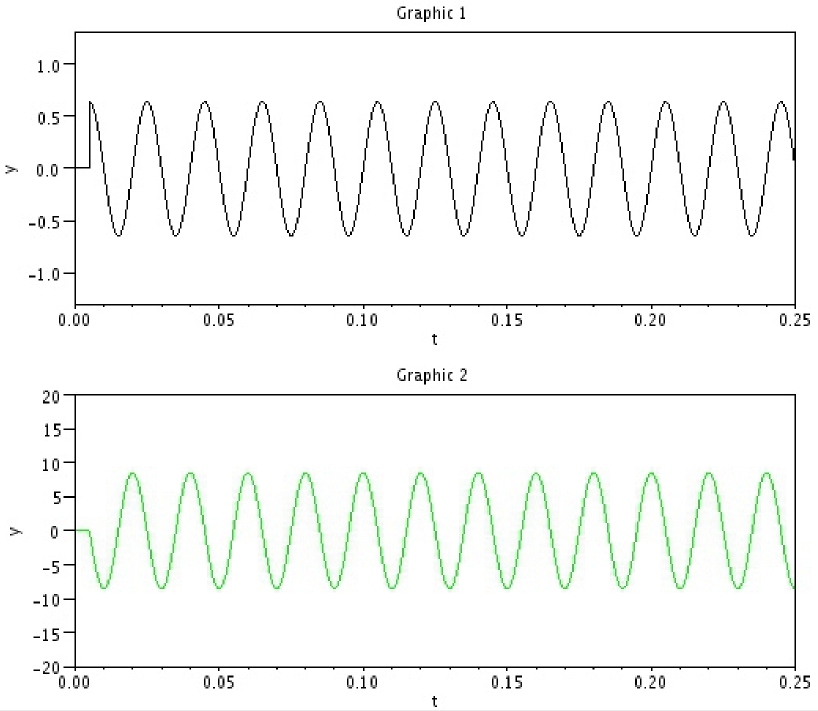


Fig 11.4: Measure of Positive sequence voltage and Fault current

Figure 11.4: LL Fault Current

```
a=(-0.5+%i*sqrt(3)/2);

//to find the sequence components of healthy phase
Ia1=Ea/(Z1eq+(Z2eq*Z0/(Z2eq+Z0)));
Va1=Ea-(Ia1*Z1eq);
Va2=Va1;
Va0=Va1;

Ia2=-(Va2/Z2eq);
Ia0=-(Va0/Z0);

I=[1 1 1;a^2 a 1;a a^2 1]*[Ia1; Ia2; Ia0];

//voltage of the healthy phase
Va=3*Va1;

//displaying the results
printf('Ia1=-j%0.3f\n',abs(Ia1));
printf(' Ia2=j%0.3f\n',abs(Ia2));
printf(' Ia0=j%0.3f\n\n',abs(Ia0));

printf(' Ia=%0.3f + j%0.3f\n',real(I(1,1)),imag(I(1,1)));
printf(' Ib=%0.3f + j%0.3f\n',real(I(2,1)),imag(I(2,1)));
printf(' Ic=%0.3f + j%0.3f\n\n',real(I(3,1)),imag(I(3,1)));

printf(' Voltage of the healthy phase Va=3Va1=%0.3f',Va);
```

This code can be downloaded from the website wwww.scilab.in

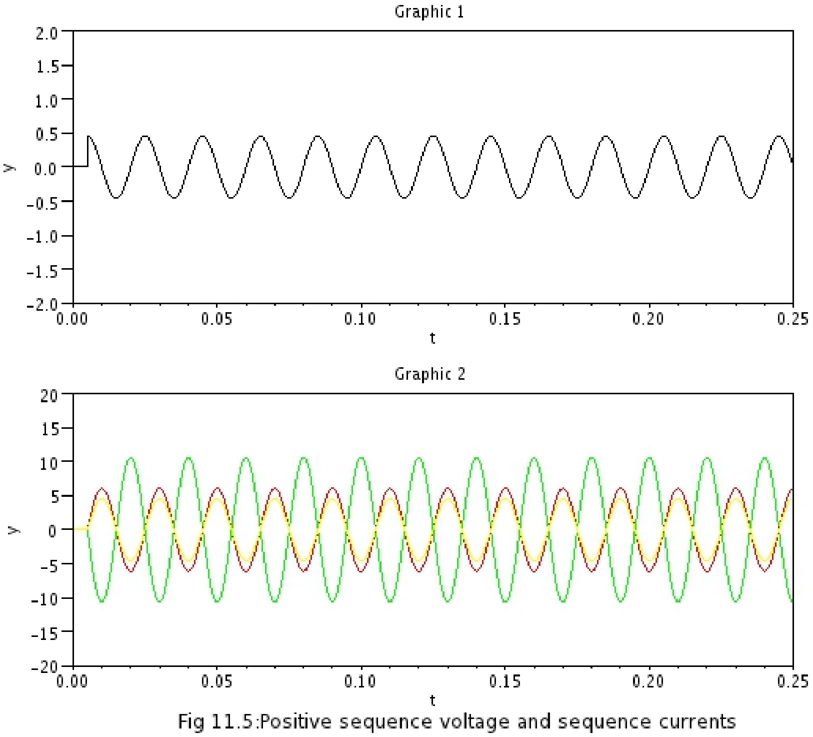


Figure 11.5: Double line to ground Fault

**Scilab code Exa 11.6** Bus Voltages and Currents Calculations

```
//Chapter 11
//Example 11.6
//page 420
//To find bus voltages and currents

clc;clear;
v_pf=1; //prefault voltage
//according to the fig.11.26
Y1dd=((%i*0.2)^-1)+((%i*0.0805)^-1);
Y1fg=-(%i*0.0805)^-1;
Y1de=Y1fg;
Y1ff=((%i*0.0805)^-1)+((%i*0.164)^-1);
Y1ee=Y1ff;
Y1ef=-(%i*0.164)^-1;
Y1gg=((%i*0.0805)^-1)+((%i*0.345)^-1)+((%i*0.69)^-1);
Y1df=0;
Y1dg=0;
Y1ed=Y1de;
Y1eg=0;
Y1fd=0;
Y1fe=Y1ef;
Y1gd=0;
Y1ge=0;
Y1gf=Y1fg;
printf('\nY-Bus and Z-Bus matrix can be written as:\n')
Y1_bus=[Y1dd Y1de Y1df Y1dg;Y1ed Y1ee Y1ef Y1eg;Y1fd Y1fe Y1ff Y1fg;Y1gd Y1ge Y1gf Y1gg];
Y2_bus=Y1_bus;
printf('\nY1_bus=');disp(Y1_bus);
printf('\nY2_bus=');disp(Y2_bus);
```

```
Y0dd=(%i*1.608)^-1;Y0de=0;Y0df=0;Y0dg=0;
Y0ed=0;Y0ee=((%i*0.0805)^-1)+((%i*0.494)^-1);Y0ef=-(%i*0.494)^-1;Y0eg=0;
Y0fd=0;Y0fe=Y0ef;Y0ff=Y0ee;Y0fg=0;
Y0gd=0;Y0de=0;Y0gf=0;Y0gg=(%i*1.712)^-1;

Y0_bus=[Y0dd Y0de Y0df Y0dg;Y0ed Y0ee Y0ef Y0eg;Y0fd Y0fe Y0ff Y0fg;Y0gd Y0de Y0gf Y0gg];
printf('\nY0_bus=');disp(Y0_bus);

//finding Z-bus matrix
Z1_bus=inv(Y1_bus);
Z2_bus=inv(Y2_bus);
Z0_bus=inv(Y0_bus);
printf('\n\nZ1bus=');disp(Z1_bus);
printf('\nZ2_bus=');disp(Z2_bus);
printf('\nZ0_bus=');disp(Z0_bus);

//to find fault current with LG fault on bus e --- case(i)
If_e=(3*v_pf)/(Z1_bus(2,2)+Z2_bus(2,2)+Z0_bus(2,2));
printf('\n\n\nFault current with LG fault on bus e is If_e= -j%0.5f\n',abs(imag(If_e)));

//to find fault current with LG fault on bus f --- case(ii)
If_f=(3*v_pf)/(Z1_bus(3,3)+Z2_bus(3,3)+Z0_bus(3,3));
printf('Fault current with LG fault on bus f is If_f = -j%0.5f\n',abs(imag(If_f)));

//to find bus voltages and line currents in case(i)
printf('\n\n\nBus voltages and currents are given below:\n\n');
Vf1_d=1-(Z1_bus(1,2)*If_e/3);
Vf1_e=1-(Z1_bus(2,2)*If_e/3);
Vf1_f=1-(Z1_bus(3,2)*If_e/3);
Vf1_g=1-(Z1_bus(4,2)*If_e/3);
disp('Vf1_d=');disp(Vf1_d);
disp('Vf1_e=');disp(Vf1_e);
disp('Vf1_f=');disp(Vf1_f);
disp('Vf1_g=');disp(Vf1_g);

printf('\n\n\n');
Vf2_d=-(Z2_bus(1,2)*If_e/3);
Vf2_e=-(Z2_bus(2,2)*If_e/3);
Vf2_f=-(Z2_bus(3,2)*If_e/3);
Vf2_g=-(Z2_bus(4,2)*If_e/3);
disp('Vf2_d=');disp(Vf2_d);
disp('Vf2_e=');disp(Vf2_e);
disp('Vf2_f=');disp(Vf2_f);
disp('Vf2_g=');disp(Vf2_g);

printf('\n\n\n');
Vf0_d=-(Z0_bus(1,2)*If_e/3);
Vf0_e=-(Z0_bus(2,2)*If_e/3);
Vf0_f=-(Z0_bus(3,2)*If_e/3);
Vf0_g=-(Z0_bus(4,2)*If_e/3);
disp('Vf0_d=');disp(Vf0_d);
disp('Vf0_e=');disp(Vf0_e);
disp('Vf0_f=');disp(Vf0_f);
disp('Vf0_g=');disp(Vf0_g);

printf('\n\n\n');
If1_fe=-Y1fe*(Vf1_f-Vf1_e);disp('If1_fe=');disp(If1_fe);
If1_de=-Y1de*(Vf1_d-Vf1_e);disp('If1_de=');disp(If1_de);
Ia1=If1_fe+If1_de;disp('Ia1=');disp(Ia1);

printf('\n\n\n');
If1_gf=-Y1gf*(Vf2_g-Vf2_f);disp('If1_gf=');disp(If1_gf);

printf('\n\n\n');
If2_fe=-Y1fe*(Vf2_f-Vf2_e);disp('If2_fe=');disp(If2_fe); //Y2fe=Y1fe
```

```
If0_fe=-Y0fe*(Vf2_f-Vf2_e);disp('If0_fe=');disp(If0_fe);
If_fe=If1_fe+If2_fe+If0_fe;disp('If_fe=');disp(If_fe);
```

