

Scilab Textbook Companion for
Modern Power System Analysis
by D. P. Kothari And I. J. Nagrath¹

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

May 19, 2016

¹Funded 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	4
1 Introduction	6
2 Inductance and Resistance of Transmission Lines	12
3 Capacitance of Transmission Lines	20
4 Representation of Power System Components	24
5 Characteristics and Performance of Power Transmission Lines	31
6 Load Flow Studies	48
7 Optimal System Operation	69
8 Automatic Generation and Voltage Control	85
9 Symmetrical Fault Analysis	87
10 Symmetrical Components	107
11 Unsymmetrical Fault Analysis	114
12 Power System Stability	134
13 Power System Security	167
14 An Introduction to State Estimation of Power Systems	176

List of Scilab Codes

Exa 1.1	Example 1	6
Exa 1.3	Example 3	8
Exa 1.4	Example 4	9
Exa 1.5	Example 5	10
Exa 2.1	self GMD Calculation	12
Exa 2.2	Reactance Of ACSR conductors	13
Exa 2.3	Inductance Of Composite Conductor Lines	14
Exa 2.5	VoltageDrop and FluxLinkage Calculations	15
Exa 2.6	Mutual Inductance Calculation	17
Exa 2.7	Bundled Conductor Three Phase Line	17
Exa 3.1	Capacitance of a single phase line	20
Exa 3.2	Charging current of a threephase line	21
Exa 3.3	Double circuit three phase transmission line	22
Exa 4.1	Per Unit Reactance Diagram	24
Exa 4.2	Per Unit Calculation	25
Exa 4.3	Excitation EMF and Reactive Power Calculation	27
Exa 4.4	Power Factor And Load Angle Calculation	28
Exa 5.1	SendingEnd voltage and voltage regulation	31
Exa 5.2	Voltage at the power station end	32
Exa 5.3	Problem with mixed end condition	34
Exa 5.4	Medium Transmission line system	35
Exa 5.5	Maximum permissible length and and Frequency	36
Exa 5.6	Incident and Reflected voltages	37
Exa 5.7	Tabulate characteristics using different methods	38
Exa 5.8	Torque angle and Station powerfactor	41
Exa 5.9	Power Voltage and Compensating equipment rating	43
Exa 5.10	MVA rating of the shunt reactor	44
Exa 5.11	SendingEnd voltage and maximum power delivered	47

Exa 6.1	Ybus using singular transformation	48
Exa 6.2	Ybus of a sample system	49
Exa 6.3	Approximate load flow solution	50
Exa 6.4	Bus voltages using GS iterations	53
Exa 6.5	Reactive power injected using GS iterations	54
Exa 6.6	Load flow solution using the NR method	57
Exa 6.7	Ybus after including regulating transformer	62
Exa 6.8	Decoupled NR method and FDLF method	64
Exa 7.1	Incremental cost and load sharing	69
Exa 7.2	Savings by optimal scheduling	72
Exa 7.3	Economical operation	74
Exa 7.4	Generation and losses incurred	75
Exa 7.5	Savings on coordination of losses	77
Exa 7.6	Loss formula coefficients calculation	80
Exa 7.7	Optimal generation schedule for hydrothermal system	81
Exa 8.1	Frequency change Calculation	85
Exa 8.2	Load sharing and System Frequency	86
Exa 9.1	Fault Current Calculation	87
Exa 9.2	Subtransient and Momentary current Calculation	89
Exa 9.3	Subtransient Current Calculation	93
Exa 9.4	Maximum MVA Calculation	94
Exa 9.5	Short Circuit Solution	97
Exa 9.6	Short Circuit Solution using Algorithm	99
Exa 9.7	Current Injection Method	102
Exa 9.8	Zbus matrix building using Algorithm	103
Exa 9.9	PostFault Currents and Voltages Calculation	105
Exa 10.1	Symmetrical components of line currents Calculation	107
Exa 10.2	Sequence Network of the System	110
Exa 10.3	Zero sequence Network	111
Exa 10.4	Zero Sequence Network	112
Exa 11.1	LG and 3Phase faults Comparision	114
Exa 11.2	Grounding Resistor voltage and Fault Current	115
Exa 11.3	Fault and subtransient currents of the system	117
Exa 11.4	LL Fault Current	121
Exa 11.5	Double line to ground Fault	123
Exa 11.6	Bus Voltages and Currents Calculations	127
Exa 11.7	Short Circuit Current Calculations	130

Exa 12.1	Calculation of stored kinetic energy and rotor acceleration	134
Exa 12.2	steady state power limit	135
Exa 12.3	Maximum Power Transferred	137
Exa 12.4	Acceleration and Rotor angle	139
Exa 12.5	Frequency Of Natural Oscilations	140
Exa 12.6	Steady State Power Limit 2	141
Exa 12.7	Critical Clearing Angle	142
Exa 12.8	Critical Clearing Angle 2	144
Exa 12.9	Critical Clearing Angle 3	147
Exa 12.10	Swing Curves For Sustained Fault and Cleared Fault at the Specified Time	148
Exa 12.11	Swing Curves For Multimachines	154
Exa 12.12	Swing Curves For Three Pole and Single Pole Switching	161
Exa 13.1	Generation Shift Factors and Line Outage Distribution Factors	167
Exa 14.1	Estimation of random variables	176
Exa 14.2	Estimation of random variables using WLSE	177
Exa 14.3	Estimation of random variables using WLSE 2	178
Exa 17.1	Reactive power sensitivity	179
Exa 17.2	Capacity of static VAR compensator	179

List of Figures

5.1	MVA rating of the shunt reactor	46
7.1	Incremental cost and load sharing	73
9.1	Fault Current Calculation	90
9.2	Subtransient and Momentary current Calculation	92
9.3	Subtransient Current Calculation	95
9.4	Maximum MVA Calculation	97
9.5	Short Circuit Solution	99
9.6	Current Injection Method	103
11.1	LG and 3Phase faults Comparison	116
11.2	Grounding Resistor voltage and Fault Current	118
11.3	Fault and subtransient currents of the system	122
11.4	LL Fault Current	124
11.5	Double line to ground Fault	126
12.1	Maximum Power Transferred	139
12.2	Critical Clearing Angle	145
12.3	Critical Clearing Angle 2	147
12.4	Swing Curves For Sustained Fault and Cleared Fault at the Specified Time	155
12.5	Swing Curves For Multimachines	162
12.6	Swing Curves For Three Pole and Single Pole Switching	166

Chapter 1

Introduction

Scilab code Exa 1.1 Example 1

```
1 //Chapter 1
2 //Example 1.1
3 //page 5
4 clear;clc;
5 fl=760e3;
6 pf=0.8;
7 lsg=0.05;
8 csg=60;
9 depre=0.12;
10 hpw=48;
11 lv=32;
12 hv=30;
13 pkwhr=0.10;
14
15 md=f1/pf;
16 printf('Maximum Demand= %.1 f kVA \n\n',md/1000);
17
18 //calculation for tariff (b)
19
20 printf('Loss in switchgear=%.2 f %% \n\n',lsg*100);
21 input_demand=md/(1-lsg);
```

```

22 input_demand=input_demand/1000;
23 cost_sw_ge=input_demand*60;
24 depreciation=depre*cost_sw_ge;
25 fixed_charges=hv*input_demand;
26 running_cost=input_demand*pf*hpw*52*pkwhr; //52 weeks
    per year
27 total_b=depreciation + fixed_charges + running_cost;
28 printf('Input Demand= %.1f kVA \n\n',input_demand);
29 printf('Cost of switchgear=Rs %d\n\n',cost_sw_ge);
30 printf('Annual charges on depreciation=Rs %d \n\n',
    depreciation);
31 printf('Annual fixed charges due to maximum demand
    corresponding to triff(b)=Rs %d \n\n',
    fixed_charges);
32 printf('Annual running cost due to kWh consumed=Rs
    %d \n\n',running_cost);
33 printf('Total charges/annum for tariff(b) = Rs %d\n\n
    n',total_b)
34
35 //calculation for tariff (a)
36 input_demand=md;
37 input_demand=input_demand/1000;
38 fixed_charges=lv*input_demand;
39 running_cost=input_demand*pf*hpw*52*pkwhr;
40 total_a=fixed_charges + running_cost;
41 printf('maximum demand corresponding to tariff(a) =
    %.f kVA \n\n',input_demand);
42 printf('Annual fixed charges=Rs %d \n\n',
    fixed_charges);
43 printf('Annual running charges for kWh consumed = Rs
    %d \n\n',running_cost);
44 printf('Total charges/annum for tariff(a) = Rs %d \n
    \n',total_a);
45 if(total_a > total_b)
46     printf('Therefore, tariff(b) is economical\n\n\n
    ');
47 else
48     printf('Therefore, tariff(a) is economical\n\n\n

```

');

Scilab code Exa 1.3 Example 3

```
1 //Chapter 1
2 //Example 1.3
3 //page 7
4 clear;clc;
5 md=25;
6 lf=0.6;
7 pcf=0.5;
8 puf=0.72;
9
10 avg_demand=lf*md;
11 installed_capacity=avg_demand/pcf;
12 reserve=installed_capacity-md;
13 daily_ener=avg_demand*24;
14 ener_inst_capa=installed_capacity*24;
15 max_energy=daily_ener/puf;
16
17 printf('Average Demand= %.2f MW \n\n',avg_demand);
18 printf('Installed capacity= %.2f MW \n\n',
    installed_capacity);
19 printf('Reserve capacity of the plant= %.2f MW \n\n',
    ,reserve);
20 printf('Daily energy produced= %d MWh \n\n',
    daily_ener);
21 printf('Energy corresponding to installed capacity
    per day= %d MWh \n\n',ener_inst_capa);
22 printf('Maximum energy that could be produced = %d
    MWh/day \n\n',max_energy);
```

Scilab code Exa 1.4 Example 4

```
1 //Chapter 1
2 //Example 1.2
3 //page 6
4 clear;clc;
5 md=20e3;
6 unit_1=14e3;
7 unit_2=10e3;
8 ener_1=1e8;
9 ener_2=7.5e6;
10 unit1_time=1;
11 unit2_time=0.45;
12
13 annual_lf_unit1=ener_1/(unit_1*24*365);
14 md_unit_2=md-unit_1;
15 annual_lf_unit2=ener_2/(md_unit_2*24*365);
16 lf_unit_2=ener_2/(md_unit_2*unit2_time*24*365);
17 unit1_cf=annual_lf_unit1;
18 unit1_puf=unit1_cf;
19 unit2_cf=ener_2/(unit_2*24*365);
20 unit2_puf=unit2_cf/unit2_time;
21 annual_lf=(ener_1+ener_2)/(md*24*365);
22
23
24 printf('Annual load factor for Unit 1 = %.2f %% \n\n',
        annual_lf_unit1*100);
25 printf('The maximum demand on Unit 2 is %d MW \n\n',
        md_unit_2/1000);
26 printf('Annual load factor for Unit 2 = %.2f %% \n\n',
        annual_lf_unit2*100);
27 printf('Load factor of Unit 2 for the time it takes
```

```

    the load= %.2f %% \n\n',lf_unit_2*100);
28 printf('Plant capacity factor of unit 1 = %.2f %% \n
    \n',unit1_cf*100);
29 printf('Plant use factor of unit 1 = %.2f %% \n\n',
    unit1_puf*100);
30 printf('Annual plant capacity factor of unit 2 = %.2
    f %% \n\n',unit2_cf*100);
31 printf('Plant use factor of unit 2 = %.2f %% \n\n',
    unit2_puf*100);
32 printf('The annual load factor of the total plant =
    %.2f %% \n\n',annual_lf*100);

```

Scilab code Exa 1.5 Example 5

```

1 //Chapter 1
2 //Example 1.2
3 //page 6
4 clear;clc;
5
6 c1_md_6pm=5;      c1_d_7pm=3;      c1_lf=0.2;
7 c2_md_11am=5;    c2_d_7pm=2;      c2_avg_load=1.2;
8 c3_md_7pm=3;     c3_avg_load=1;
9
10 md_system=c1_d_7pm + c2_d_7pm + c3_md_7pm;
11 sum_mds=c1_md_6pm + c2_md_11am + c3_md_7pm;
12 df=sum_mds/md_system;
13
14 printf('Maximum demand of the system is %d kW at 7p.
    m \n',md_system);
15 printf('Sum of the individual maximum demands = %d
    kW \n',sum_mds);
16 printf('Diversity factor= %.3f \n\n',df);
17

```

```
18 c1_avg_load=c1_md_6pm*c1_lf;
19 c2_lf=c2_avg_load/c2_md_11am;
20 c3_lf=c3_avg_load/c3_md_7pm;
21
22 printf('Consumer1 -->\t Avg_load= %.2 f kW \t LF= %.1
      f %% \n',c1_avg_load,c1_lf*100);
23 printf('Consumer2 -->\t Avg_load= %.2 f kW \t LF= %.1
      f %% \n',c2_avg_load,c2_lf*100);
24 printf('Consumer3 -->\t Avg_load= %.2 f kW \t LF= %.1
      f %% \n\n',c3_avg_load,c3_lf*100);
25
26 avg_load=c1_avg_load + c2_avg_load + c3_avg_load;
27 lf=avg_load/md_system;
28
29 printf('Combined average load = %.1 f kW \n',avg_load
      );
30 printf('Combined load factor= %.1 f %% \n\n',lf*100);
```

Chapter 2

Inductance and Resistance of Transmission Lines

Scilab code Exa 2.1 self GMD Calculation

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

Scilab code Exa 2.2 Reactance Of ACSR conductors

```
1 //Chapter 2
2 //Example 2.2
3 //page 57
4 //To find reactance of the conductor
5 clear;clc;
6 f=50; //frequency
7 D=5.04; //diameter of the entire ACSR
8 d=1.68; //diameter of each conductor
9 Dsteel=D-2*d; //diameter of steel strand
10 //As shown in fig
11 D12=d;
12 D13=(sqrt(3)*d);
13 D14=2*d;
14 D15=D13;
15 D16=D12;
16 //neglecting the central sttel conductor ,we have the
    6 possibilities
17 D1=(0.7788*d)*D12*D13*D14*D15*D16;
18 //we have total of 6 conductors ,hence
19 D2=D1;
20 D3=D1;
21 D4=D1;
22 D5=D1;
23 D6=D1;
24 Ds=(D1*D2*D3*D4*D5*D6)^(1/(6*6)); //GMR;
25 //since the spacing between lines is lm=100cm
26 l=100;
27 L=0.461*log10(l/Ds); //Inductance of each conductor
28 L1=2*L; // loop inductance
29 Xl=2*pi*f*L1*10^(-3); //reactance of the line
```

```

30 printf("\n\nInductance of each conductor=%0.4 f mH/km
    \n\n",L);
31 printf("Loop Inductance=%0.4 f mH/km\n\n",L1);
32 printf("Loop Reactance=%f ohms/km\n\n",X1);

```

Scilab code Exa 2.3 Inductance Of Composite Conductor Lines

```

1 //Chapter 2
2 //Example 2.3
3 //page 58
4 //To find inductance of each side of the line and
    that of the complete line
5 clear;clc;
6 //to find mutual GMD
7 D14=sqrt(8*8+2*2);
8 D15=sqrt(8*8+6*6);
9 D24=sqrt(8*8+2*2);
10 D25=sqrt(8*8+2*2);
11 D34=sqrt(8*8+6*6);
12 D35=sqrt(8*8+2*2);
13 //sixth root of six mutual distances
14 Dm=(D14*D15*D24*D25*D34*D35)^(1/6); //mutual GMD
    between lines
15
16 //to find GMR of Side A conductors
17 D11=0.7788*2.5*10^(-3);
18 D22=D11;
19 D33=D11;
20 D12=4;
21 D21=D12;
22 D13=8;
23 D31=8;
24 D23=4;

```

```

25 D32=D23;
26 //ninth root nine distances in Side A
27 Da=(D11*D12*D13*D21*D22*D23*D31*D32*D33)^(1/9);
28
29 //to find GMR of Side A conductors
30 D44=0.7788*5*10^(-3);
31 D45=4;
32 D54=D45;
33 D55=D44;
34 //fourth root of four distances in Side B
35 Db=(D44*D45*D54*D55)^(1/4);
36
37 La=0.461*log10(Dm/Da); //inductance line A
38 Lb=0.461*log10(Dm/Db); //inductance line B
39
40 L=La+Lb; //loop inductance
41
42 printf("\n\nMutual GMD between lines = %0.4f m\n\n",
        Dm);
43 printf("GMR of Side A conductors = %0.4f m\n\n",Da);
44 printf("GMR of Side B conductors = %0.4f m\n\n",Db);
45 printf("Inductance of line A = %0.4f mH/km\n\n",La);
46 printf("Inductance of line B = %0.4f mH/km\n\n",Lb);
47 printf("Loop Inductance of the lines = %0.4f mH/km\n
        \n",L);

```

Scilab code Exa 2.5 VoltageDrop and FluxLinkage Calculations

```

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

```

```

5 //To find voltage drop in each of three-phase wires
6
7 clear;clc;
8 Ia=-30+%i*50;
9 Ib=-25+%i*55;
10 Ic=-(Ia+Ib);
11
12 //(a) to find flux linkages with neutral and voltage
    induce in it
13 Dan=4.5;
14 Dbn=3; //from figure
15 Dcn=1.5;
16 Phi_n=2*10^(-7)*(Ia*log(1/Dan)+Ib*log(1/Dbn)+Ic*log
    (1/Dcn));
17 Vn=%i*2*%pi*50*Phi_n*15000; //voltage induced for 15
    km long TL
18 Vn=abs(Vn) ;
19 printf("\nFlux linkages of the neutral wire = %f Wb-
    T/m\n\n",Phi_n);
20 printf("Voltage induced in the neutral = %d\n\n",Vn)
    ;
21
22 //(b) to find voltage drop in each phase
23 Phi_a=2*10^(-7)*(Ia*log(1/(0.7788*0.005))+Ib*log
    (1/1.5)+Ic*log(1/3));
24 Phi_b=2*10^(-7)*(Ib*log(1/(0.7788*0.005))+Ia*log
    (1/1.5)+Ic*log(1/1.5));
25 Phi_c=2*10^(-7)*(Ic*log(1/(0.7788*0.005))+Ib*log
    (1/1.5)+Ia*log(1/3));
26
27 delta_Va=%i*2*%pi*50*Phi_a*15000; //like we did for
    neutral voltage
28 delta_Vb=%i*2*%pi*50*Phi_b*15000;
29 delta_Vc=%i*2*%pi*50*Phi_c*15000;
30
31 printf("The Voltage drop of phase a(in volts) =");
    disp(delta_Va);
32 printf("\n\nThe Voltage drop of phase b(in volts) ="

```

```

    );disp(delta_Vb);
33 printf("\n\nThe Voltage drop of phase c(in volts) ="
    );disp(delta_Vc);

```

Scilab code Exa 2.6 Mutual Inductance Calculation

```

1 //Chapter 2
2 //Example 2.6
3 //page 65
4 //To find mutual inductance between power line and
   telephone line and voltage induced in telephone
   line
5
6 clear;clc;
7 D1=sqrt(1.1*1.1+2*2); //from figure 2.14
8 D2=sqrt(1.9*1.9+2*2); //from figure 2.14
9 Mpt=0.921*log10(D2/D1); //mutual inductance
10 Vt=abs(%i*2*%pi*50*Mpt*10^(-3)*100); //when 100A is
   flowing in the power lines
11
12 printf("\n\nMutual inductance between power line and
   telephone line = %f mH/km\n\n",Mpt);
13 printf("\n\nVoltage induced in the telephone circuit
   = %.3f V/km\n\n",Vt);

```

Scilab code Exa 2.7 Bundled Conductor Three Phase Line

```

1 //Chapter 2
2 //Example 2.7

```

```

3 //page 69
4 //To find inductive reactance of for the three phase
   bundled conductors
5 clear;clc;
6 r=0.01725; //radius of each conductor
7 //from the figure we can declare the distances
8 d=7;
9 s=0.4;
10 //Mutual GMD between bundles of phases a and b
11 Dab=(d*(d+s)*(d-s)*d)^(1/4);
12 //Mutual GMD between bundles of phases b and c
13 Dbc=Dab ; //by symmetry
14 //Mutual GMD between bundles of phases c and a
15 Dca=(2*d*(2*d+s)*(2*d-s)*2*d)^(1/4);
16 //Equivalent GMD is calculated as
17 Deq=(Dab*Dbc*Dca)^(1/3);
18 //self GMD is given by
19 Ds=(0.7788*1.725*10^(-2)*0.4*0.7788*1.725*10^(-2)
   *0.4)^(1/4);
20 //Inductive reactance per phase is given by
21 Xl=2*%pi*50*10^(-3)*0.461*log10(Deq/Ds); //10^(-3)
   because per km is asked
22 printf("\n\nMutual GMD between bundles of phases a
   and b = %0.3fm\n\n",Dab);
23 printf("Mutual GMD between bundles of phases b and c
   = %0.3fm\n\n",Dbc);
24 printf("Mutual GMD between bundles of phases c and a
   = %0.3fm\n\n",Dca);
25 printf("Equivalent GMD = %0.3fm\n\n",Deq);
26 printf("Self GMD of the bundles = %0.3fm\n\n",Ds);
27 printf("Inductive reactance per phase = %0.3f ohms/
   km\n\n",Xl);
28
29 //now let us compute reactance when center to
   centerr distances are used
30 Deq1=(d*d*2*d)^(1/3);
31 Xl1=2*%pi*50*0.461*10^(-3)*log10(Deq1/Ds);
32 printf("\n When radius of conductors are neglected

```

```

    and only distance between conductors are used , we
    get below results:\n\n");
33 printf("Equivalent mean distance is = %f\n\n",Deq1);
34 printf("Inductive reactance per phase = %0.3f ohms/
    km\n\n",X11);
35
36 //when bundle of conductors are replaced by an
    equivalent single conductor
37 cond_dia=sqrt(2)*1.725*10^(-3); //conductor diameter
    for same cross-sectional area
38 X12=2*pi*50*0.461*10^(-3)*log10(Deq1/cond_dia);
39 printf("\nWhen bundle of conductors are replaced by
    an equivalent single conductor:\n\n");
40 printf("Inductive reactance per phase = %0.3f ohms/
    km\n\n",X12) ;
41 percentage_increase=((X12-X11)/X11)*100;
42 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

```
1 //Chapter 3
2 //Example 3.1
3 //page 87
4 //To calculate the capacitance to neutral of a
   single phase line
5 clear;clc;
6 r=0.328; //radius of the conductors
7 D=300; //distance between the conductors
8 h=750; //height of the conductors
9
10 //calculating capacitance neglecting the presence of
   ground
11 //using Eq (3.6)
12 Cn=(0.0242/(log10(D/r)));
13 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);
14
15 //using Eq (3.7)
```



```

16 Cn=(0.0242)/log10((D/(2*r))+((D^2)/(4*r^2)-1)^0.5);
17 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);
18
19 //Consudering the effect of earth and neglecting the
    non uniformity of the charge
20 Cn=(0.0242)/log10(D/(r*(1+((D^2)/(4*h^2))))^0.5));
21 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

```

1 //Chapter 3
2 //Example 3.2
3 //page 88
4 //To claculate the capacitance to neutral and
    charging current of a three phase transmission
    line
5 clear;clc;
6 d=350; //distance between adjacent lines
7 r=1.05/2; //radius of the conductor
8 v=110e3; //line voltage;
9 f=50;
10
11 Deq=(d*d*2*d)^(1/3); //GMD or equivalent
12
13 Cn=(0.0242/log10(Deq/r));
14
15 Xn=1/(2*pi*f*Cn*10^(-6)); // Cn is in uF hence we

```

```

        add 10^6 while printing
16
17 Ic=(v/sqrt(3))/Xn;
18
19 printf("\nCapacitance to neutral is = %f uF/km\n\n",
        Cn);
20 printf("Capacitive reactance of the line is = %f ohm/
        km to neutral\n\n",Xn);
21 printf("Charging Current = %0.2f A/km\n\n",Ic);

```

Scilab code Exa 3.3 Double circuit three phase transmission line

```

1 //Chapter 3
2 //Example 3.3
3 //page 88
4 //To calculate the capacitance to neutral and
        charging current of a double circuit three phase
        transmission line
5 clear;clc;
6
7 //After deriving the equation for Cn from the
        textbook and starting calculation from Eq 3.36
        onwards
8
9 r=0.865*10^(-2); frequency=50; v=110e3;
10 h=6; d=8; j=8; //Referring to fig given in the
        textbook
11
12 i=((j/2)^2+((d-h)/2)^2)^(1/2);
13 f=(j^2+h^2)^(1/2);
14 g=(7^2+4^2)^(1/2);
15
16