**Scilab code Exa 11.7** Short Circuit Current Calculations

```
//Chapter 11
//Example 11.7
//page 423
//To find short circuit currents

clc;clear;
v_pf=1; //prefault voltage
a=0.5+0.8660254*%i;
//according to the fig.11.28 we can write Z-bus matrix for positive and negative phase sequence
printf('\nstep by step for finding Z1_bus\n')

//Bus1 to referance bus
Z1_bus=[0.15];
printf('Bus1 to reference\nZ1_bus=');disp(Z1_bus);

//Bus2 to Bus1
Z1_bus=[Z1_bus 0.15;0.15 0.15+0.2];
printf('\nBus2 to Bus1\nZ1_bus=');disp(Z1_bus);

//Bus2 to reference bus
Z1_bus=Z1_bus-(1/(Z1_bus(2,2)+0.15))*[Z1_bus(1:2,2:2)]*[Z1_bus(2:2,1:2)];
Z1_bus=(%i*Z1_bus);
Z2_bus=Z1_bus;
printf('\nBus2 to Reference\nZ1_bus=');disp(Z1_bus);
printf('\nZ2_bus=');disp(Z2_bus);

//according to the fig.11.29 we can write Z-bus matrix for zero phase sequence
printf('\nstep by step for finding Z0_bus\n')
//Bus1 to referance bus
Z0_bus=[0.05];
printf('\nBus1 to reference \nZ0_bus=');disp(Z0_bus);

//Bus2 to Bus1
Z0_bus=[Z0_bus 0.05;0.05 0.05+0.4];
printf('\nBus1 to Bus1 \nZ0_bus=');disp(Z0_bus);

//Bus2 to reference bus
Z0_bus=Z0_bus-(1/(Z0_bus(2,2)+0.05))*[Z0_bus(1:2,2:2)]*[Z0_bus(2:2,1:2)];
Z0_bus=(%i*Z0_bus);
printf('\nBus2 to reference \nZ0_bus=');disp(Z0_bus);

//to find positive sequence of fault current
printf('\n\n\nFault current calculation\n')
If1_1=v_pf/(Z1_bus(1,1)+Z2_bus(1,1)+Z0_bus(1,1));printf('If1_1 = -j%0.5f',abs(imag(If1_1)));
printf('\nFault current=If1=3If1_1=-j%0.1f\n\n',abs(imag(3*If1_1)));

Vf1_1=1-Z1_bus(1,1)*If1_1;
Vf1_2=1-Z1_bus(2,1)*If1_1;

Vf2_1=-Z2_bus(1,1)*If1_1;
Vf2_2=-Z2_bus(2,1)*If1_1;

Vf0_1=-Z0_bus(1,1)*If1_1;
Vf0_2=-Z0_bus(2,1)*If1_1;

If1_12=((%i*0.2)^-1)*(Vf1_1-Vf1_2);
```

```
If2_12=((%i*0.2)^-1)*(Vf2_1-Vf2_2);
If0_12=((%i*0.4)^-1)*(Vf0_1-Vf0_2);

If=[1 1 1;a^2 a 1;a a^2 1]*[If1_12;If2_12;If0_12];

printf('\n\n\nShort circuit current on the transmission line in all the three phases\n')
printf('\nIf_a_12=');
disp(If(1,1));

printf('\nIf_b_12=');
disp(If(2,1));

printf('\nIf_b_12=');
disp(If(3,1));

//short circuit current phase(a) of the generator
If1_G=((0.15*%i)^-1)*(1-Vf1_1)*(cosd(-30)+%i*sind(-30));
If2_G=((0.15*%i)^-1)*(0-Vf2_1)*(cosd(30)+%i*sind(30));
If0_G=0;
printf('\n\n\nshort circuit current phase(a) of the generator\n')
Ifa_G=If1_G+If2_G+If0_G; printf('Ifa_G = -j%0.5f', abs(imag(Ifa_G)));

//Voltage of the healthy phases of the bus 1.
printf('\n\n\nVoltage of the healthy phases of the bus 1\n')
Vf_b_1=Vf1_1*(cosd(240)+%i*sind(240))+Vf2_1*(cosd(120)+%i*sind(120))+Vf0_1; printf('Vf_b_1=%0.4f - j%0.5f',real(Vf_b_1),abs(imag(Vf_b_1)));
Vf_c_1=Vf1_1*(cosd(120)+%i*sind(120))+Vf2_1*(cosd(240)+%i*sind(240))+Vf0_1; printf('\nVf_c_1=%0.4f + j%0.5f',real(Vf_c_1),abs(imag(Vf_c_1)));
```

# Chapter 12

# Power System Stability

**Scilab code Exa 12.1** Calculation of stored kinetic energy and rotor acceleration

```
//Chapter 12
//Example 12.1
//page 439
//To find stored kinetic energy,rotor acceleration, change in torque angle and rotor speed
clear;clc;
G=100; //base machine rating
H=8.0; //inertia constant
P=4; //no of poles
//(a)To find stored energy in rotor at synchronous speed
stored_energy=G*H;
printf('\nStored energy = %d MJ',stored_energy);

//(b)To find rotor acceleration when mechanical input is raised 80MW for an electrical load of 50 MW
Pa=30; //nett power
f=50; //frequency
M=stored_energy/(180*f);
```

```
alpha=Pa/M; //rotor acceleration
printf('\n\nRotor acceleration = %0.1f elect deg/s^2',alpha);

//(c)To calculate change in torque angle and rotor speed when the above acceleration is maintained for 10 cycles
change_angle=0.5*alpha*(10*20*10^(-3));
printf('\n\nChange in torque angle = %0.2f elect degrees',change_angle);
change_angle=60*alpha/(2*360);
printf('\nChange in torque angle = %0.3f rpm/s',change_angle);
speed=(120*f/P)+(change_angle*0.2);
printf('\n\nRoor speed at the end of 10 cycles = %0.3f rpm',speed);
```

**Scilab code Exa 12.2** steady state power limit

```
//Chapter 12
//Example 12.1
//page 448
//To calculate steady state power limit
clear;clc;

Xdg=1*%i;  //generator's
Xdm=1*%i;  //motor's
Xt=0.1*%i; //transformers
Xl=0.25*%i; //transmission line's
Xc=-1*%i; //static capacitor's
Xi=1*%i; //inductive reactor
Eg=1.2; //generator's internal voltage
Em=1; //motor's internal voltage
```

```

//case(i) steady state power limit without reactor
P1=(abs(Eg)*abs(Em))/(abs(Xdg+Xt+Xl+Xt+Xdm));
printf('\n\n Steady state power limit without reactor = %0.5f pu',P1);

//case(ii) steady state power limit with capacitive reactor
//three arms of star connected reactances are
Xa=Xdg+Xt+Xl; //from generator side
Xb=Xdm+Xt;   //from load side
Xc=Xc; //from reactor side

//converting star to delta
//reactance between generator side to load side is
Xab=(Xa*Xb+Xb*Xc+Xc*Xa)/Xc;
//power limit is
P2=(abs(Eg)*abs(Em))/(abs(Xab));
printf('\n\n Steady state power limit with capacitive reactor = %0.5f pu',P2);

//case(iii) steady state power limit with inductive reactor
//three arms of star connected reactances are
Xa=Xdg+Xt+Xl; //from generator side
Xb=Xdm+Xt;   //from load side
Xc=Xi; //from reactor side

//converting star to delta
//reactance between generator side to load side is
Xab=(Xa*Xb+Xb*Xc+Xc*Xa)/Xc;
//power limit is
P3=(abs(Eg)*abs(Em))/(abs(Xab));
printf('\n\n Steady state power limit with inductive reactor = %0.5f pu',P3);
```

**Scilab code Exa 12.3** Maximum Power Transferred

```
//Chapter 12
//Example 12.3
//page 450
//To calculate maximum power transferred
clear;clc;

Vt=1.0; //generator terminal voltage
V=1.0 ; //infinite bus voltage
Pe=1.0 ; //power delivered
Xd=0.25*%i ; //generator's transient reactance
Xl=0.5*%i ; //transmission line's reactance
Xt=0.1*%i; //transformer's reactance

//to calculate alpha
alpha=asind(Pe*abs(Xt+Xl/2)/(abs(Vt)*abs(V)));
printf('\n\nAlpha=%0.1f deg',alpha);

//current to infinite bus
I=(Vt*(cosd(alpha)+%i*sind(alpha))-V)/(Xt+Xl/2);
printf('\nCurrent to infinte bus=%d+j%0.3f pu',real(I),imag(I));

//votage behind transient reactance
E=Vt+I*(Xd+Xt+Xl/2);
printf('\nVoltage behind transient reactance= E''=%0.3f+j%0.1f pu = %0.3f @%0.1f deg pu\n\n',real(E),imag(E),abs(E),atand(imag(E)/real(E)));

delta=0:0.001:180;

```

```
//case(a) Maximum power when system is healthy
X12=Xd+Xt+Xl/2;
Pmax=abs(V)*abs(E)/abs(X12);
Pe1=Pmax*sind(delta);
printf('Maximum power that can be transferred under the following condition is')
printf('\n\n(a)System Healthy:');
printf('\nPmax=%0.2f pu',Pmax);
printf('\nPe=%0.2f sin(delta) pu',Pmax);

//case(b) One line short in the middle
//converting bus3 to delta40
Xa=Xd+Xt; //generator side
Xb=Xl; //healthy transmission line side
Xc=Xl/2; //unhealthy line side
X12=(Xa*Xb+Xb*Xc+Xc*Xa)/(Xc);
Pmax=abs(V)*abs(E)/abs(X12);
Pe2=Pmax*sind(delta);
printf('\n\n(b)One line shorted in the middle:');
printf('\nPmax=%0.4f pu',Pmax);
printf('\nPe=%0.4f sin(delta) pu',Pmax);

//case(c) One line open
X12=Xd+Xt+Xl;
Pmax=abs(V)*abs(E)/abs(X12);
Pe3=Pmax*sind(delta);
printf('\n\n(c)One line open :');
printf('\nPmax=%0.4f pu',Pmax);
printf('\nPe=%0.4f sin(delta) pu',Pmax);

//plotting Power angle curves
plot(delta,Pe1,delta,Pe2,delta,Pe3);
legend(['1.79sin(delta)';'0.694sin(delta)';'1.265sin(delta)']);
title("Power angle curves");
xlabel("Delta");
ylabel("Pe");
```

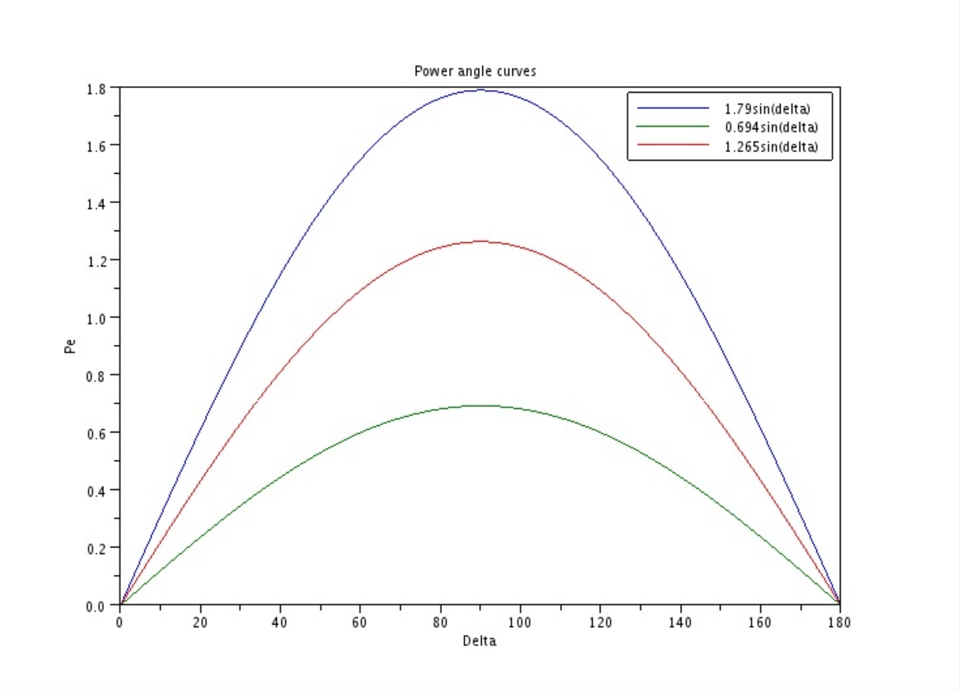


Figure 12.1: Maximum Power Transferred

**Scilab code Exa 12.4** Acceleration and Rotor angle

```
//Chapter 12
//Example 12.4
//page 453
//To calculate acceleration and rotor angle
clear;clc;

delta0=33.9; //initial rotor angle
```

```
H=4; //inertia constant
f=50; //frequency
Pm=1; //mechanical power input
t=0.05; //time interval
angular_acceleration=(Pm-0.694*sind(delta0))*180*f/H;
delta_change=0.5*angular_acceleration*t^2;
delta_new=delta0+delta_change;
new_angular_acceleration=(Pm-0.694*sind(delta_new))*180*f/H;

printf('\n\nInitial rotor angular acceleration = %d elect deg/s^2',angular_acceleration);
printf('\nDelta_change=%0.1f deg',delta_change);
printf('\nNew delta =delta1=%0.1f deg',delta_new);
printf('\nAngular acceleration at the end of 0.05s = %d elect deg/s^2\n\n',new_angular_acceleration);
```

**Scilab code Exa 12.5** Frequency Of Natural Oscilations

```
//Chapter 12
//Example 12.5
//page 456
//To calculate frequency of natural oscilations
clear;clc;

E=1.2; //no load voltage
V=1; //infinite bus voltage
Xg=1.2; // synchronous generator reactance
Xtl=0.6 //transformer anf transmission line reactance
H=4; //inertia constant

```

```
//case(i) 50% loading
delta0=asind(0.5);
synchronizing_coefficien=(abs(E)*abs(V)*cosd(delta0))/(Xg+Xtl);
M=H/(%pi*50);
p=%i*sqrt(synchronizing_coefficien/M);
f=abs(p)/(2*%pi);
printf('\n\ncase(i) For 50%% loading');
printf('\nDelta_0=%d deg',delta0);
printf('\nsynchronizing_coefficient=%0.3f MW(pu)/elect rad',synchronizing_coefficien);
printf('\nM=%0.4f s^2/elect rad',M);
printf('\nFrequency of oscillations=%0.2f rad/sec = %0.3f Hz\n',abs(p),f);

//case(i) 80% loading
delta0=asind(0.8);
synchronizing_coefficien=(abs(E)*abs(V)*cosd(delta0))/(Xg+Xtl);
M=H/(%pi*50);
p=%i*sqrt(synchronizing_coefficien/M);
f=abs(p)/(2*%pi);
printf('\n\ncase(ii) For 80%% loading');
printf('\nDelta_0=%d deg',delta0);
printf('\nSynchronizing_coefficient=%0.3f MW(pu)/elect rad',synchronizing_coefficien);
printf('\nM=%0.4f s^2/elect rad',M);
printf('\nFrequency of oscillations=%0.2f rad/sec = %0.3f Hz\n',abs(p),f);
```

**Scilab code Exa 12.6** Steady State Power Limit 2

```
//Chapter 12
```

```
//Example 12.6
//page 457
//To find steady state power limit
clear;clc;

V=1.0; //infinite bus volatge
Vt=1.2; //terminal volatge
Xd=0.5*%i; //synchronous generator reactance
X=%i; //series reactance
//by solving the expressions given in the textbook
theta=acosd(0.5/1.8);
printf('\n\ntheta=%0.3f deg',theta);
Vt=Vt*(cosd(theta)+%i*sind(theta));
printf('\nVt=%0.3f+j%0.3f pu',real(Vt),imag(Vt));
I=(Vt-V)/X;
printf('\nI=%0.3f+j%0.3f pu',real(I),imag(I));
E=Vt+Xd*I;
printf('\nE=%0.3f @ %d deg pu',abs(E),atand(imag(E)/real(E)));
Pmax=(abs(E)*abs(V))/abs(X+Xd);
printf('\n\nSteady state power limit is given by:\tPmax=%0.3f pu',Pmax);
E=1.2;Pmax=(abs(E)*abs(V))/abs(X+Xd);
printf('\n\nIf the generator emf is held fixed at a value 1.2pu,steady state power limit would be :\tPmax=%0.2f pu\n\n',Pmax);
```

**Scilab code Exa 12.7** Critcal Clearing Angle

```
//Chapter 12
//Example 12.7
//page 475
//To calculate critcal clearing angle
```

```
clear;clc;

Xd=0.25; //direct axis transient reactance of the generator
Xl1=0.5; Xl2=0.4; //reactances of transmission line
E=1.2; //voltage behind transient reactance
Xinf=0.05; //reactnce before infinite bus
V=1; //infinite bus voltage
Pm=1; //mechanical input to the generator
delta=0:1:180;

//Normal operation (prefault)
X1=Xd+(Xl1*Xl2/(Xl1+Xl2))+Xinf; //equivalent reactance between sending ened and receiving end
//Power angle equation before the fault is
Pe1=(E*V/X1)*sind(delta);
//prefault operating power =1.0pu
delta0=asin(1/max(Pe1));
printf('Normal Operation (prefault):\n');
printf('X1=%0.3f PU\n',X1);
printf('Pe1=%0.1fsin(delta)\n\n',max(Pe1));

//during fault there will be no power transfer
Pe2=0;
printf('During Fault:\n');
printf('Pe2=%d\n\n',Pe2);

//Post fault operation(fault cleared by opening the faulted line)
X3=Xd+Xl1+Xinf;
Pe3=(E*V/X3)*sind(delta);
delta_max=%pi-asin(Pm/max(Pe3));
//from A1 and A2, we solve A1=A2
deff('[y]=fx(delta_cr)',"y=1.5*cos(delta_cr)+delta_cr-1.293-Pm*(delta_cr-delta0)");
delta_cr=fsolve(0.45,fx);
printf('Post fault operation(fault cleared by opening the faulted line):\n');
printf('X3=%0.1fPU\n',X3);
printf('Pe3=%0.1fsin(delta)\n',max(Pe3));
printf('Delta_cr=%0.4f rad =%0.2f deg',delta_cr,delta_cr*180/%pi);
plot(delta,Pe1,delta,Pe3,delta,Pm);
legend('Pe1=2.3sin(delta)','Pe3=1.5sin(delta)','Pm=1');
title('Power angle Diagram for example 12.7');
xlabel('delta (in degrees)------>');
ylabel('Electrical output (Pe)------>');
```

**Scilab code Exa 12.8** Critcal Clearing Angle 2

```
//Chapter 12
//Example 12.8
//page 477
//To calculate critcal clearing angle
clear;clc;

Xd=0.25; //direct axis transient reactance of the generator
Xl1=0.28; Xl2_1=0.14;Xl2_2=0.14; //reactances of transmission line
E=1.2; //voltage behind transient reactance
Xinf=0.17; //reactnce before infinite bus
V=1; //infinite bus voltage
Pm=1; //mechanical input to the generator
Xtr=0.15; //transformer reactance
delta=0:1:180;

//prefault operation
```