```

```

17 Cn=4*pi*8.85*10^(-12)/(log((((i^2)*(g^2)*j*h)/((r
    ^3)*(f^2*d)))^(1/3)));
18
19 Cn=Cn*1000 ; //Cn is in per m.to convert it to per
    km,we multiply by 1000
20 WCn=2*pi*frequency*Cn;
21
22 Icp=(v/sqrt(3))*WCn;
23
24 Icc=Icp/2;
25
26 printf("\nTotal capacitance to neutral for two
    conductors in parallel = %0.6f uF/km \n\n",Cn
    *10^(6));
27 printf("Charging current/phase = %0.3f A/km \n\n",
    Icp);
28 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

```
1 //Chapter 4
2 //Example 4.1
3 //page 103
4 // to draw the per unit reactance diagram
5 clear;clc;
6 mvab=30; kvb=33; //MVA base and KVA base are
   selected
7
8 gen1_mva=30; gen1_kv=10.5; gen1_x=1.6; //Generator
   No.1 details
9 gen2_mva=15; gen2_kv=6.6; gen2_x=1.2; //Generator
   No.2 details
10 gen3_mva=25; gen3_kv=6.6; gen3_x=0.56; //Generator
   No.3 details
11
12 t1_mva=15; t1_hv=33; t1_lv=11; t1_x=15.2; //
   Transformer T1 details
13 t2_mva=15; t2_hv=33; t2_lv=6.2; t2_x=16; //
   Transformer T1 details
```

```

14
15 t1_x=20.5; //Transmission line recatance
16
17 //Loads are neglected as said in the problem
18
19 t1_pu=(t1_x*mvab)/kvb^2;
20 t1_pu=(t1_x*mvab)/kvb^2;
21 t2_pu=(t2_x*mvab)/kvb^2;
22 gen1_kv_base=t1_lv;
23 gen1_pu=(gen1_x*mvab)/gen1_kv_base^2;
24 gen2_kv_base=t2_lv;
25 gen2_pu=(gen2_x*mvab)/gen2_kv_base^2;
26 gen3_pu=(gen3_x*mvab)/gen2_kv_base^2;
27
28 //diplaying the results on console
29
30 printf('Per unit impedance of the components of the
        given power system are as follows :\n\n');
31
32 printf('Transmission line: %0.3f \n\n',t1_pu);
33
34 printf('Transformer T1: %0.3f \n\n',t1_pu);
35
36 printf('Transformer T2: %0.3f \n\n',t2_pu);
37
38 printf('Generator 1: %0.3f \n\n',gen1_pu);
39
40 printf('Generator 2: %0.3f \n\n',gen2_pu);
41
42 printf('Generator 3: %0.3f \n\n',gen3_pu);

```

Scilab code Exa 4.2 Per Unit Calculation

```

1 //Chapter 4
2 //Example 4.2
3 //page 104
4 // To draw the per unit reactance diagram when pu
   values are specified based on equipment rating
5 clear;clc;
6 mvab=30; kvb=11; //MVA base and KVA base are
   selected in the circuit of generator 1
7
8 gen1_mva=30; gen1_kv=10.5; gen1_x=0.435; //
   Generator No.1 details
9 gen2_mva=15; gen2_kv=6.6; gen2_x=0.413; //Generator
   No.2 details
10 gen3_mva=25; gen3_kv=6.6; gen3_x=0.3214; //
   Generator No.3 details
11
12 t1_mva=15; t1_hv=33; t1_lv=11; t1_x=0.209; //
   Transformer T1 details
13 t2_mva=15; t2_hv=33; t2_lv=6.2; t2_x=0.220; //
   Transformer T1 details
14
15 t1_x=20.5; //Transmission line recatance
16
17 //Loads are neglected as said in the problem
18
19 t1_pu=(t1_x*mvab)/t1_hv^2;
20 t1_pu=t1_x*(mvab/t1_mva);
21 t2_pu=t2_x*(mvab/t2_mva);
22 gen1_pu=gen1_x*(mvab/gen1_mva)*(gen1_kv/kvb)^2;
23 gen2_kv_base=t2_lv;
24 gen2_pu=gen2_x*(mvab/gen2_mva)*(gen2_kv/gen2_kv_base
   )^2;
25 gen3_kv_base=t2_lv;
26 gen3_pu=gen3_x*(mvab/gen3_mva)*(gen3_kv/gen3_kv_base
   )^2;
27
28 //diplaying the results on console
29

```

```

30 printf('Per unit impedance of the components of the
      given power system are as follows :\n\n');
31
32 printf('Transmission line: %0.3f \n\n',t1_pu);
33
34 printf('Transformer T1: %0.3f \n\n',t1_pu);
35
36 printf('Transformer T2: %0.3f \n\n',t2_pu);
37
38 printf('Generator 1: %0.3f \n\n',gen1_pu);
39
40 printf('Generator 2: %0.3f \n\n',gen2_pu);
41
42 printf('Generator 3: %0.3f \n\n',gen3_pu);

```

Scilab code Exa 4.3 Excitation EMF and Reactive Power Calculation

```

1 //Taking Base value MVA and KVA
2 clear;clc;
3 mvab=645; //Base MVA in 3-phase
4 kvb=24; //Base KV,line-to-line
5
6 vl=24/kvb; //Load voltage
7 xs=1.2;
8 xs=(xs*mvab)/kvb^2; // xs converted to its pu
9
10 //since the generator is operating at full load &
      0.9pf
11 pf_angle=acos(0.9);
12 Ia=1*(cos(pf_angle)-%i*sin(pf_angle)); //load
      current
13 //to find excitation emf
14 ef=vl+%i*xs*Ia;

```

```

15 delta=atand(imag(ef)/real(ef)); //positive for
    leading
16 ef=abs(ef)*kvb; //pu to actual unit conversion
17 if(delta>0) then lead_lag='leading';
18 else lead_lag='lagging';
19 end
20 printf('Excitation emf= %0.2f kV at an angle %0.3f (
    %s) \n\n',ef,delta,lead_lag);
21 //to find reactive power drawn by load
22 Q=v1*abs(imag(Ia));
23 Q=Q*mvab; //pu to actual unit conversion
24 printf('Reactive power drawn by laod= %d MVAR',Q);

```

Scilab code Exa 4.4 Power Factor And Load Angle Calculation

```

1 //Taking Base value MVA and KVA
2 clear;clc;
3 global mvab
4 mvab=645; //Base MVA in 3-phase
5 kvb=24; //Base KV,line-to-line
6 vt=24/kvb; //Terminal voltage
7 xs=1.2;
8 xs=(xs*mvab)/kvb^2; // xs converted to its pu
9
10 //since the generator is operating at full load &
    0.9 pf
11 pf_angle=acos(0.9);
12 Ia=1*(cos(pf_angle)-%i*sin(pf_angle)); //load
    current
13 //to find excitation emf
14 ef=vt+%i*xs*Ia;
15 ef=abs(ef);
16 P=1*0.9; //at Full load

```



```

17
18 ////////// writing an inline function //////////////////////////////////
19 function [pf,lead_lag,Q]=excitation_change(P,ef,vt,
      xs)
20 sin_delta=(P*xs)/(ef*vt);
21 delta=asind(sin_delta);
22 ef0=ef*(cosd(delta)+(%i*sind(delta)));
23 Ia=(ef0-vt)/(%i*xs);
24 Ia_mag=abs(Ia);Ia_ang=atand(imag(Ia)/real(Ia)); //
      Magnitude and angle of Ia
25 pf=cosd(abs(Ia_ang));
26 if(Ia_ang>0) then lead_lag='leading';
27     elseif (Ia_ang==0) then lead_lag='unity pf';
28     else lead_lag='lagging';
29     end
30 Q=vt*Ia_mag*sind(abs(Ia_ang));
31 Q=abs(Q)*mvab;
32 endfunction
33 //
      ////////////////////////////////////////////
34
35
36 // First Case when Ef is increased by 20% at same
      real load now
37 ef1=ef*1.2;
38 [pf1,lead_lag1,Q1]=excitation_change(P,ef1,vt,xs);
39 disp("Case (i): When Ef is increased by 20% ");
40 printf('\n\tPower factor pf= %0.2f %s \n',pf1,
      lead_lag1);
41 printf('\tReactive power drawn by the load = %0.1f
      MVAR \n',Q1);
42
43 //Second Case when Ef is decreased by 20% at same
      real load now
44 ef2=ef*0.8;
45 [pf2,lead_lag2,Q2]=excitation_change(P,ef2,vt,xs);
46 disp("Case (ii): When Ef is decreased by 20% ");

```

```
47 printf('\n\tPower factor pf= %0.2f %s \n',pf2,
    lead_lag2);
48 printf('\tReactive power drawn by the load = %0.1f
    MVAR \n',Q2);
49
50 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

```
1 //Chapter 5
2 //Example 5.1
3 //page 132
4 //To find sending-end voltage and voltage regulation
5 clc;clear;
6
7 load1=5000; //kW
8 pf=0.707;
9 Vr=10000; //receiving end voltage
10 R=0.0195*20;
11 X=2*%pi*50*0.63*10^-3*20;
12
13 //to find sending end voltage and voltage regulation
14 I=load1*1000/(Vr*pf);
15 Vs=Vr+I*(R*pf+X*sin(acos(pf)));
16 voltage_regulation=(Vs-Vr)*100/Vr;
17 printf('\n\nReceiving current =I=%d A\n',I);
```

```

18 printf('Sending end voltage =Vs=%d V\n',Vs);
19 printf('Voltage Regulation=%0.2f %%',
    voltage_regulation);
20
21 //to find the value of the capacitor to be connected
    in parallel to the load
22 voltage_regulation_desi=voltage_regulation/2;
23 Vs=(voltage_regulation_desi/100)*Vr+Vr;
24 //by solving the equations (i) and (ii)
25 pf=0.911;
26 Ir=549;
27 Ic=(Ir*(pf-%i*sin(acos(pf))))-(707*(0.707-%i*0.707))
    ;
28 Xc=(Vr/imag(Ic));
29 c=(2*pi*50*Xc)^-1;
30 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);
31
32 //to find efficiency in both the cases
33 //case(i)
34 losses=I*I*R*10^-3;
35 n=(load1/(load1+losses))*100;
36 printf('\n Efficiency in : \nCase(i) \t n=%0.1f%%',n
    );
37 //caase(ii)
38 losses=Ir*Ir*R*10^-3;
39 n=(load1/(load1+losses))*100;
40 printf('\nCase(ii) \t n=%0.1f%%',n);

```

Scilab code Exa 5.2 Voltage at the power station end

```

1 //Chapter 5
2 //Example 5.2
3 //page 134
4 //To find voltage at the bus at the power station
   end
5 clc;clear;
6
7 base_MVA=5;
8 base_kV=33;
9 pf=0.85;
10 cable_impedance=(8+%i*2.5);
11 cable_impedance=cable_impedance*base_MVA/(base_kV^2)
   ;
12
13 transf_imp_star=(0.06+%i*0.36)/3; //equivalent star
   impedance of winding of the transformer
14 Zt=(transf_imp_star*5/(6.6^2))+((0.5+%i*3.75)
   *5/(33^2));
15 total=cable_impedance+2*Zt;
16
17 load_MVA=1;
18 load_voltage=6/6.6;
19 load_current=1/load_voltage;
20
21 Vs=load_voltage+load_current*(real(total)*pf+imag(
   total)*sin(acos(pf)));
22 Vs=Vs*6.6;
23 printf('\\n\\nCable impedance= (%0.3 f+j%0.4 f) pu\\n',
   real(cable_impedance),imag(cable_impedance));
24 printf('\\nEquivalent star impedance of 6.6kV winding
   of the transformer =( %0.2 f+j%0.2 f) pu\\n',real(
   transf_imp_star),imag(transf_imp_star));
25 printf('\\nPer unit transformer impedance, Zt=(%0.4 f+
   j%0.3 f) pu\\n',real(Zt),imag(Zt));
26 printf('\\nTotal series impedance=(%0.3 f+j%0.3 f) pu\\n
   ',real(total),imag(total));
27 printf('\\nSending end Voltage =|Vs|=%0.2 f kV (line-to
   -line)',Vs);

```

Scilab code Exa 5.3 Problem with mixed end condition

```
1 //Chapter 5
2 //Example 5.3
3 //page 135
4 //problem with mixed end condition
5 clc;clear;
6 Vr=3000; //receiving end voltage
7 pfs=0.8; //sending end power factor
8 Ps=2000*103; //sending end active power
9 z=0.4+%i*0.4; //series impedance
10 Ss=Ps/pfs; //sending end VA
11 Qs=Ss*sqrt(1-pfs2); //sending end reactive power
12
13 //by substituting all the values to the equation (
14 deff(' [y]=fx ( I ) ', "y=(Vr2)*(I2)+2*Vr*(I2)*(real(z)
15 *((Ps-real(z)*(I2))/Vr)+imag(z)*((Qs-imag(z)*(I
16 ^2))/Vr))+(abs(z))^2*(I4)-(Ss2)");
17 I=fsolve(100,fx);
18
19 pfR=(Ps-real(z)*(I2))/(Vr*I); //Cos(phi_r)
20 Pr=Vr*I*pfR;
21 Vs=(Ps/(I*pfs));
22
23 printf(' \nLoad Current |I|= %0.2 f A',I);
24 printf(' \nLoad Pr=%d W',Pr);
25 printf(' \nReceiving end power factor=%0.2 f',pfR);
26 printf(' \nSupply Voltage=%0.2 fV',Vs);
```

Scilab code Exa 5.4 Medium Transmission line system

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

Scilab code Exa 5.5 Maximum permissible length and and Frequency

```
1 //Chapter 5
2 //Example 5.5
3 //page 147
4 //to find maximum permissible length and and
   frequency
5 clc; clear;
6 R=0.125*400;
7 X=0.4*400;
8 Y=2.8*(10-6)*400*%i;
9 Z=R+X*%i;
10
11 //(i) At no-load
12 A=1+(Y*Z/2);
13 C=Y*(1+Y*Z/6);
14 VR_line=220000/abs(A);
15 Is=abs(C)*VR_line/sqrt(3);
16 printf('\\n\\n |VR|line = %d kV',VR_line/1000);
17 printf('\\n |Is| = %d A',Is);
18
19 //(ii) to find maximum permissible length
20 //By solving the equations shown in the book,we get
21 l=sqrt((1-0.936)/(0.56*10-(6)));
22 printf('\\n\\n Maximum permissible length of the line
   = %d km',l);
23
24 //(iii) to find maximum permissible frequency for
   the case(i)
25 //By solving the equations shown in the book,we get
26 f=sqrt((1-0.88)*50*50)/(0.5*1.12*10-3*160));
27 printf('\\n\\n Maximum permissible frequency = %0.1f
```


Hz\n\n',f);

Scilab code Exa 5.6 Incident and Reflected voltages

```
1 //Chapter 5
2 //Example 5.6
3 //page 149
4 //to find incident and reflected voltages
5 clear;clc;
6
7 R=0.125;
8 X=0.4;
9 y=%i*2.8*10^(-6);
10 z=R+%i*X;
11
12 r=sqrt(y*z); //propogation constant
13 a=real(r); //attenuation constant
14 b=imag(r); //phase constant
15
16 //(a) At the receiving-end;
17 Vr=220000;
18 Inci_vol=Vr/(sqrt(3)*2);
19 Refl_vol=Vr/(sqrt(3)*2);
20 printf('\n\nIncident Vvoltage=%0.2f kV',Inci_vol
        /1000);
21 printf('\n\nReflected Vvoltage=%0.2f kV',Refl_vol
        /1000);
22
23 //(b) At 200km from the receiving-end
24 x=200;
25 Inci_vol=Inci_vol*exp(a*x)*exp(%i*b*x);
26 Refl_vol=Refl_vol*exp(-a*x)*exp(-%i*b*x);
27 printf('\n\nIncident voltage=%0.2f @ %0.1f deg kV',
```

```

    abs(Inci_vol)/1000, atand(imag(Inci_vol)/real(
    Inci_vol)));
28 printf('\nReflected voltage=%0.2f @ %0.1f deg kV',
    abs(Refl_vol)/1000, atand(imag(Refl_vol)/real(
    Refl_vol)));
29
30 //(c) Resultant voltage at 200km from the receiving-
    end
31 res=Inci_vol+Refl_vol;
32 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

```

1 //Chapter 5
2 //Example 5.7
3 //page 138
4 //to tabulate characteristics of a system using
    different methods
5 clear;clc;
6
7 Z=40+125*i;
8 Y=i*10^(-3);
9 Ir=((50*10^6)/(220000*0.8*sqrt(3)))*(cosd(-36.9)+i*
    sind(-36.9));
10 Vr=220000/sqrt(3);
11
12 //(a) Short line approximation
13 Vs=Vr+Ir*Z;
14 Vs_line1=Vs*sqrt(3);
15 Is1=Ir;
16 pfs1=cos(atan(imag(Vs)/real(Vs))+acos(0.8));
17 Ps1=sqrt(3)*abs(Vs_line1)*abs(Is1)*pfs1;

```

```

18
19 //(b) Nominal pi method
20 A=1+Y*Z/2;
21 D=A;
22 B=Z;
23 C=Y*(1+Y*Z/4);
24 Vs=A*Vr+B*Ir;
25 Is2=C*Vr+D*Ir;
26 Vs_line2=sqrt(3)*Vs;
27 pfs2=cos(atan(imag(Is2)/real(Is2))-atan(imag(Vs)/
    real(Vs)));
28 Ps2=sqrt(3)*abs(Vs_line2)*abs(Is2)*pfs2;
29
30 //(c) Exact transmission line equations
31 rl=sqrt(Z*Y); //propogation constant
32 Zc=sqrt(Z/Y); //characteristic impedance
33 A=cosh(rl);
34 B=Zc*sinh(rl);
35 C=sinh(rl)/Zc;
36 D=cosh(rl);
37 Vs=A*Vr+B*Ir;
38 Is3=C*Vr+D*Ir;
39 Vs_line3=sqrt(3)*Vs;
40 pfs3=cos(atan(imag(Is3)/real(Is3))-atan(imag(Vs)/
    real(Vs)));
41 Ps3=sqrt(3)*abs(Vs_line3)*abs(Is3)*pfs3;
42
43 //(d) Approximation
44 A=(1+Y*Z/2);
45 B=Z*(1+Y*Z/6);
46 C=Y*(1+Y*Z/6);
47 D=A;
48 Vs=A*Vr+B*Ir;
49 Is4=C*Vr+D*Ir;
50 Vs_line4=sqrt(3)*Vs;
51 pfs4=cos(atan(imag(Is4)/real(Is4))-atan(imag(Vs)/
    real(Vs)));
52 Ps4=sqrt(3)*abs(Vs_line4)*abs(Is4)*pfs4;

```

```

53
54 //converting all the values to their standard form
    before writing it to table
55
56 //voltage to kV
57 Vs_line1=abs(Vs_line1)/1000;
58 Vs_line2=abs(Vs_line2)/1000;
59 Vs_line3=abs(Vs_line3)/1000;
60 Vs_line4=abs(Vs_line4)/1000;
61
62 //Current to kA
63 Is1=Is1/1000;
64 Is2=Is2/1000;
65 Is3=Is3/1000;
66 Is4=Is4/1000;
67
68 //power to MW
69 Ps1=Ps1/1000000;
70 Ps2=Ps2/1000000;
71 Ps3=Ps3/1000000;
72 Ps4=Ps4/1000000;
73
74 //preparing table
75 printf("\n\
    n-----
    ");
76 printf('\n          \t\tShort line \t\t          Nominal
    Pi \t\t Exact          \t\t Approximation');
77 printf("\
    n-----
    ");
78 printf('\n|Vs|line\t\t%0.2fkV          \t\t %0.2fkV\t\t\
    t %0.2fkV \t\t %0.2fkV',Vs_line1,Vs_line2,
    Vs_line3,Vs_line4);
79 printf('\nIs          \t\t%0.3f@%0.1fdeg kA \t\t%0.2f@%0
    .1fdeg kA\t\t%0.4f@%0.1fdeg kA\t\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)));
80 printf('\n pfs      \t\t%0.3f lagging      \t\t%0.3f
    leading \t\t%0.3f leading \t\t%0.3f leading ',pfs1
    ,pfs2,pfs3,pfs4);
81 printf('\n Ps      \t\t%0.2f MW      \t\t%0.2f MW
    \t\t%0.2f MW \t\t%0.2f MW',Ps1,Ps2,Ps3,Ps4);
82 printf("\n
    n -----
    \n\n\n");

```

Scilab code Exa 5.8 Torque angle and Station powerfactor

```

1 //Chapter 5
2 //Example 5.8
3 //page 162
4 //to estimate the torque angle and station
    powerfactor
5 clear;clc;
6 Sd1=15+%i*5;
7 Sd2=25+%i*15;
8 //case(a) cable impedance=j0.05pu
9 r=0;
10 x=%i*0.05;
11 PG1=20;
12 PG2=20;
13 Ps=5;Pr=5;
14 V1=1;
15 V2=1;
16 d1=asind(Ps*abs(x)/(V1*V2)); //delta1
17 V1=V1*(cosd(d1)+%i*sind(d1));
18 Qs=((abs(V1)^2)/abs(x))-((abs(V1)*abs(V2))*cosd(d1)
    /(abs(x)));
19 Qr=((abs(V1)*abs(V2))*cosd(d1)/(abs(x)))-(abs(V1)

```

```

    ^2)/abs(x));
20 Q1=Qs-Qr;
21 Ss=Ps+%i*Qs;
22 Sr=Pr+%i*Qr;
23 Sg1=Sd1+Ss;
24 Sg2=Sd2-Sr;
25 pf1=cos(atan(imag(Sg1)/real(Sg1)));
26 pf2=cos(atan(imag(Sg2)/real(Sg2)));
27 printf('\n\nCase(a)\nTotal load on station1=%d+j%0.3
    f pu',real(Sg1),imag(Sg1));
28 printf('\nPower factor of station1=%0.3f pu lagging'
    ,pf1);
29 printf('\n\nTotal load on station2=%d+j%0.3f pu',real
    (Sg2),imag(Sg2));
30 printf('\nPower factor of station2=%0.3f pu lagging'
    ,pf2);
31 //case(b) cable impedance=0.005+j0.05;
32 r=0.005;
33 PG1=20;
34 V1=1;V2=1;
35 Ps=5;
36 //from the eq(i) in the textbook,we can calculate d1
37 z=r+x;
38 theta=atand(imag(z)/real(z));
39 z=abs(z);
40 d1=acosd(z*(V1^2*cosd(theta)/z-Ps)/(V1*V2))-theta;
41 Qs=(V1^2*sind(theta)/z)-(V1*V2*sind(theta+d1)/z);
42 Qg1=5+Qs;
43 Pr=(V1*V2*cosd(theta-d1)/z)-(V1^2*cosd(theta)/z);
44 Pg2=25-Pr;
45 Qr=(V1*V2*sind(theta-d1)/z)-(V1^2*sind(theta)/z);
46 Qg2=15-Qr;
47 Ss=Ps+%i*Qs;
48 Sr=Pr+%i*Qr;
49 Sg1=Sd1+Ss;
50 Sg2=Sd2-Sr;
51 pf1=cos(atan(imag(Sg1)/real(Sg1)));
52 pf2=cos(atan(imag(Sg2)/real(Sg2)));

```

```

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

```

Scilab code Exa 5.9 Power Voltage and Compensating equipment rating

```

1 //Chapter 5
2 //Example 5.9
3 //page 165
4 //to determine power,voltage,compensating equipment
    rating
5 clear;clc;
6 A=0.85;
7 B=200;
8
9 //case(a)
10 Vs=275000;
11 Vr=275000;
12 a=5;b=75; //alpha and beta
13 Qr=0;
14 //from equation 5.62
15 d=b-asind((B/(Vs*Vr))*(Qr+(A*Vr^2*sind(b-a)/B))); //
    delta
16 Pr=(Vs*Vr*cosd(b-d)/B)-(A*Vr^2*cosd(b-a)/B);
17 printf('\n\nncase(a)\nPower at unity powerfactor that
    can be received =%0.1f MW',Pr/10^6);
18
19 //case(b)

```