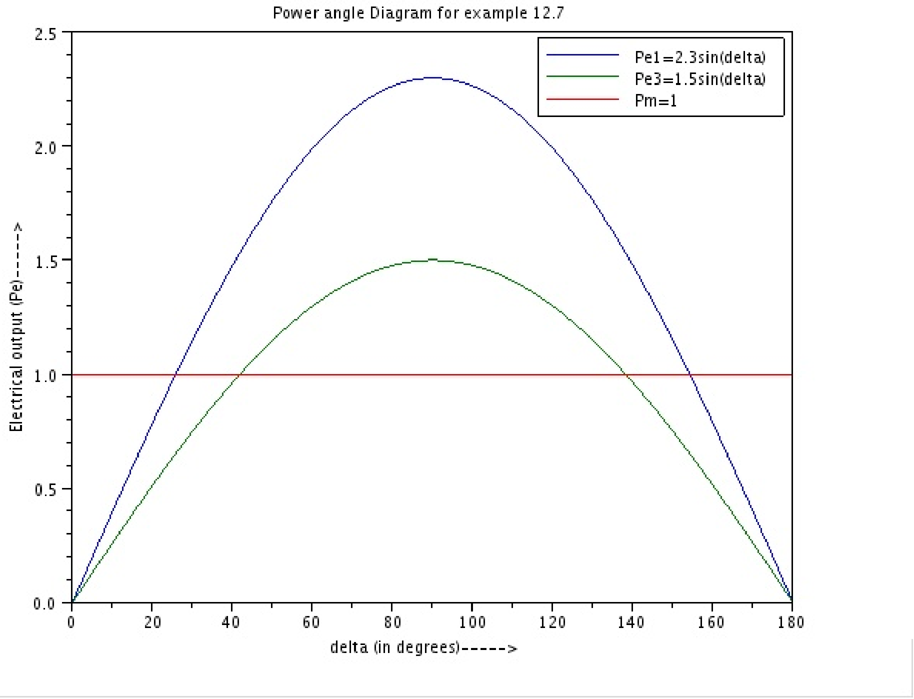


Figure 12.2: Critcal Clearing Angle

```
X1=Xd+Xinf+(Xtr+Xl1+Xtr)/2; //transfer reactance between generator and infinte bus
Pe1=E*V*sind(delta)/X1;
delta0=asin(1/max(Pe1));
printf('Normal Operation (prefault):\n');
printf('X1=%0.3f PU\n',X1);
printf('Pe1=%0.2fsin(delta)\n',max(Pe1));
printf('delta0=%0.3fPU\n\n',delta0);
//during fault there will be no power transfer
//using star delta transformation given in the textbook
X2=2.424;
Pe2=E*V*sind(delta)/X2;
printf('During Fault:\n');
printf('X2=%0.3f PU\n',X2);
printf('Pe2=%0.3fsin(delta)\n\n',max(Pe2));

//Post fault operation(faulty line switched off)
X3=Xd+Xinf+(Xtr+Xl1+Xtr);
Pe3=E*V*sind(delta)/X3;
delta_max=%pi-asin(Pm/max(Pe3));
//from A1 and A2, we solve A1=A2
deff('[y]=fx(delta_cr)',"y=-delta0+max(Pe2)*cos(delta_cr)-0.399-0.661-max(Pe3)*cos(delta_cr)+delta_max");
delta_cr=fsolve(0.45,fx);
printf('Post fault operation(faulty line switched off):\n');
printf('X3=%0.1fPU\n',X3);
printf('Pe3=%0.1fsin(delta)\n',max(Pe3));
printf('Delta_cr=%0.4f rad =%0.2f deg',delta_cr,delta_cr*180/%pi);
plot(delta,Pe1,delta,Pe2,delta,Pe3,delta,Pm);
legend('Pe1=1.69sin(delta)','Pe2=0.495sin(delta)','Pe3=1.2sin(delta)','Pm=1');
title('Power angle Diagram for example 12.8');
xlabel('delta (in degrees)------>');
ylabel('Electrical output (Pe)------>');
```

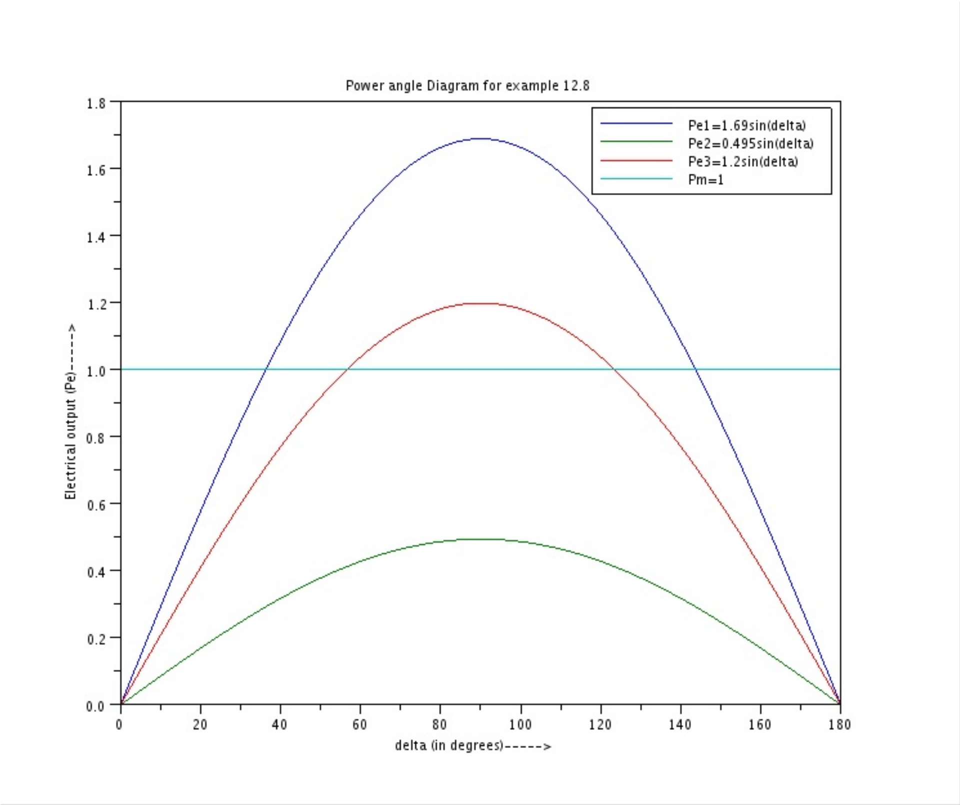


Figure 12.3: Critcal Clearing Angle 2

```
f=get("current_figure")
f.figure_position=[0,15]
f.figure_size=[750,750]
```

**Scilab code Exa 12.9** Critcal Clearing Angle 3

```
//Chapter 12
```

```
//Example 12.9
//page 479
//To calculate critcal clearing angle
clear;clc;
Pmax1=2; // prefault(2 lines)
Pmax2=0.5; //deuring fault
Pmax3=1.5; //post fault(1 line)
Pm=1; //initial loading

delta0=asin(Pm/Pmax1);
delta_max=%pi-asin(Pm/Pmax3);

//to find critical angle,using eq.12.67
delta_cr=acos((Pm*(delta_max-delta0)-Pmax2*cos(delta0)+Pmax3*cos(delta_max))/(Pmax3-Pmax2));
printf('Pmax1=%0.1f PU\t Pmax2=%0.2f PU\t Pmax3=%0.2f PU\n\n',Pmax1,Pmax2,Pmax3);
printf('Delta0=%0.3f rad\n\n',delta0);
printf('Delta_max=%0.3f rad\n\n',delta_max);
printf('Delta_cr=%0.3f rad =%0.2f deg\n\n',delta_cr,delta_cr*180/%pi);
```

**Scilab code Exa 12.10** Swing Curves For Sustained Fault and Cleared Fault at the Specified Time

```
//Chapter 12
//Example 12.10
//page 482
//To plot swing curves for sustained fault and fault cleared at 2.5 and 6.25 cycles
clear;clc;
P_delivered=18;
MVA_base=20;
```

```
Xd=0.35;E=1.1;
Xl=0.2;
V=1;

H=2.52;
f=50;
M=H/(180*f);

/////////Prefault//////////////////////////
X1=Xd+Xl/2;
delta=0:0.1:180;
Pe1=E*V*sind(delta)/X1;
P_initial=P_delivered/MVA_base;Pm=P_initial;
delta0=asind(P_initial/max(Pe1));


//////during fault////////
X2=1.25;  //from delta to star conversion
Pe2=E*V*sind(delta)/X2;

//////postfault:with faulted line switched off
    /////////
X3=Xd+Xl;
Pe3=E*V*sind(delta)/X3;

Pa_0minus=0;
Pa_0plus=Pm-max(Pe2)*sind(delta0);
Pa_avg=(Pa_0minus+Pa_0plus)/2;


/////for a sustained fault///////////
P_max=max(Pe2);
delta_delta=0; //initially
delta=21.64; //initially
delta_old=21.64;
delta_t=0.05;
z1=21.64
n=10;
```

```
T=0;
printf('Point-by-point calculation of swing curve for sustained fault delta_t=0.05sec\n');
printf('------------------------------------------------------------\n');
printf('t\t\tPmax\t\t sin(delta)\t\tPa\t\t y\t\tdelta\n');
printf('------------------------------------------------------------\n');
printf('%0.3f sec\t%0.3f\t\t %0.3f\t\t\t%0.3f\t\t%0.3f\t\t%0.3f\n',0.000,P_max,sind(delta),(0.9-P_max*sind(delta))/2,8.929*(0.9-P_max*sind(delta))/2,delta);
for i=1:n
    t=i*delta_t;
    if i==1 then
        Pa=(0.9-P_max*sind(delta_old))/2;
    else
        Pa=0.9-P_max*sind(delta_old);
    end

    y=(delta_t^2)*Pa/M;
    delta_delta=delta_delta+y;
    delta=delta+delta_delta;
    z1=[z1,delta];T=[T,t];
    printf('%0.3f sec\t%0.3f\t\t %0.3f\t\t\t%0.3f\t\t%0.3f\t\t%0.3f\n',t,P_max,sind(delta),0.9-P_max*sind(delta),8.929*(0.9-P_max*sind(delta)),delta);
    delta_old=delta;
end


//////Fault cleared in 2.5 cycles(time to clear fault=0.05sec)/////