```

20 Pr=150*10^6;
21 d=b-acosd((B/(Vs*Vr))*(Pr+(A*Vr^2*cosd(b-a)/B))); //
    delta
22 Qr=(Vs*Vr*sind(b-d)/B)-(A*Vr^2*sind(b-a)/B);
23 Qc=-Qr;
24 printf('\n\ncase(b)\nRating of the compensating
    equipment = %0.2f MVAR',Qc/10^6);
25 printf('\n\ni.e the compensating equipment must feed
    positive VARs into the line');
26
27
28 //case(c)
29 Pr=150*10^6;
30 Vs=275000;
31 //by solving the two conditions given as (i) and (ii
    ), we get
32 Vr=244.9*10^3;
33 printf('\n\ncase(c)\nReceiving end voltage = %0.1f
    kV',Vr/1000);

```

Scilab code Exa 5.10 MVA rating of the shunt reactor

```

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

```



```

12 A=1+(Y*Z/2);
13 B=Z;
14 Vs=275;
15 Vr=275;
16 r=(Vs*Vr)/abs(B);
17 Ce=abs(A/B)*Vr^2;
18 printf('Radius of the receiving-end circle=%0.1f MVA
    \n\n',r);
19 printf('Location of the center of receiving-end
    circle= %0.1f MVA\n\n',Ce);
20 printf('From the graph, 55 MVA shunt reactor is
    required\n\n');
21 theta=180+82.5;
22 x=-75:0.01:450;
23 a=Ce*cosd(theta); //to draw the circle
24 b=Ce*sind(theta);
25 y=sqrt(r^2-(x-a)^2)+b;
26 x1=a:0.001:0;
27 y1=tand(theta)*x1;
28 plot(x,y,x1,y1);
29 title('Circle diagram for example 5.10');
30 xlabel('MW');
31 ylabel('MVAR');
32 plot(a,b,'markersize',150);
33 xgrid(2)
34 set(gca(),'grid',[0,0])
35 get("current_axes");
36 xstring(-75,25,'55 MVAR');
37 xstring(-75,-25,'83.5 deg');
38 xstring(-20,-300,'487.6 MVA');
39 xstring(300,-100,'544.3 MVA');

```

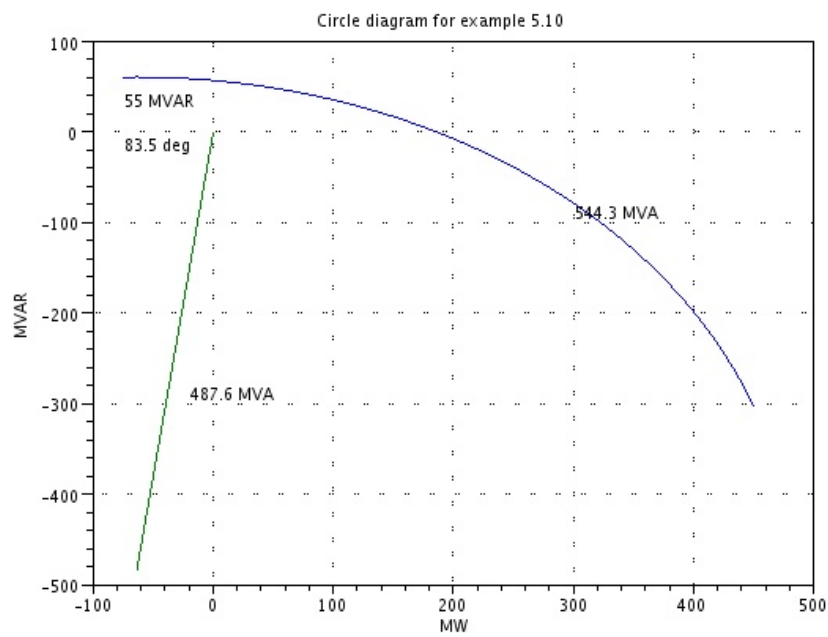


Figure 5.1: MVA rating of the shunt reactor

Scilab code Exa 5.11 SendingEnd voltage and maximum power delivered

```
1 //Chapter 5
2 //Example 5.11
3 //page 172
4 //To determine sending-end voltage.maximum power
   delivered
5 clear;clc;
6
7 A=0.93*(cosd(1.5)+%i*sind(1.5));
8 B=115*(cosd(77)+%i*sind(77));
9 Vr=275;
10 Ce=abs(A/B)*Vr^2;
11 printf('Centre of the receiving end circle is = %0.1
   f MVA\n\n',Ce);
12 CrP=850;Vs=CrP*abs(B)/Vr;
13 printf('(a) From the diagram,\n\tCrP=%d \n \tSending
   end voltage |Vs|= %0.1f kV\n\n',CrP,Vs);
14 Vs=295; //given
15 r=(Vs*Vr)/abs(B);
16 Pr_m=556; //from the diagram
17 printf('(b) Radius of the circle diagram = %0.1f MVA
   \n\t PR_max=%d MW\n\n',r,Pr_m);
18 Ps=295; //from the diagram;
19 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

```
1 //Chapter 6
2 //Example 6.1
3 //page 195
4 //To Ybus using singular transformation
5
6 clear;clc;
7 printf('Let us solve this problem by giving values
      given in the table 6.1 instead of keeping it in
      variables ');
8
9 y10=1;y20=1;y30=1;y40=1;
10 y34=2-%i*6;y23=0.666-%i*2;
11 y12=2-%i*6;y24=1-%i*3;
12 y13=1-%i*3;
13
14 Y=[y10 0 0 0 0 0 0 0 0;
15     0 y20 0 0 0 0 0 0 0;
16     0 0 y30 0 0 0 0 0 0;
17     0 0 0 y40 0 0 0 0 0;
18     0 0 0 0 y34 0 0 0 0;
19     0 0 0 0 0 y23 0 0 0;
```

```

20     0 0 0 0 0 0 0 y12 0 0;
21     0 0 0 0 0 0 0 0 y24 0;
22     0 0 0 0 0 0 0 0 0 y13];
23 A=[1 0 0 0;
24     0 1 0 0;
25     0 0 1 0;
26     0 0 0 1;
27     0 0 1 -1;
28     0 -1 1 0;
29     1 -1 0 0;
30     0 -1 0 1;
31     -1 0 1 0];
32 printf('\n\n Ybus matrix using singular
      transformation for the system of fig.6.2 is \n
      Ybus= ');
33 Y=A'*Y*A;
34 disp(Y);
35 // for verification let us calculate as given in the
      text book
36 printf('\n\n For verification ,calculating Ybus
      substituting as given in the text book\n Ybus(
      verifiaction)=');
37 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)];
38 disp(Yveri);

```

Scilab code Exa 6.2 Ybus of a sample system

```

1 //Chapter 6
2 //Example 6.2
3 //page 195
4 //To Ybus of sample system

```

```

5  clear;clc;
6
7  y10=1;y20=1;y30=1;y40=1;
8  y34=2-%i*6;y23=0.666-%i*2;
9  y12=2-%i*6;y24=1-%i*3;
10 y13=1-%i*3;
11
12 //to form Ybus matrix
13 Y11=y13;Y12=0;Y13=-y13;Y14=0;
14 Y21=0;Y22=y23+y24;Y23=-y23;Y24=-y24;
15 Y31=-y13;Y32=-y23;Y33=y13+y23+y34;Y34=-y34;
16 Y41=0;Y42=-y24;Y43=-y34;Y44=y34+y24;
17
18 //case(i) line shown dotted is not connected
19 Ybus=[Y11 Y12 Y13 Y14;
20       Y21 Y22 Y23 Y24;
21       Y31 Y32 Y33 Y34;
22       Y41 Y42 Y43 Y44];
23 printf('(i) Assuming that the line shown is not
        connected \n Ybus= ');disp(Ybus);
24 //case(ii) line shown dotted is connected
25 Y12=Y12-y12;Y21=Y12;
26 Y11=Y11+y12;
27 Y22=Y22+y12;
28
29 Ybus=[Y11 Y12 Y13 Y14;
30       Y21 Y22 Y23 Y24;
31       Y31 Y32 Y33 Y34;
32       Y41 Y42 Y43 Y44];
33 printf('\n\n(ii) Assuming that the line shown is
        connected \n Ybus= ');disp(Ybus);

```

Scilab code Exa 6.3 Approximate load flow solution

```

1 //Chapter 6
2 //Example 6.3
3 //page 201
4 //To find an approximate load flow solution
5 clear;clc;
6
7 //
8 //Realdemand      Reactive demand      Real generation
9 //      Reactive generation      Bus
10 //
11 //
12 //
13 //
14
15 Pd1=1;           Qd1=0.5;           Pg1=0;
16                Qg1=0; //initialization 1
17 Pd2=1;           Qd2=0.4;           Pg2=4;
18                Qg2=0; //initialization 2
19 Pd3=2;           Qd3=1;           Pg3=0;
20                Qg3=0; //initialization 3
21 Pd4=2;           Qd4=1;           Pg4=0;
22                Qg4=0; //initialization 4
23
24 Pg1=Pd1+Pd2+Pd3+Pd4-Pg2;
25
26 //Ybus matrix from the network
27 Ybus=[-21.667*%i 5*%i 6.667*%i 10*%i;
28        5*%i -21.667*%i 10*%i 6.667*%i;
29        6.667*%i 10*%i -16.667*%i 0;
30        10*%i 6.667*%i 0 -16.667*%i];
31 printf('Ybus matrix of the system is given by \nYbus
32 ='); disp(Ybus);
33 //as given in the text book using approximate load
34 //flow equations and simplifying (ii),(iii),(iv)
35 //delta matrix(x) is of the from A*x=B
36 A=[-5 21.667 -10 -6.667;
37    -6.667 -10 16.667 0;
38    -10 -6.667 0 16.667

```

```

28     1 0 0 0];
29
30 B=[3; -2; -2;0];
31
32 delta=inv(A)*B; //solving for delta
33 printf('\nDelta of the system is given by \ndelta(
    rad)=');disp(delta);
34
35 Q1=-5*cos(delta(2,1))-6.667*cos(delta(3,1))-10*cos(
    delta(4,1))+21.667;
36 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;
37 Q3=-6.667*cos(delta(3,1))-10*cos(delta(3,1)-delta
    (2,1))+16.667;
38 Q4=-10*cos(delta(4,1))-6.667*cos(delta(4,1)-delta
    (2,1))+16.667;
39
40 Q=[Q1;Q2;Q3;Q4];
41 printf('\nInjected reactive power at the buses is
    given by \nQi(in pu)=');disp(Q);
42
43 Qg1=Q1+Qd1;
44 Qg2=Q2+Qd2;
45 Qg3=Q3+Qd3;
46 Qg4=Q4+Qd4;
47
48 Qg=[Qg1;Qg2;Qg3;Qg4];
49 printf('\n Reactive power generation at the four
    buses are \nQgi(in pu)=');disp(Qg);
50 Qd=[Qd1;Qd2;Qd3;Qd4];
51 Q1=sum(Qg)-sum(Qd);
52 printf('\nReactive power losses are QL=%0.5f pu',Q1)
    ;
53
54 printf('\n\nLine Flows are given as:\n');
55 P13=(abs(Ybus(1,3)))*sin(delta(1,1)-delta(3,1));P31
    ==-P13;printf('\nP13=-P31=%0.3f pu',P13);
56 P12=(abs(Ybus(1,2)))*sin(delta(1,1)-delta(2,1));P21

```



```

    =-P12;printf('\nP12=-P21=%0.3 f pu',P12);
57 P14=(abs(Ybus(1,4)))*sin(delta(1,1)-delta(4,1));P41
    =-P14;printf('\nP14=-P41=%0.3 f pu',P14);
58
59 Q13=abs(Ybus(1,3))-(abs(Ybus(1,3)))*cos(delta(1,1)-
    delta(3,1));Q31=-Q13;printf('\n\nQ13=-Q31=%0.3 f
    pu',Q13);
60 Q12=abs(Ybus(1,2))-(abs(Ybus(1,2)))*cos(delta(1,1)-
    delta(2,1));Q21=-Q12;printf('\nQ12=-Q21=%0.3 f pu',
    Q12);
61 Q14=abs(Ybus(1,4))-(abs(Ybus(1,4)))*cos(delta(1,1)-
    delta(4,1));Q41=-Q14;printf('\nQ14=-Q41=%0.3 f pu',
    Q14);

```

Scilab code Exa 6.4 Bus voltages using GS iterations

```

1 //Chapter 6
2 //Example 6.4
3 //page 209
4 //To find bus voltages using GS iterations
5 clear;clc;
6
7 //Ybus matrix from the network
8 Ybus=[3-9%i -2+6%i -1+3%i 0;
9       -2+6%i 3.666-11%i -0.666+2%i -1+3%i
10      -1+3%i -0.666+2%i 3.666-11%i -2+6%i
11      0 -1+3%i -2+6%i 3-9%i]
12
13 //
14 //Pi      Qi      Vi      Remarks      Bus no
    //

```

```

15 P1=0;      Q1=0;      V1=1.04;      //Slack bus      1
16 P2=0.5;   Q2=-0.2;   V2=1;      //PQbus          2
17 P3=-1.0;  Q3=0.5;   V3=1;      //PQbus          3
18 P4=0.3;   Q4=-0.1;   V4=1;      //PQbus          4
19 //
      //////////////////////////////////////
20
21 n=1;
22 for i=1:n
23     V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-Ybus
        (2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
24     V3=(1/Ybus(3,3))*(((P3-%i*Q3)/conj(V3))-Ybus
        (3,1)*V1-Ybus(3,2)*V2-Ybus(3,4)*V4);
25     V4=(1/Ybus(4,4))*(((P4-%i*Q4)/conj(V4))-Ybus
        (4,1)*V1-Ybus(4,2)*V2-Ybus(4,3)*V3);
26 end
27
28 printf('\nAt the end of iteration %d the voltages at
        the buses are:\n\nV1=',n);disp(V1);printf('pu');
29 printf('\n\nV2=');disp(V2);printf('pu');
30 printf('\n\nV3=');disp(V3);printf('pu');
31 printf('\n\nV4=');disp(V4);printf('pu');

```

Scilab code Exa 6.5 Reactive power injected using GS iterations

```

1 //Chapter 6
2 //Example 6.5
3 //page 210
4 //To find bus voltages and Reactive power injected
  using GS iterations
5 clear;clc;
6

```

```

7 //Ybus matrix from the network
8 Ybus=[3-9*i -2+6*i -1+3*i 0;
9       -2+6*i 3.666-11*i -0.666+2*i -1+3*i
10      -1+3*i -0.666+2*i 3.666-11*i -2+6*i
11      0 -1+3*i -2+6*i 3-9*i]
12
13 //Case(i)
14
15 //
16 //Pi      Qi      Vi      Remarks      Bus no
17 //
18 P1=0;      Q1=0;      V1=1.04;    //Slack bus  1
19 P2=0.5;    Q2=0.2;    V2=1.04;    //PVbus      2
20 P3=-1.0;   Q3=0.5;    V3=1;        //PQbus      3
21 P4=0.3;    Q4=-0.1;   V4=1;        //PQbus      4
22 //
23 //////////////////////////////////////////////////
24
25 printf('\\nCase(i) When 0.2<Q2<1 pu and running for 1
26 iteration ,we get \\n\\n');
27 Q2min=0.2; Q2max=1;
28 n=1;
29
30 for i=1:n
31     if Q2<Q2min then
32         Q2=Q2min;
33         V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-Ybus
34         (2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
35     elseif Q2>Q2max then
36         Q2=Q2max;
37         V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-
38         Ybus(2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
39     else
40         Q2=-imag(conj(V2)*Ybus(2,1)*V1+conj(V2)*
41         Ybus(2,2)*V2+Ybus(2,3)*V3+Ybus(2,4)*V4)
42         );

```

```

35         [mag,delta2]=polar((1/Ybus(2,2))*(((P2-%i*
           Q2)/(conj(V2)))-Ybus(2,1)*V1-Ybus(2,3)*
           V3-Ybus(2,4)*V4));
36         V2=abs(V2)*(cos(delta2)+%i*sin(delta2));
37     end
38     V3=(1/Ybus(3,3))*(((P3-%i*Q3)/conj(V3))-Ybus
           (3,1)*V1-Ybus(3,2)*V2-Ybus(3,4)*V4);
39     V4=(1/Ybus(4,4))*(((P4-%i*Q4)/conj(V4))-Ybus
           (4,1)*V1-Ybus(4,2)*V2-Ybus(4,3)*V3);
40 end
41
42 printf('Q2=');disp(Q2);printf('pu');
43 printf('\n\n\ndelta2=');disp(abs(delta2));printf('
rad');
44 printf('\n\n\nV1=');disp(V1);printf('pu');
45 printf('\n\n\nV2=');disp(V2);printf('pu');
46 printf('\n\n\nV3=');disp(V3);printf('pu');
47 printf('\n\n\nV4=');disp(V4);printf('pu');
48
49
50 // case(ii)
51
52 printf('\n\n\nCase(ii) When 0.25<Q2<1 pu and running
for 1 iteration ,we get \n\n');
53
54 //
           ////////////////////////////////////////////////////////////////////
55 //Pi      Qi      Vi      Remarks      Bus no
           //
56 P1=0;      Q1=0;      V1=1.04;    //Slack bus    1
57 P2=0.5;      Q2=0;      V2=1.04;    //PVbus        2
58 P3=-1.0;    Q3=0.5;    V3=1;       //PQbus        3
59 P4=0.3;      Q4=-0.1;   V4=1;       //PQbus        4
60 //
           ////////////////////////////////////////////////////////////////////
61

```

```

62 Q2min=0.25;Q2max=1;
63 n=1;
64
65 for i=1:n
66     if Q2<Q2min then
67         Q2=Q2min;
68         V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-Ybus
           (2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
69     elseif Q2>Q2max then
70         Q2=Q2max;
71         V2=(1/Ybus(2,2))*(((P2-%i*Q2)/conj(V2))-
           Ybus(2,1)*V1-Ybus(2,3)*V3-Ybus(2,4)*V4);
72     else
73         Q2=-imag(conj(V2)*Ybus(2,1)*V1+conj(V2)*(
           Ybus(2,2)*V2+Ybus(2,3)*V3+Ybus(2,4)*V4
           ));
74         [mag,delta2]=polar((1/Ybus(2,2))*(((P2-%i*
           Q2)/(conj(V2)))-Ybus(2,1)*V1-Ybus(2,3)*
           V3-Ybus(2,4)*V4));
75         V2=abs(V2)*(cos(delta2)+%i*sin(delta2));
76     end
77     V3=(1/Ybus(3,3))*(((P3-%i*Q3)/conj(V3))-Ybus
           (3,1)*V1-Ybus(3,2)*V2-Ybus(3,4)*V4);
78     V4=(1/Ybus(4,4))*(((P4-%i*Q4)/conj(V4))-Ybus
           (4,1)*V1-Ybus(4,2)*V2-Ybus(4,3)*V3);
79 end
80
81 printf('Q2=');disp(Q2);printf('pu');
82 printf('\n\n\nV1=');disp(V1);printf('pu');
83 printf('\n\n\nV2=');disp(V2);printf('pu');
84 printf('\n\n\nV3=');disp(V3);printf('pu');
85 printf('\n\n\nV4=');disp(V4);printf('pu');

```

Scilab code Exa 6.6 Load flow solution using the NR method

```

1 //Chapter 6
2 //Example 6.6
3 //page 218
4 //To find load flow solution using the NR method
5 clear;clc;
6
7 //
8 //Pd      Qd      Pg      Qg      V
9 //      Bus  Type/////
10 Pd1=2.0;   Qd1=1.0;   Pg1=0;   Qg1=0;   V1=1.04;
11 //1      slack bus
12 Pd2=0;     Qd2=0;     Pg2=0.5; Qg2=1;   V2=1;
13 //2      PQ bus
14 Pd3=1.5;   Qd3=0.6;   Pg3=0.0; Qg3=0;   V3=1.04;
15 //3      PV bus
16
17 //
18 ///////////////////////////////////////////////////
19
20 [V1_mag,V1_ang]=polar(V1);
21 [V2_mag,V2_ang]=polar(V2);
22 [V3_mag,V3_ang]=polar(V3);
23 y_series=1/(0.02+%i*0.08);
24 y_self=2*y_series;
25 y_off=-1*y_series;
26 Ybus=[y_self y_off y_off;y_off y_self y_off;y_off
27        y_off y_self];
28
29 [y_bus_mag_21,y_bus_ang_21]=polar(Ybus(2,1));
30 [y_bus_mag_22,y_bus_ang_22]=polar(Ybus(2,2));
31 [y_bus_mag_23,y_bus_ang_23]=polar(Ybus(2,3));
32 [y_bus_mag_31,y_bus_ang_31]=polar(Ybus(3,1));

```

```

26 [y_bus_mag_32 , y_bus_ang_32]=polar(Ybus(3,2));
27 [y_bus_mag_33 , y_bus_ang_33]=polar(Ybus(3,3));
28 [y_bus_mag_11 , y_bus_ang_11]=polar(Ybus(1,1));
29
30 //direct computer solution has been found as below
    by running for 3 iterations
31
32 n=3;
33 for i=1:n
34 //from eq.6.27 and 6.28
35 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);
36
37 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);
38
39 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);
40
41 P2=real(P2);
42 P3=real(P3);
43 Q2=real(Q2);
44
45 delta_P2=(Pg2-Pd2)-(P2);
46 delta_P3=(Pg3-Pd3)-(P3);
47 delta_P2=(Pg2-Pd2)-(P2);
48 delta_Q2=(Qg2-Qd2)-(Q2);
49
50 //forming jacobian matrix by differentiating
    expressions of P2,P3,Q2
51 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);
52 j12=-V2_mag*V3_mag*y_bus_mag_23*sin(y_bus_ang_23+
    V3_ang-V2_ang);
53 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);
54
55 j21=-V2_mag*V3_mag*y_bus_mag_32*sin(y_bus_ang_32+
    V2_ang-V3_ang);
56 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);
57 j23=V3_mag*y_bus_mag_32*cos(y_bus_ang_32+V2_ang-
    V3_ang);
58
59 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);
60 j32=-V2_mag*V3_mag*y_bus_mag_23*cos(y_bus_ang_23+
    V3_ang-V2_ang);
61 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);
62
63 J=[j11 j12 j13;j21 j22 j23;j31 j32 j33];
64 J=real(J);
65
66 //power residuals
67 PR=[delta_P2;delta_P3;delta_Q2];
68
69 //changes in variables
70 ch_var=inv(J)*PR;
71
72 V2_ang=V2_ang+ch_var(1,1);
73 V3_ang=V3_ang+ch_var(2,1);
74 V2_mag=V2_mag+ch_var(3,1);

```



```

75
76 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);
77 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);
78
79 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);
80 Qg3=Q3+Qd3;
81
82 end
83
84 S1=real(P1)+%i*real(Q1);
85 S2=P2+%i*Q2;
86 S3=P3+%i*Q3;
87
88 printf('\n\nThe final results are given below:\n\n');
89 printf('V2=%0.3f @ %0.3f rad\n',V2_mag,V2_ang);
90 printf('V3=%0.3f @ %0.3f rad\n',V3_mag,V3_ang);
91 printf('Qg3=%0.2f pu(with in limits)\n',Qg3);
92 printf('\n\nS1=');disp(S1);printf('pu');
93 printf('\n\nS2=');disp(S2);printf("pu");
94 printf('\n\nS3=');disp(S3);printf("pu");
95 printf('\n\nTransmission losses=%0.3f pu\n',(real(P1
    )+P2+P3));
96
97 //Line Flows
98
99 //V_mag=[V1_mag V2_mag V3_mag];
100 //V_ang=[V1_ang V2_ang V3_ang];
101 v1=V1_mag*(cos(V1_ang)+%i*sin(V1_ang));
102 v2=V2_mag*(cos(V2_ang)+%i*sin(V2_ang));

```

```

103 v3=V3_mag*(cos(V3_ang)+%i*sin(V3_ang));
104 V=[v1 v2 v3];
105 for i=1:3
106     for j=1:3
107         s(i,j)=conj(V(i))*(V(i)-V(j))*(2.941-%i
            *11.764)+conj(V(i))*V(i)*(%i*0.01);
108         s(j,i)=conj(V(j))*(V(j)-V(i))*(2.941-%i
            *11.764)+conj(V(j))*V(j)*(%i*0.01);
109     end
110 end
111 P=real(s);
112 Q=-imag(s);
113 printf('\nLine Flows\nThe following matrix shows the
        real part of line flows(in pu)');disp(P);
114 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

```

1 //Chapter 6
2 //Example 6.7
3 //page 234
4 //To find modified Ybus after including regulating
    transformer
5 clear;clc;
6
7 y34=2-%i*6;y23=0.666-%i*2;
8 y12=2-%i*6;y24=1-%i*3;
9 y13=1-%i*3;
10
11 //case(i) when a=1/1.04;
12 a=1/1.04;
13 //to form Ybus matrix

```