```

```
P_max1=max(Pe2);
P_max2=max(Pe3);
delta_delta=0; //initially
delta=21.64; //initially
delta_old=21.64;
delta_t=0.05;
z2=21.64
n=10;
T=0;
printf('\n\nComputations of swing curves for fault cleared at 2.5 cycles(0.05sec)\n');
printf('------------------------------------------------------------------\n');
printf('t\t\tPmax\t\t sin(delta)\t\tPa\t\t y\t\tdelta\n');
printf('------------------------------------------------------------------\n');
printf('%0.3f sec\t%0.3f\t\t %0.3f\t\t\t%0.3f\t\t%0.3f\t\t%0.3f\n',0.000,P_max,sind(delta),(0.9-P_max*sind(delta))/2,8.929*(0.9-P_max*sind(delta))/2,delta);
for i=1:n
    t=i*delta_t;
    if i==1 then
        Pa=(0.9-P_max*sind(delta_old))/2;
        P_max=P_max1;
    elseif i==2 then
        Pa=((0.9-P_max2*sind(delta_old))+((0.9-P_max1*sind(delta_old))))/2;
        P_max=P_max2;
    else
        Pa=0.9-P_max2*sind(delta_old);
        P_max=P_max2;
    end

    y=(delta_t^2)*Pa/M;
    delta_delta=delta_delta+y;
    delta=delta+delta_delta;
    z2=[z2,delta];T=[T,t];

    if i==1 then
        delta_old=delta;
      printf('%0.3f sec\t%0.3f\t\t %0.3f\t\t\t%0.3f\t\t%0.3f\t\t%0.3f\n',t,P_max,sind(delta),((0.9-P_max2*sind(delta_old))+((0.9-P_max1*sind(delta_old))))/2,8.929*((0.9-P_max2*sind(delta_old))+((0.9-P_max1*sind(delta_old))))/2,delta);
    else
        printf('%0.3f sec\t%0.3f\t\t %0.3f\t\t\t%0.3f\t\t%0.3f\t\t%0.3f\n',t,P_max,sind(delta),0.9-P_max*sind(delta),8.929*(0.9-P_max*sind(delta)),delta);
        delta_old=delta;
    end

end



//////Fault cleared in 6.25 cycles(time to clear fault=0.125sec)/////

P_max1=max(Pe2);
P_max2=max(Pe3);
P_max=P_max1;
delta_delta=0; //initially
delta=21.64; //initially
delta_old=21.64;
delta_t=0.05;
z3=21.64
n=10;
T=0;
printf('\n\nComputations of swing curves for fault
   cleared at 6.25 cycles(0.125sec)\n');
printf('
   ------------------------------------------------------------------
   \n');
printf('t\t\tPmax\t\t sin(delta)\t\tPa\t\t y\t\
   tdelta\n');
printf('
   ------------------------------------------------------------------
   \n');
printf('%0.3f sec\t%0.3f\t\t %0.3f\t\t\t%0.3f\t\t%0
   .3f\t\t%0.3f\n',0.000,P_max,sind(delta),(0.9-
   P_max*sind(delta))/2,8.929*(0.9-P_max*sind(delta)
   )/2,delta);
for i=1:n
    t=i*delta_t;
    if i==1 then
        Pa=(0.9-P_max1*sind(delta_old))/2;
        P_max=P_max1;
    elseif i==2 then
        Pa=(0.9-P_max1*sind(delta_old));
        P_max=P_max1;
    elseif i==3 then
        Pa=(0.9-P_max1*sind(delta_old));
        P_max=P_max2;
    else
        Pa=0.9-P_max2*sind(delta_old);
        P_max=P_max2;
    end

    y=(delta_t^2)*Pa/M;
    delta_delta=delta_delta+y;
    delta=delta+delta_delta;
    z3=[z3,delta];
    T=[T,t];
    printf('%0.3f sec\t%0.3f\t\t %0.3f\t\t\t%0.3f\t\
       t%0.3f\t\t%0.3f\n',t,P_max,sind(delta),0.9-
       P_max*sind(delta),8.929*(0.9-P_max*sind(delta
       )),delta);
    delta_old=delta;

end

plot(T,z1,T,z2,T,z3);
set(gca(),"grid",[1 1]);
legend('Sustained Fault','Fault cleared at 2.5 cycles','Fault cleared at 6.25 cycles',[,2]);

title('Swing Curves for Example 12.10 for a sustained fault and for clearing in 2.5 and 6.25 cycles','fontsize',2.4);
xlabel('Time (in seconds)—————>');
ylabel('Torque Angle (delta,deg)—————>');
f=get("current_figure")
f.figure_position=[0,15]
f.figure_size=[645,1000]
```

**Scilab code Exa 12.11** Swing Curves For Multimachines

```
//Chapter 12
//Example 12.11
//page 488
//To plot swing curves for fault cleared at 0.275s and 0.08s of a multimachine system
clear;clc;

xd1=%i*0.067;xd2=%i*0.1;

//primitive admittances of the lines
y45=1/(0.018+%i*0.11);    B45=%i*0.113;
```

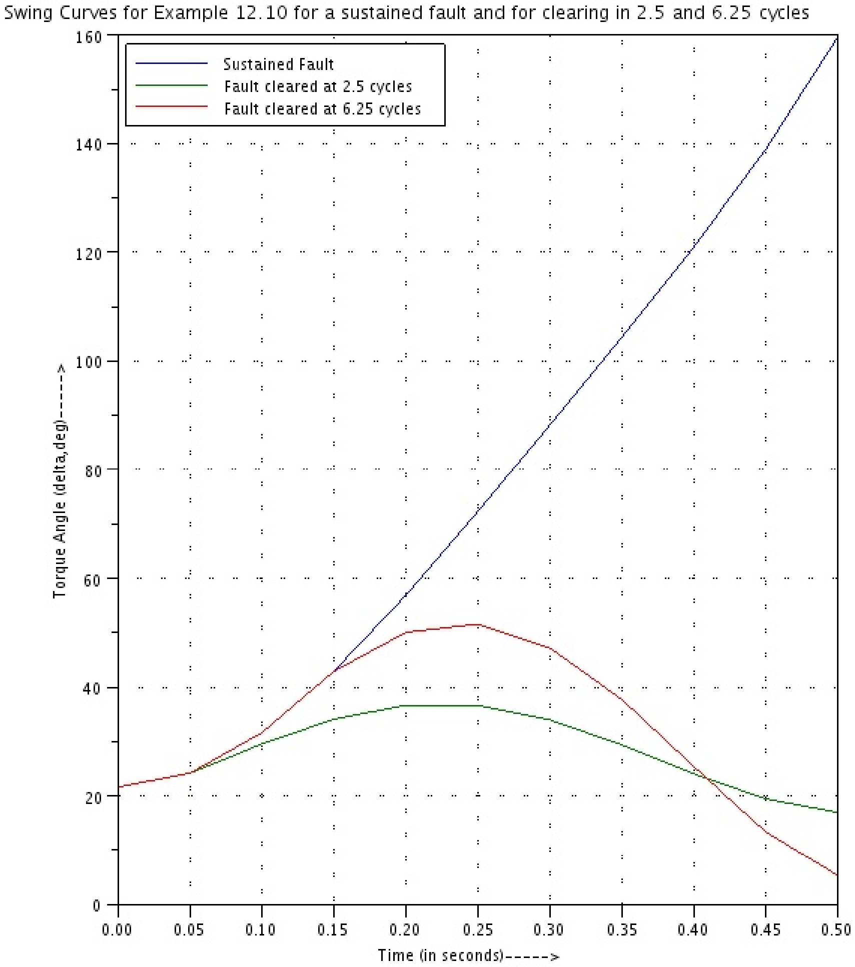


Figure 12.4: Swing Curves For Sustained Fault and Cleared Fault at the Specified Time

```
y51=1/(0.004+%i*0.0235);   B51=%i*0.098;
y41=1/(0.007+%i*0.04);    B41=%i*0.041;
z24=(%i*0.022);
z35=(%i*0.04);

//Bus data and prefault load-flow values in PU
V1=1.0;                       P1=-3.8083;    Q1=-0.2799;    Pl1=0;    Ql2=0;
V2=1.0194+%i*0.1475;        P2=3.25;        Q2=0.6986;        Pl2=0;    Ql2=0;
V3=1.0121+%i*0.1271;        P3=2.10;        Q3=0.3110;        Pl3=0;    Ql3=0;
V4=1.0146+%i*0.0767;         P4=0;           Q4=1.0;          Pl4=1.0;  Ql4=0.44;
V5=1.0102+%i*0.0439;        P5=0;           Q5=0;           Pl5=0.5;  Ql5=0.16;


// To find voltage behind transient reactances before the occurance of fault

I2=(P2-%i*Q2)/(V2');
E2=V2+I2*xd1;
E1=V1;
I3=(P3-%i*Q3)/(V3');
E3=V3+I3*xd2;

//converting loads into their admittances
Yl4=(Pl4-%i*Ql4)/(V4*V4');
Yl5=(Pl5-%i*Ql5)/(V5*V5');

//forming augmented Bus admittance matrix before the occurance of fault
Y11=y41+y51;Y12=0;Y13=0;Y14=-y41;Y15=-y51;
Y21=Y12;Y22=1/(xd1+z24);Y23=0;Y24=-(1/(z24+xd1));Y25=0;
Y31=0;Y32=0;Y33=1/(z35+xd2);Y34=0;Y35=-1/(z35+xd2);
Y41=Y14;Y42=Y24;Y43=Y34;Y44=y41+Yl4+y45+B45+B41-Y24;
```

```
Y45=-y45;
Y51=Y15;Y52=Y25;Y53=Y35;Y54=Y45;Y55=Y15+y45+y51+B45+B51-Y35;

Ybus=[Y11 Y12 Y13 Y14 Y15;
      Y21 Y22 Y23 Y24 Y25;
      Y31 Y32 Y33 Y34 Y35;
      Y41 Y42 Y43 Y44 Y45;
      Y51 Y52 Y53 Y54 Y55];

printf('\n Augmented prefault bus admittance matrix (in PU) is given by\n\n Ybus=\n');
disp(Ybus);
///////to find the Ybus during fault
Ybus_1=Ybus([1:3,5],[1:3,5]);
n=4
for k=1:n-1
    for j=1:n-1
        Ybus_during_fault(k,j)=Ybus_1(k,j)-(Ybus_1(k,n)*Ybus_1(n,j))/Ybus_1(n,n);
    end
end
printf('\n\n\n Bus admittance matrix during fault (in PU) is given by\n\n Ybus_during_fault=\n');
disp(Ybus_during_fault);

//to find Ybus after the fault has been cleared
Y45=0;Y54=0;Y44=Y44-y45-B45;Y55=Y55-y45-B45;
Ybus_2=[Y11 Y12 Y13 Y14 Y15;
      Y21 Y22 Y23 Y24 Y25;
      Y31 Y32 Y33 Y34 Y35;
      Y41 Y42 Y43 Y44 Y45;
      Y51 Y52 Y53 Y54 Y55];

//eliminating node 5 from Ybus_2
n=5
for k=1:n-1
    for j=1:n-1
```