```

14 Y11=y13+y12;Y12=-y12;Y13=-y13;Y14=0;
15 Y21=-y12;Y22=y12+y23+y24;Y23=-y23;Y24=-y24;
16 Y31=-y13;Y32=-y23;Y33=(a^2)*y34+y23+y13;Y34=-(a')*
    y34;
17 Y41=0;Y42=-y24;Y43=-a'*y34;Y44=y34+y24;
18
19
20 Ybus=[Y11 Y12 Y13 Y14;
21       Y21 Y22 Y23 Y24;
22       Y31 Y32 Y33 Y34;
23       Y41 Y42 Y43 Y44];
24 printf('Case(i) When a=1/1.04');
25 printf('\nYbus=');disp(Ybus);
26 printf('\nObserve the changes in elements between
    bus 3&4 when compared with the result of
    example_6.2');
27
28 //case(ii) when a=e^(-j3)
29
30 a=cosd(3)-%i*sind(3);
31 //to form Ybus matrix
32 Y11=y13+y12;Y12=-y12;Y13=-y13;Y14=0;
33 Y21=-y12;Y22=y12+y23+y24;Y23=-y23;Y24=-y24;
34 Y31=-y13;Y32=-y23;Y33=(abs(a)^2)*y34+y23+y13;Y34=(a
    ')*(-y34);
35 Y41=0;Y42=-y24;Y43=a*(-y34);Y44=y34+y24;
36
37
38 Ybus=[Y11 Y12 Y13 Y14;
39       Y21 Y22 Y23 Y24;
40       Y31 Y32 Y33 Y34;
41       Y41 Y42 Y43 Y44];
42 printf('\n\nCase(ii) When a=e^(-j3)');
43 printf('\nYbus=');disp(Ybus);
44 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

```

1 //Chapter 6
2 //Example 6.8
3 //page 226
4 //To find load flow solution using the decoupled NR
   method and FDLF method
5 clear;clc;
6
7 //
   ///////////////////////////////////////////////////////////////////
8 //Pd      Qd      Pg      Qg      V
   Bus      Type/////
9 //
   ///////////////////////////////////////////////////////////////////
10 Pd1=2.0;    Qd1=1.0;    Pg1=0;    Qg1=0;    V1=1.04;
   //1    slack bus
11 Pd2=0;      Qd2=0;      Pg2=0.5;    Qg2=1;    V2=1;
   //2    PQ bus
12 Pd3=1.5;    Qd3=0.6;    Pg3=0.0;    Qg3=0;    V3=1.04;
   //3    PV bus
13 //
   ///////////////////////////////////////////////////////////////////
14 [V1_mag ,V1_ang]=polar(V1);
15 [V2_mag ,V2_ang]=polar(V2);
16 [V3_mag ,V3_ang]=polar(V3);
17 y_series=1/(0.02+%i*0.08);
18 y_self=2*y_series;
19 y_off=-1*y_series;

```

```

20 Ybus=[y_self y_off y_off;y_off y_self y_off;y_off
      y_off y_self];
21
22 [y_bus_mag_21 ,y_bus_ang_21]=polar(Ybus(2,1));
23 [y_bus_mag_22 ,y_bus_ang_22]=polar(Ybus(2,2));
24 [y_bus_mag_23 ,y_bus_ang_23]=polar(Ybus(2,3));
25 [y_bus_mag_31 ,y_bus_ang_31]=polar(Ybus(3,1));
26 [y_bus_mag_32 ,y_bus_ang_32]=polar(Ybus(3,2));
27 [y_bus_mag_33 ,y_bus_ang_33]=polar(Ybus(3,3));
28 [y_bus_mag_11 ,y_bus_ang_11]=polar(Ybus(1,1));
29
30 //case(a) Decoupled NR method :
31 printf( '\ncase(a) Decoupled NR method :\n' ) ;
32
33 H22=0.96+23.508i;
34 H23=-1.04+11.764i;
35 H33=25.89i;
36 L22=1+23.508i;
37 H=[H22 H23;H23 H33];
38 delta_P=[0.73;-1.62];
39
40 delta_V_ang=inv(H)*delta_P;
41 delta_V2_ang=delta_V_ang(1,1);
42 delta_V3_ang=delta_V_ang(2,1);
43 printf( '\ndelta_Angle_V2=' ); disp(real(delta_V2_ang))
      ;
44 printf( '\ndelta_Angle_V3=' ); disp(real(delta_V3_ang))
      ;
45 V2_ang=V2_ang-delta_V2_ang;
46 V3_ang=V3_ang-delta_V3_ang;
47
48 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);
49
50 printf( '\nQ2=' ); disp(real(Q2));
51 delta_Q2=(Qg2-Qd2)-(Q2);

```

```

52 printf( '\ndelta_Q2=' ); disp( real( delta_Q2 ) );
53 L=[L22];
54 delta_v=inv(L)*delta_Q2;
55 delta_V2=delta_v*V2_mag;
56
57 printf( '\ndelta_V2=%0.3 f ', delta_V2 );
58 V2_mag=V2_mag+delta_V2;
59 printf( '\n\nV2=%0.3 f pu ', V2_mag );
60
61 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);
62
63 printf( '\n\nQ3=' ); disp( real( Q3 ) );
64
65 //case(b) FDLF method:
66
67 printf( '\n\n\ncase(b) FDLF method :\n' ) ;
68
69 //
      ////////////////////////////////////////////////////////////////////
70 //Pd          Qd          Pg          Qg          V
      //          Bus      Type////////
71 //
      ////////////////////////////////////////////////////////////////////
72 Pd1=2.0;      Qd1=1.0;      Pg1=0;      Qg1=0;      V1=1.04;
      //1      slack bus
73 Pd2=0;        Qd2=0;        Pg2=0.5;    Qg2=1;      V2=1;
      //2      PQ bus
74 Pd3=1.5;      Qd3=0.6;      Pg3=0.0;    Qg3=0;      V3=1.04;
      //3      PV bus
75 //
      ////////////////////////////////////////////////////////////////////
76 [V1_mag, V1_ang]=polar(V1);

```

```

77 [V2_mag,V2_ang]=polar(V2);
78 [V3_mag,V3_ang]=polar(V3);
79 y_series=1/(0.02+%i*0.08);
80 y_self=2*y_series;
81 y_off=-1*y_series;
82 Ybus=[y_self y_off y_off;y_off y_self y_off;y_off
      y_off y_self];
83
84 [y_bus_mag_21,y_bus_ang_21]=polar(Ybus(2,1));
85 [y_bus_mag_22,y_bus_ang_22]=polar(Ybus(2,2));
86 [y_bus_mag_23,y_bus_ang_23]=polar(Ybus(2,3));
87 [y_bus_mag_31,y_bus_ang_31]=polar(Ybus(3,1));
88 [y_bus_mag_32,y_bus_ang_32]=polar(Ybus(3,2));
89 [y_bus_mag_33,y_bus_ang_33]=polar(Ybus(3,3));
90 [y_bus_mag_11,y_bus_ang_11]=polar(Ybus(1,1));
91
92 B22=-23.508;
93 B23=11.764;
94 B32=B23;
95 B33=B22;
96
97 B=[-B22 -B23;-B32 -B33];
98
99 delta_P=[0.73;-1.557];
100
101 delta_V_ang=inv(B)*delta_P;
102 delta_V2_ang=delta_V_ang(1,1);
103 delta_V3_ang=delta_V_ang(2,1);
104 printf('\ndelta_Angle_V2=');disp(real(delta_V2_ang))
      ;
105 printf('\ndelta_Angle_V3=');disp(real(delta_V3_ang))
      ;
106 V2_ang=V2_ang-delta_V2_ang;
107 V3_ang=V3_ang-delta_V3_ang;
108
109 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);
110
111 delta_Q2 = (Qg2 - Qd2) - (Q2);
112
113 delta_v = inv([-B22]) * delta_Q2;
114 delta_V2 = delta_v * V2_mag;
115
116 printf('\ndelta_V2=%0.3f', delta_V2);
117 V2_mag = V2_mag + delta_V2;
118 printf('\n\nV2=%0.3f pu', V2_mag);
119
120 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);
121
122 printf('\n\nQ3='); disp(real(Q3));

```

Chapter 7

Optimal System Operation

Scilab code Exa 7.1 Incremental cost and load sharing

```
1 //Chapter 7
2 //Example 7.1
3 //page 246
4 //To find incremental cost and load sharing
5 clear;clc;
6
7 ///Let us use the program given in the Appendix G in
   the textbook to write
8 //a function that returns the value of lamda and
   Loading of each generator
9 //when the total load on the plant is sent to the
   function
10
11 function [lamdaprev,Pg]=optimum(Pd)
12     n=2; //number of generators
13     Alpha=[0.2 0.25];
14     Beta=[40 30];
15     lamda=35; //initial guess for lambda
16     lamdaprev=lamda;
17     eps=1; //tolerance
18     deltalamda=0.25; //increment in lamda
```

```

19         Pgmax=[125 125];
20         Pgmin=[20 20];
21         Pg=100*ones(n,1);
22         while abs(sum(Pg)-Pd)>eps
23             for i=1:n
24                 Pg(i)=(lamda-Beta(i))/Alpha(i);
25                 if Pg(i)>Pgmax(i) then
26                     Pg(i)=Pgmax(i);
27                 end
28                 if Pg(i)<Pgmin(i) then
29                     Pg(i)=Pgmin(i);
30                 end
31             end
32             if (sum(Pg)-Pd)<0 then
33                 lamdaprev=lamda;
34                 lamda=lamda+deltalamda;
35             else
36                 lamdaprev=lamda;
37                 lamda=lamda-deltalamda;
38             end
39         end
40 endfunction
41
42
43 //to draw the table 7.1
44 printf('Table 7.1 Output of each unit and plant
         output for various values of lamda\n')
45 printf('
         _____\
         n');
46 printf('Plant Lamda,          Unit 1          Unit 2
         Plant Output  \n');
47 printf('Rs/MWh          Pg1 ,MW          Pg2 ,MW
         (Pg1+Pg2) ,MW  \n');
48 printf('
         _____\
         n');
49

```

```

50 Pd_matrix=[40 76 130 150 175 220 231.25 250];
51 for i=1:8
52     [lamda ,Pg]=optimum(Pd_matrix(i));
53     printf( '%0.2 f          %0.2 f          %0.2 f
              %0.2 f\n' , lamda ,Pg(1) ,Pg(2) ,Pg(1)+Pg
              (2));
54 end
55 printf( '
              n' );
56
57 //To draw the Graphs 7.3 and 7.4
58
59 Pd_test=40:3.75:250;
60 [Pd_ro ,Pd_co]=size(Pd_test)
61 for i=1:Pd_co
62     [lamda ,Pg]=optimum(Pd_test(i));
63     lamda_test(i)=lamda;
64     Pg1_test(i)=Pg(1);
65     Pg2_test(i)=Pg(2);
66 end
67 Pg1_test=Pg1_test.'; //transposing without
    conjugating
68 Pg2_test=Pg2_test.';
69 lamda_test=lamda_test.';
70
71 subplot(211)
72 plot(Pd_test ,lamda_test);
73 title('Incremental Fuel cost versus plant output');
74 xlabel('Plant output ,MW');
75 ylabel('Incremental fuel cost ,Rs/MWh');
76 set(gca() , "grid" , [0,0])
77 get("current_axes");
78
79 subplot(212)
80 plot(Pd_test ,Pg1_test ,Pd_test ,Pg2_test);
81 title('Output of each unit versus plant output');
82 xlabel('Plant output ,MW');

```

```

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

```

Scilab code Exa 7.2 Savings by optimal scheduling

```

1 //Chapter 7
2 //Example 7.2
3 //page 248
4 //To find the saving in fuel cost by optimal
   scheduling
5 clear;clc;
6
7 //Example reveals that for optimal load sharing
   units 1&2 has to take up 50MW and 80MW
   respectively
8 //If each unit supplies 65MW,increase in cost for
   units 1&2 are
9
10 Increase1=integrate('0.2*Pg1+40','Pg1',50,65);
11 Increase2=integrate('0.25*Pg2+30','Pg2',80,65);
12 printf('\nIncrease in cost for unit 1 is = %0.1f Rs/
   hr ',Increase1);
13 printf('\n\nIncrease in cost for unit 2 is = %0.3f
   Rs/hr ',Increase2);
14 printf('\n\nNet saving caused by optimum scheduling
   is = %0.3f Rs/hr ',Increase1+Increase2);
15 printf('\n\nTotal yearly saving assuming continuous
   operation= Rs %d',(Increase1+Increase2)*24*365);