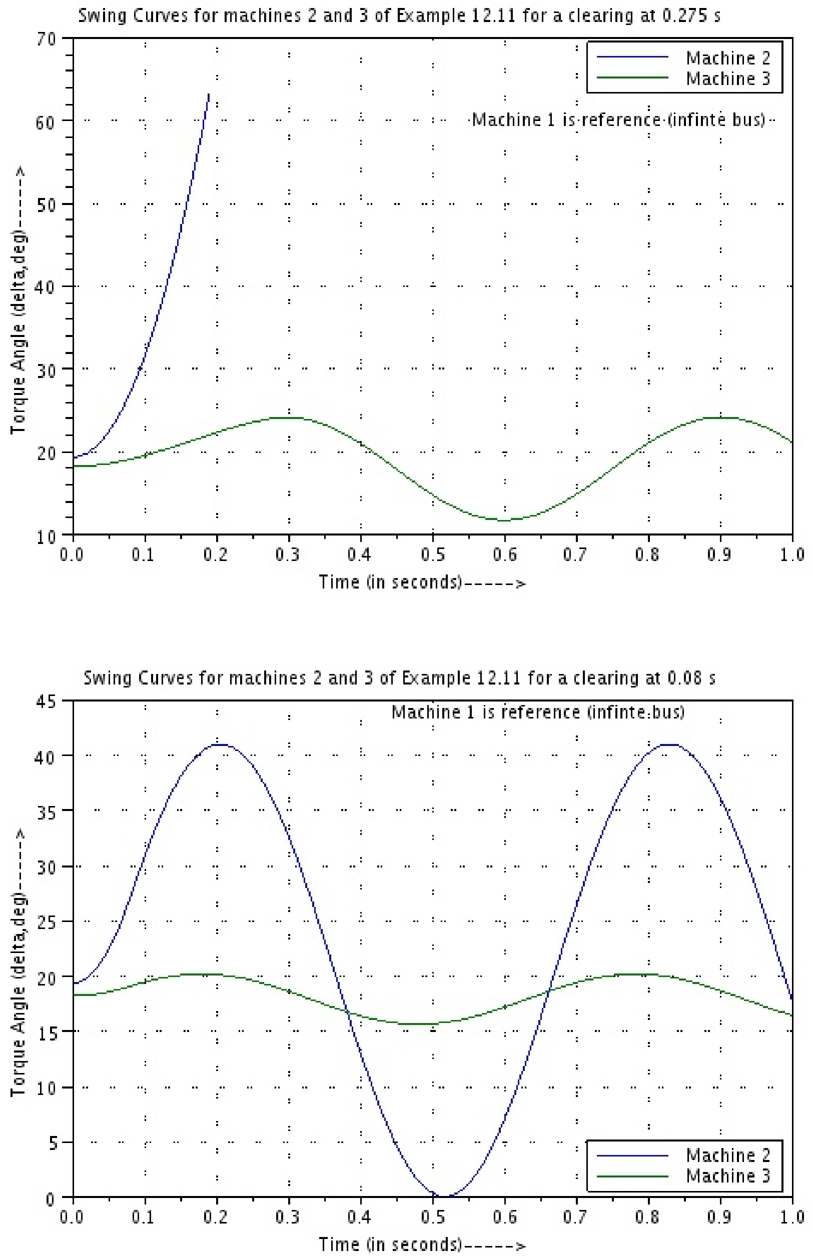
```
        Ybus_3(k,j)=Ybus_2(k,j)-(Ybus_2(k,n)*Ybus_2(n,j))/Ybus_2(n,n);
    end
end

//eliminating node 4 to get post fault Ybus
n=4
for k=1:n-1
    for j=1:n-1
        Ybus_post_fault(k,j)=Ybus_3(k,j)-(Ybus_3(k,n)*Ybus_3(n,j))/Ybus_3(n,n);
    end
end
printf('\n\n\n Bus admittance matrix postfault (in PU) is given by\n\n Ybus_post_fault=\n');
disp(Ybus_post_fault);
printf('\n\n\n');
//During fault power angle equation
delta3=0:0.1:180;
Pe2f=0;
Pe3f=(abs(E3'))^2*real(Ybus_during_fault(3,3))+abs(E1')*abs(E3')*abs(Ybus_during_fault(3,1))*cosd(delta3-atand(imag(Ybus_during_fault(1,3))/real(Ybus_during_fault(1,3))));

//Postfault power angle equations
delta2=0:0.1:180;
Pe2pf=(abs(E2'))^2*real(Ybus_post_fault(2,2))+abs(E1')*abs(E2')*abs(Ybus_post_fault(2,1))*cosd(delta2-atand(imag(Ybus_post_fault(1,2))/real(Ybus_post_fault(1,2))));
Pe3pf=(abs(E3'))^2*real(Ybus_post_fault(3,3))+abs(E1')*abs(E3')*abs(Ybus_post_fault(3,1))*cosd(delta3-atand(imag(Ybus_post_fault(1,3))/real(Ybus_post_fault(1,3))));

//mechanical inputs which are assumed to be constant are given by
```

```
Pm2=max(real(E2*I2'));
Pm3=max(real(E3*I3'));

//xdot function defining the swing equations of each of the machines
function xdot=mac2(t,x,tc)
    xdot(1)=x(2);
    if t>tc then
      xdot(2)=180*50*(Pm2-(0.6012+8.365*sind(x(1)-1.662)))/12;//swing equation after clearing the fault
    else
        xdot(2)=180*50*(Pm2-(0))/12;  //swing equation before clearing the fault
    end

endfunction

function xdot=mac3(t,x,tc)
    xdot(1)=x(2);
    if t>tc then
      xdot(2)=180*50*(Pm3-(0.1823+6.5282*sind(x(1)-0.8466)))/9;//swing equation after clearing the fault
    else
        xdot(2)=180*50*(Pm3-(0.1561+5.531*sind(x(1)-0.755)))/9; //swing equation before clearing the fault
    end

endfunction

//to find the solution of swing equation to draw the swing curves

//to draw the swing curves for machines 2 and 3 for example12.11 for clearing at 0.275 sec
subplot(2,1,1)
```

```
x_1_0=[19.354398,0]';t0=0; T=0:0.01:1;T=T';
x_2_0=[18.2459,0]';tc=0.275;
sol1=ode(x_1_0,t0,T,mac2);
sol2=ode(x_2_0,t0,T,mac3);

plot(T(1:20),sol1(1,1:20)',T,sol2(1,:)');
set(gca(),"grid",[1 1]);
legend('Machine 2','Machine 3',[,1]);
title('Swing Curves for machines 2 and 3 of Example 12.11 for a clearing at '+string(tc)+' s');
xstring(0.55,59,'Machine 1 is reference (infinte bus)');
xlabel('Time (in seconds)------>');
ylabel('Torque Angle (delta,deg)------>');


//to draw the swing curves for machines 2 and 3 for example12.11 for clearing at 0.08 sec
subplot(2,1,2)
x_1_0=[19.354398,0]';t0=0; T=0:0.01:1;T=T';
x_2_0=[18.2459,0]';tc=0.08;
sol1=ode(x_1_0,t0,T,mac2);
sol2=ode(x_2_0,t0,T,mac3);

plot(T,sol1(1,:)',T,sol2(1,:)');
set(gca(),"grid",[1 1]);
legend('Machine 2','Machine 3',[,4]);
title('Swing Curves for machines 2 and 3 of Example 12.11 for a clearing at '+string(tc)+' s');
xstring(0.44,43,'Machine 1 is reference (infinte bus)');
xlabel('Time (in seconds)------>');
ylabel('Torque Angle (delta,deg)------>');

f=get("current_figure");
f.figure_position=[0,15];
f.figure_size=[565,1000];
```

**Scilab code Exa 12.12** Swing Curves For Three Pole and Single Pole Switching

```
//Chapter 12
//Example 12.12
//page 500
//To plot swing curves for single pole and three pole switching
clear;clc;

Xg0=0.1;Xg1=0.3;Xg2=0.15;E=1.2;H=4.167;
Xt=0.1;
Xl0=1.0;Xl1=0.3;Xl2=0.3;V=1;

//transfer reactance during LG fault(fault not cleared) by star delta transformation is given by
X12_fault=1.45;

//transfer reactance after LG faulted line open is given by
X12_fault_open=1.22;

//transfer reactance when all the lines are healthy is given by
X12_healthy=0.8;

//power angle equations
delta=0:0.1:180;

//Prefault condition
```


Swing Curves for machines 2 and 3 of Example 12.11 for a clearing at 0.275 s

Swing Curves for machines 2 and 3 of Example 12.11 for a clearing at 0.08 s

Figure 12.5: Swing Curves For Multimachines

```
Pe1=(E*V)*sind(delta)/X12_healthy;
//for an initial load of 1PU
delta0=asind(1/1.5);

//during fault
Pe2=(E*V)*sind(delta)/X12_fault;

//during single pole switching
Pe3=(E*V)*sind(delta)/X12_fault_open;

//during three pole switching
Pe4=0;

//after reclosure
Pe5=Pe1;

Pm=1.0;

//xdot function defining the swing equations of machine during single poling
function xdot=mac_1_pole(t,x,tc,tr)
    xdot(1)=x(2);
    if (t<=tc) then
      xdot(2)=180*50*(Pm-(0.827*sind(x(1))))/12;  //swing equation before clearing the faulted line
    elseif (t>tc)&(t<tr) then
      xdot(2)=180*50*(Pm-(0.985*sind(x(1))))/12;//swing equation during single pole switching
    elseif (t>=tr) then
      xdot(2)=180*50*(Pm-(1.5*sind(x(1))))/12; //after reclosure
    end
endfunction

//xdot function defining the swing equations of machine during three poling
function xdot=mac_3_pole(t,x,tc,tr)
```

```
    xdot(1)=x(2);
    if (t>tc)&(t<tr) then
      xdot(2)=180*50*(Pm-0)/4.167;//swing equation during three pole switching
    elseif (t<=tc) then
      xdot(2)=180*50*(Pm-(0.827*sind(x(1))))/4.167; //swing equation before clearing the faulted line
    elseif (t>=tr) then
      xdot(2)=180*50*(Pm-(1.5*sind(x(1))))/4.167; //after reclosure
    end
endfunction

//to find the solution of swing equation to draw the swing curves

//to draw the swing curves for three pole switching with reclosure
subplot(2,1,1)
x_1_0=[41.8,0]';t0=0; T=0:0.001:0.65;T=T';
tc=0.075;tr=0.325;
sol1=ode(x_1_0,t0,T,mac_3_pole);
plot(T,sol1(1,:)');
set(gca(),"grid",[1 1]);
title('Swing Curve for three pole switching at '+string(tc)+' s'+' and reclosure at '+string(tr)+' s','fontsize',3);
xset("font size",3)
xstring(0.2,300,'MACHINE UNSTABLE');
xlabel('Time (in seconds)------>');
ylabel('Torque Angle (delta,deg)------>');

//to draw the swing curves for single pole switching with reclosure
subplot(2,1,2)
x_1_0=[41.8,0]';t0=0; T=0:0.001:2.2;T=T';
tc=0.075;tr=0.325;
```

```
sol2=ode(x_1_0,t0,T,mac_1_pole);
plot(T,sol2(1,:)');
set(gca(),"grid",[1 1]);
title('Swing Curve for single pole switching at '+string(tc)+' s'+' and reclosure at '+string(tr)+' s','fontsize',3);
xset("font size",3)
xstring(1.2,50,'MACHINE STABLE');
xlabel('Time (in seconds)------>');
ylabel('Torque Angle (delta,deg)------>');

f=get("current_figure");
f.figure_position=[0,15];
f.figure_size=[560,1000];
```

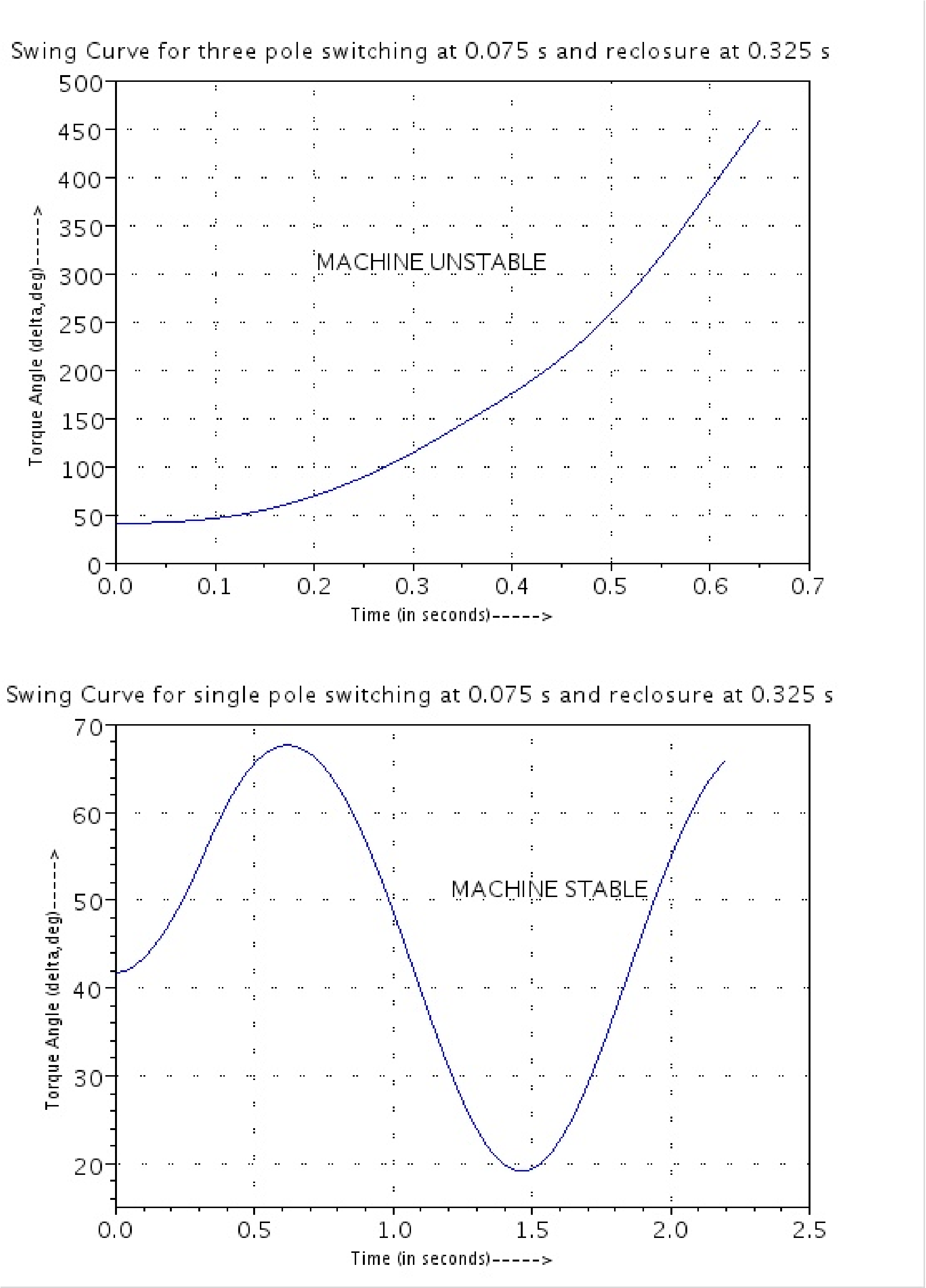


Figure 12.6: Swing Curves For Three Pole and Single Pole Switching

# Chapter 13

# Power System Security

**Scilab code Exa 13.1** Generation Shift Factors and Line Outage Distribution Factors

```
//Chapter 13
//Example 13.1
//page 522
//To find the generation shift factors and the line outage distribution factors
clear;clc;

//this problem can be thought to be solved by using gauss−siedel method using Zbus(X matrix given in table 13.1),but then in this method we need total line charging admittances to ground at each bus. Hence we cant solve this problem only using the given table 13.1,And we can use gauss−siedel method using Ybus by taking the values of impedances and line charging admittances of the system which is taken from the textbook "[1] Computer Methods in Power System Analysis,Stagg and El−Abiad,Page No 284"
//−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
```

```

// Function to form the Ybus for primitve admittance values and line charging admittance values
function Ybus=formYbus(y_l,y_lc)
    Ybus=[y_l(1)+y_l(2)+y_lc(1)+y_lc(2)    -y_l(1)    -y_l(2)    0    0;
          -y_l(1)    y_l(1)+y_l(3)+y_l(4)+y_l(5)+y_lc(1)+y_lc(3)+y_lc(4)+y_lc(5)    -y_l(3)    -y_l(4)    -y_l(5);
          -y_l(2)    -y_l(3)    y_l(2)+y_l(3)+y_l(6)+y_lc(2)+y_lc(3)+y_lc(6)    -y_l(6)    0;
          0    -y_l(4)    -y_l(6)    y_l(6)+y_l(4)+y_l(7)+y_lc(6)+y_lc(4)+y_lc(7)    -y_l(7);
          0    -y_l(5)    0    -y_l(7)    y_l(5)+y_l(7)+y_lc(5)+y_lc(7)];
endfunction

//Function to incorporate load flow analysis for a given system
function P_line=load_flow(E,Pg,Qg,Pl,Ql,y_l,y_lc)

    //to retrieve Ybus for the given network parameters
    Y=formYbus(y_l,y_lc);

    //to form primitive admittance matrix and primitive line charging admittances that required later in the program
    yl=[0 y_l(1) y_l(2) 0 0;
       y_l(1) 0 y_l(3) y_l(4) y_l(5);
       y_l(2) y_l(3) 0 y_l(6) 0;
       0 y_l(4) y_l(6) 0 y_l(7);
       0 y_l(5) 0 y_l(7) 0];
    yc=[0 y_lc(1) y_lc(2) 0 0;
       y_lc(1) 0 y_lc(3) y_lc(4) y_lc(5);
```

```
      y_lc(2) y_lc(3) 0 y_lc(6) 0;
      0 y_lc(4) y_lc(6) 0 y_lc(7);
      0 y_lc(5) 0 y_lc(7) 0];

    // to optimize the evaluation, constants like KLs and YLs are evaluated only once outside the loop
    KL2=((Pg(2)-Pl(2))-(Qg(2)-Ql(2)))/Y(2,2);
    KL3=((Pg(3)-Pl(3))-(Qg(3)-Ql(3)))/Y(3,3);
    KL4=((Pg(4)-Pl(4))-(Qg(4)-Ql(4)))/Y(4,4);
    KL5=((Pg(5)-Pl(5))-(Qg(5)-Ql(5)))/Y(5,5);

    YL21=Y(2,1)/Y(2,2);    YL23=Y(2,3)/Y(2,2);  YL24=Y(2,4)/Y(2,2);    YL25=Y(2,5)/Y(2,2);
    YL31=Y(3,1)/Y(3,3);    YL32=Y(3,2)/Y(3,3);  YL34=Y(3,4)/Y(3,3);
    YL42=Y(4,2)/Y(4,4);    YL43=Y(4,3)/Y(4,4);  YL45=Y(4,5)/Y(4,4);
    YL52=Y(5,2)/Y(5,5);    YL54=Y(5,4)/Y(5,5);

    //to calculate bus voltages (Refer[1] stagg,pg 285)
    n=100;
    for i=1:n
        E(1)=E(1);
        E(2)=(KL2/E(2)')-YL21*E(1)-YL23*E(3)-YL24*E(4)-YL25*E(5);
        E(3)=(KL3/E(3)')-YL31*E(1)-YL32*E(2)-YL34*E(4);
        E(4)=(KL4/E(4)')-YL42*E(2)-YL43*E(3)-YL45*E(5);
        E(5)=(KL5/E(5)')-YL52*E(2)-YL54*E(4);
    end
    // to calculate line flows(Refer[1] stagg,pg 291)
    for i=1:5
        for j=1:5
            S(i,j)=E(i)'*(E(i)-E(j))*yl(i,j)+E(i)'*E
```

```
                (i)*yc(i,j);
        end
    end
    P_line=conj(S); //since P_line=P-jQ=conj(S)

endfunction
//////////////////////////////////////////////////////////////////////
//First we will calculate the line flows for the system which operating under normal condition (without any congincy)[taken as Base system for comparision]//
//////////////////////////////////////////////////////////////////////
//ypq                         y'pq/2            line no        Buscode(p-q)
yl1=1/(0.02+%i*0.06);    ylc_1=%i*0.030;    //l=1          line 1-2
yl2=1/(0.08+%i*0.24);    ylc_2=%i*0.025;    //l=2          line 1-3
yl3=1/(0.06+%i*0.18);    ylc_3=%i*0.020;    //l=3          line 2-3
yl4=1/(0.06+%i*0.18);    ylc_4=%i*0.020;    //l=4          line 2-4
yl5=1/(0.04+%i*0.12);    ylc_5=%i*0.015;    //l=5          line 2-5
yl6=1/(0.01+%i*0.03);    ylc_6=%i*0.010;    //l=6          line 3-4
yl7=1/(0.08+%i*0.24);    ylc_7=%i*0.025;    //l=7          line 4-5

y_l_vector=[yl1 yl2 yl3 yl4 yl5 yl6 yl7];
y_lc_vector=[ylc_1 ylc_2 ylc_3 ylc_4 ylc_5 ylc_6 ylc_7];


```

```
//Assumed voltage                 Generation                          load                  Buscode
//                                MW                  MVAR            MW              MVAR
E1=1.06+%i*0;                     Pg1=0;              Qg1=%i*0;       Pl1=0;          Ql1=%i*0;         //1
E2=1+%i*0;                        Pg2=0.4;            Qg2=%i*0.3;     Pl2=0.2;        Ql2=%i*0.1;       //2
E3=1+%i*0;                        Pg3=0;              Qg3=%i*0;       Pl3=0.45;       Ql3=%i*0.15;      //3
E4=1+%i*0;                        Pg4=0;              Qg4=%i*0;       Pl4=0.40;       Ql4=%i*0.05;      //4
E5=1+%i*0;                        Pg5=0;              Qg5=%i*0;       Pl5=0.60;       Ql5=%i*0.10;      //5

E=[E1 E2 E3 E4 E5];    Pg=[Pg1 Pg2 Pg3 Pg4 Pg5];    Qg=[Qg1 Qg2 Qg3 Qg4 Qg5];
Pl=[Pl1 Pl2 Pl3 Pl4 Pl5];    Ql=[Ql1 Ql2 Ql3 Ql4 Ql5];

P_base=load_flow(E,Pg,Qg,Pl,Ql,y_l_vector,y_lc_vector);
P_base=P_base*100; //converting back to MW and MVARs




////////////////////////////////////////////////////////////////////
//To find generation shift factor let us remove the generator at each of PV buses and calculate line flows//
////////////////////////////////////////////////////////////////////

```

```
//(i)when generator at slack bus trips
  Pg1_old=Pg1; //required for the calculation of change in MWs
  Pg1=0;Qg1=0; //generation remains same
  Pg=[Pg1 Pg2 Pg3 Pg4 Pg5];    Qg=[Qg1 Qg2 Qg3 Qg4 Qg5]; //updating the changed values

 //conducting load flow studies
  P_G_1=load_flow(E,Pg,Qg,Pl,Ql,y_l_vector,y_lc_vector);
  P_G_1=P_G_1*100; //converting back to MW and MVARs
  alpha1=(real(P_G_1)-real(P_base))/((Pg1_old-Pg1+0.001)*100); //0.001 is added to eliminate divide by zero error
  alpha1=tril(alpha1); //only lower triangular matrix is required
  l1=[alpha1(2,1) alpha1(3,1) alpha1(3,2) alpha1(4,2) alpha1(5,2) alpha1(4,3) alpha1(5,4)];
//(ii)When generator at Bus2 trips
  Pg2_old=Pg2; //required for the calculation of change in MWs
  Pg2=0;Qg2=0; Pg1=0;Qg1=0;
  Pg=[Pg1 Pg2 Pg3 Pg4 Pg5];    Qg=[Qg1 Qg2 Qg3 Qg4 Qg5]; //updating the changed values

 //conducting load flow studies
  P_G_2=load_flow(E,Pg,Qg,Pl,Ql,y_l_vector,y_lc_vector);
  P_G_2=P_G_2*100; //converting back to MW and MVARs
  alpha2=(real(P_G_2)-real(P_base))/((Pg2_old-Pg2)*100);
  alpha2=tril(alpha2); //only lower triangular matrix is required
  l2=[alpha2(2,1) alpha2(3,1) alpha2(3,2) alpha2(4,2) alpha2(5,2) alpha2(4,3) alpha2(5,4)];
//To print the results of generator shift factors

printf('Generator Shift Factor for Five-bus System\n
');
printf('------------------------------------------------------------------------\n');
printf('Lines\t\t\t Bus 1 \t\t\tBus 2\n');
printf('------------------------------------------------------------------------\n');
for i=1:7
        printf('l = %d\t\t\t  %d\t\t\t%0.4f\n',i,l1(i),l2(i));
end
printf('------------------------------------------------------------------------\n');
//////////////////////////////////////////////////////////////////////////
//To find Line Outage Distribution Factors let us remove each line and calculate the line flows//
//////////////////////////////////////////////////////////////////////////

//changing the network back to normal system
Pg2=0.4;           Qg2=%i*0.3;

//copying the original values of the network parameters
y_l_vector_normal=y_l_vector; y_lc_vector_normal=y_lc_vector;
  Pg=[Pg1 Pg2 Pg3 Pg4 Pg5];    Qg=[Qg1 Qg2 Qg3 Qg4 Qg5]; //updating the changed values
//when jth line trips the load flow analysis is done as follows
for j=1:7
    y_l_vector(j)=0;y_lc_vector(j)=0;
    P_L=load_flow(E,Pg,Qg,Pl,Ql,y_l_vector,
        y_lc_vector); //load flow anlysis
     P_L=P_L*100;
     select j,
       case 1 then fi0=P_base(2,1),
       case 2 then fi0=P_base(3,1),
       case 3 then fi0=P_base(3,2),
       case 4 then fi0=P_base(4,2),
       case 5 then fi0=P_base(5,2),
       case 6 then fi0=P_base(4,3),
       case 7 then fi0=P_base(5,4),
     end
    d0=(real(P_L)-real(P_base))/real(fi0);
    d(:,j)=[d0(2,1); d0(3,1); d0(3,2); d0(4,2); d0(5,2); d0(4,3); d0(5,4)];
    y_l_vector=y_l_vector_normal; y_lc_vector=y_lc_vector_normal; //changing the system back to normal
end

//when a line trips power flow in that line is zero, this is not accounted in load flow. So accounting this by making all diagonal elemnts of d=0
for i=1:7
    d(i,i)=0;
end

//To print the results of line outage distribution factors
printf('\n\n\n\nLine Outage Distribution Factor for Five-bus System\n');
printf('------------------------------------------------------------------------------------n');
printf('Lines \t\tj=1\t\tj=2\t\tj=3\t\tj=4\t\tj=5\t\tj=6\t\tj=7\n');
printf('------------------------------------------------------------------------------------n');
```

```
for l=1:7
    printf('l = %d\t\t%0.4f\t\t%0.4f\t\t%0.4f\t\t%0.4f\t\t%0.4f\t\t%0.4f\t\t%0.4f\n',l,d(l,1),d(l,2),d(l,3),d(l,4),d(l,5),d(l,6),d(l,7));
end
printf('
   ----------------------------------------------------------------------------
   n\n\n');
```

# Chapter 14

# An Introduction to State Estimation of Power Systems

**Scilab code Exa 14.1** Estimation of random variables

```
//Chapter 14
//Example 14.1
//page 533
//To estimate the values of the random variables x1
   and x2
clear;clc;

H=[1 0;0 1;1 1];   //given matrix
k=inv(H'*H)*H'; // from eq 14.9
y=['y1';'y2';'y3'];
k=string(k);
x=[k(1,1)+y(1,1)+k(1,2)+y(2,1)+"+"+k(1,3)+y(3,1) ;k
   (2,1)+y(1,1)+"+"+k(2,2)+y(2,1)+"+"+k(2,3)+y(3,1)
   ];
printf('Estimate of x =\n');
disp(x);
```

**Scilab code Exa 14.2** Estimation of random variables using WLSE

```
//Chapter 14
//Example 14.2
//page 534
//To estimate the values of the random variables x1 and x2 using WLSE
clear;clc;

w=diag([0.1;1;0.1]); //assumed matrix
H=[1 0;0 1;1 1];  //given matrix
k=inv(H'*w*H)*H'*w; // from eq 14.12b
y=['y1';'y2';'y3'];
Px=k*k';
k=string(k);
x=[k(1,1)+y(1,1)+k(1,2)+y(2,1)+"+"+k(1,3)+y(3,1) ;k(2,1)+y(1,1)+"+"+k(2,2)+y(2,1)+"+"+k(2,3)+y(3,1)];
printf('The weighted least square s estimate of the vector x =\n');
disp(x);
printf('\n\nThe matrix k is in this case found to be \n');
disp(k);
//covariance of measurement is assumed is assumed to be unit matrix
printf('\n\nThe covariance of the estimation error is obtained as Px=\n');
disp(Px);

printf('\n\n\n Now choosing W=1\n');
w=diag([1;1;1]); //assumed matrix
```

```
H=[1 0;0 1;1 1];  //given matrix
k=inv(H'*w*H)*H'*w; // from eq 14.12b
Px=k*k';
printf('\n\nThe matrix k is in this case found to be \n');
disp(k);
printf('\n\nThe covariance of the estimation error is obtained as Px=\n');
disp(Px);
```

**Scilab code Exa 14.3** Estimation of random variables using WLSE 2

```
//Chapter 14
//Example 14.3
//page 538
//To estimate the values of the random variables x1 and x2 using WLSE
clear;clc;
i=0; x=1;y=8.5
printf('------------------------------------------------\n');
printf('iteration\t\tx(I)\n');
printf('------------------------------------------------\n');
printf('\t%d\t\t%0.3f\n',i,x);
for i=1:1:10
    k=(1/3)*x^-2      //expression for the value of k has been printed wrongly in the textbook
    x=x+(k)*(y-x^3);
    printf('\t%d\t\t%0.3f\n',i,x);

end
```

# Chapter 17

# Voltage Stability

**Scilab code Exa 17.1** Reactive power sensitivity

```
//Chapter 17
//Example 17.1
//page 602
//To find reactive power sensitivity at the bus
clear;clc;
Q_nom=1; //given
Ksh=0.8; V=1.0; //assumed
Qnet=(V^2-Ksh*V^2)*Q_nom;
//senstivity=dQnet/dV
s=2*V-2*V*Ksh;
printf('Reactive power Sensitivity at the bus is = %0.2f pu',s);
```

**Scilab code Exa 17.2** Capacity of static VAR compensator

```
//Chapter 17
//Example 17.2
```

```
//page 602
//To find capacity of static VAR compensator
clear;clc;

delta_V=5/100; //allowable voltage fluctuation
S_sc=5000; //system short circuit capacity in MVA
delta_Q=delta_V*S_sc; //size of the compensator
printf('The capacity of the static VAR compensator
    is +%d MVAR',delta_Q);
```