```

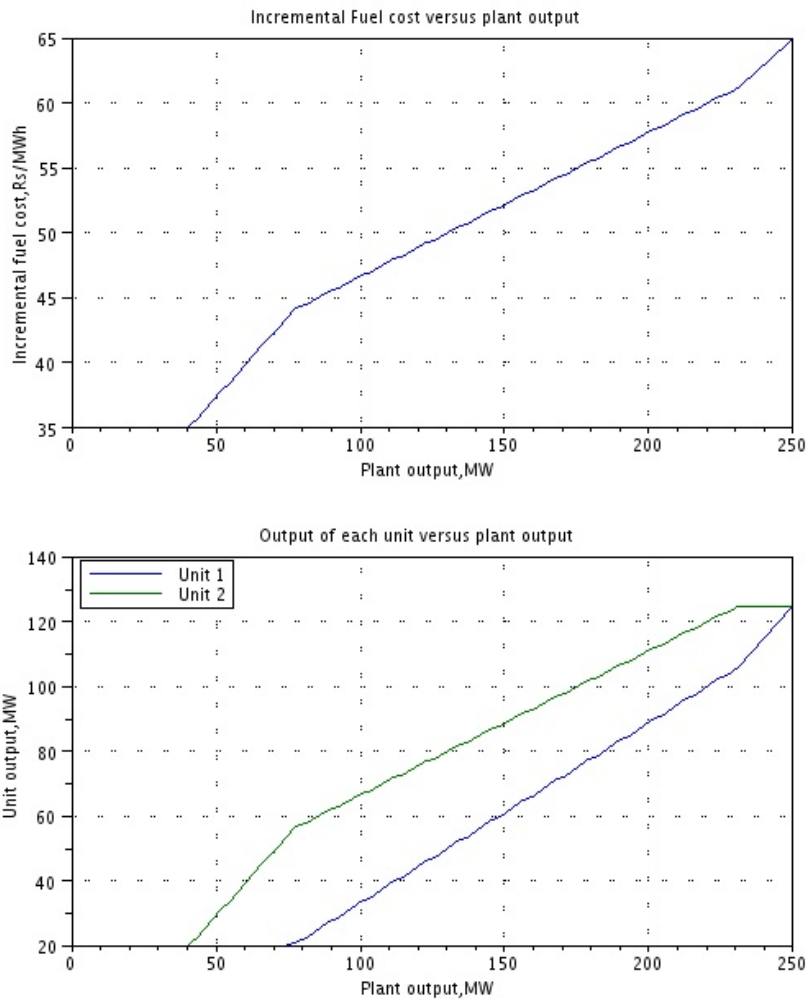


Figure 7.1: Incremental cost and load sharing

Scilab code Exa 7.3 Economical operation

```
1 //Chapter 7
2 //Example 7.3
3 //page 249
4 //To find the economical operation
5 clear;clc;
6
7 //from the table we got as the output in the
  example_7_1
8 //for optimum operation of load 220MW,unit 1&2 must
  be loaded 100MW and 120MW respwctively
9 //and for a load of 76MW,unit 1&2 must be loaded 20
  MW and 56MW respwctively
10 start_up=400;
11 //case(i)
12 printf('\nCase(i)');
13 //total fuel cost for the load of 220MW during 6AM
  to 6PM
14 Pg1=100;
15 Pg2=120;
16 C1=0.1*Pg1^2+40*Pg1+120;
17 C2=0.125*Pg2^2+30*Pg2+100;
18 total1=(C1+C2)*12;
19 printf('\nTotal fuel cost for the load of 220MW
  during 6AM to 6PM = Rs. %d',total1);
20
21 //total fuel cost for the load of 76MW during 6PM to
  6AM
22 Pg1=20;
23 Pg2=56;
24 C1=0.1*Pg1^2+40*Pg1+120;
```

```

25 C2=0.125*Pg2^2+30*Pg2+100;
26 total2=(C1+C2)*12;
27 printf('\nTotal fuel cost for the load of 76MW
    during 6PM to 6AM if both the units run = Rs. %d'
    ,total2);
28
29 total=total1+total2; //total fuel cost for 24hrs
30
31 printf('\nTotal fuel cost for the load during 24hrs
    if both the units run = Rs. %d',total);
32
33 //case(ii)
34 printf('\n\nCase(ii)');
35 //If during light load condition unit2 is On and
    Unit1 is Off then
36 Pg2=76;
37 C2=0.125*Pg2^2+30*Pg2+100;
38 total2=C2*12;
39 total_case2=total1+total2+start_up;
40
41 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

```

1 //Chapter 7
2 //Example 7.4
3 //page 263
4 //To find required generation for each plant and
    losses incurred
5 clear;clc;
6

```

```

7  ///Let us use the program given in the Appendix G in
   the textbook which includes penalty factor also
   to write
8  //a function that returns the value of lamda,Loading
   of each generator and losses
9  //when the total load on the plant is sent to the
   function
10
11 function [lamda ,Pg,PL]=optimum2(Pd)
12 n=2; //no of generators
13 Alpha=[0.02 0.04];
14 Beta=[16 20];
15 lamda=20; //initial value of lamda
16 lamdaprev=lamda;
17 eps=1; //tolerance
18 deltalambda=0.1;
19 Pgmax=[200 200];
20 Pgmin=[0 0];
21 B=[0.001 0;0 0];
22 Pg=zeros(n,1);
23 noofiter=0;
24 PL=0;
25 Pg=zeros(n,1);
26 while abs(sum(Pg)-Pd-PL)>eps
27     for i=1:n
28         sigma=B(i,:)*Pg-B(i,i)*Pg(i);
29         Pg(i)=(1-(Beta(i)/lamda)-(2*sigma))/(Alpha(i)
           )/lamda+2*B(i,i));
30         PL=Pg.'*B*Pg;
31         if Pg(i)>Pgmax(i) then
32             Pg(i)=Pgmax(i);
33         end
34         if Pg(i)<Pgmin(i) then
35             Pg(i)=Pgmin(i);
36         end
37     end
38     PL=Pg.'*B*Pg;
39     if (sum(Pg)-Pd-PL)<0 then

```



```

40         lamdaprev=lamda;
41         lamda=lamda+deltalamda;
42     else
43         lamdaprev=lamda;
44         lamda=lamda-deltalamda;
45     end
46     noofiter=noofiter+1;
47     Pg;
48 end
49 endfunction
50
51 //In this example let us take the answer .i.e load(
    Pd)=237.04MW and calculate
52 //lamda so that we can use the algorithm used in the
    textbook
53 Pd=237.04
54 [lamda_test,Pg_test,PL_test]=optimum2(Pd);
55 printf('\nLagrange''s multiplier (lamda) is\n Lamda
    =%0.1f',lamda_test);
56 printf('\n\nRequired generation for optimum loading
    are \n Pg1=%0.2f MW \n Pg2=%d MW\n',Pg_test(1),
    Pg_test(2));
57 printf('\nThe transmission power loss is\n PL=%0.2f
    MW',PL_test);
58 printf('\n\nThe load is \n Pd=%0.2f MW',Pd);

```

Scilab code Exa 7.5 Savings on coordination of losses

```

1 //Chapter 7
2 //Example 7.5
3 //page 264
4 //To find savings when losses are coordinated
5 clear;clc;

```

```

6
7 function [lamdaprev,Pg]=optimum(Pd)
8     n=2; //number of generators
9     Alpha=[0.02 0.04];
10    Beta=[16 20];
11    lamda=20; //initial guess for lambda
12    lamdaprev=lamda;
13    eps=1; //tolerance
14    deltalamda=0.25; //increment in lamda
15    Pgmax=[200 200];
16    Pgmin=[0 0];
17    Pg=100*ones(n,1);
18    while abs(sum(Pg)-Pd)>eps
19        for i=1:n
20            Pg(i)=(lamda-Beta(i))/Alpha(i);
21            if Pg(i)>Pgmax(i) then
22                Pg(i)=Pgmax(i);
23            end
24            if Pg(i)<Pgmin(i) then
25                Pg(i)=Pgmin(i);
26            end
27        end
28        if (sum(Pg)-Pd)<0 then
29            lamdaprev=lamda;
30            lamda=lamda+deltalamda;
31        else
32            lamdaprev=lamda;
33            lamda=lamda-deltalamda;
34        end
35    end
36 endfunction
37
38 //the above function "optimum" doesn't coordinate
39     losses
40 //case(i) when the losses are included but not
41     coordinated
42 [lamda_case1,Pg_case1]=optimum(237.04);

```

```

42 //since Pg2 does not supply transmission losses and
    the losses are supplied only by Pg1
43 Pg2_1=Pg_case1(2);
44 //to get Pg1 we will solve  $Pg1+Pg2=0.001*Pg1^2+237.04$ 
45 //the above equation can be written as  $(0.001*Pg1^2) - Pg1 +(237.04-Pg2) =0$ 
46 p=poly([0.001 -1 (237.04+Pg2_1)],"Pg1");
47 Pg1_1=roots(p);
48 Pg1_1=Pg1_1(1);
49
50 printf('\ncase(i) when the losses are included but
    not coordinated ');
51 printf('\nPg1=%0.2 f MW    Pg2=%0.2 f MW',Pg1_1 ,Pg2_1)
    ;
52
53 //case(ii) when the losses are also coordinated
54 //we have the solution for case(ii) from example_7_4
55 Pg1_2=128.57; Pg2_2=125; //case(ii)
56
57 printf('\n\ncase(ii) when the losses are coordinated
    ');
58 printf('\nPg1=%0.2 f MW    Pg2=%0.2 f MW',Pg1_2 ,Pg2_2)
    ;
59
60 //saving at plant 1 is
61 saving1=integrate('0.02*Pg1+16','Pg1',Pg1_2 ,Pg1_1);
62 printf('\n\nSaving at plant 1 due to loss
    coordination is = Rs %0.2 f/hr',saving1);
63
64 //saving at plant 2 is
65 saving2=integrate('0.04*Pg2+20','Pg2',Pg2_2 ,Pg2_1);
66 printf('\n\nSaving at plant 2 due to loss
    coordination is = Rs %0.2 f/hr',saving2);
67
68 //net savings achieved
69 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

```
1 //Chapter 7
2 //Example 7.6
3 //page 268
4 //To calculate the loss formula coefficients of the
   system
5 clear;clc;
6
7 Ia=2-%i*0.5;      Ic=1-%i*0.25;
8 Ib=1.6-%i*0.4;    Id=3.6-%i*0.9;
9 Za=0.015+%i*0.06;  Zc=0.01+%i*0.04;
10 Zb=0.015+%i*0.06;  Zd=0.01+%i*0.04;
11
12 ID=Id+Ic ;//total load current
13
14 //calculation of current distribution factors
15 printf('\nCurrent distribution factors are :\n')
16 Ma1=(ID/ID);
17 Ma2=(0/ID);
18 Mb1=(-Ic/ID);
19 Mb2=(Id/ID);
20 Mc1=(Ic/ID);
21 Mc2=(Ic/ID);
22 Md1=(Id/ID);
23 Md2=(Id/ID);
24 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);
25
26 //bus voltage calculation
```

```

27 [V1_mag ,V1_ang]=polar (1.0+Ia*Za);
28 [V2_mag ,V2_ang]=polar (1+Ib*Zb);
29 V1_ang=real (V1_ang)*180/%pi;
30 V2_ang=real (V2_ang)*180/%pi;
31 printf( '\n\nBus voltages are given by \nV1=%0.3 f @
        %0.2 fdeg PU\tV2=%0.3 f @ %0.2 fdeg PU',V1_mag ,
        V1_ang ,V2_mag ,V2_ang);
32
33 //current phase angles at the plants
34 sigma1=atand( imag (Ia)/real (Ia));
35 sigma2=atand( imag (Ib+Ic)/real (Ib+Ic));
36 printf( '\n\nCurrent phase angles at the plants\
        nSigma1=%ddeg\tSigma2=%ddeg',sigma1 ,sigma2);
37
38 //plant power factors
39 pf1=cosd(V1_ang-sigma1);
40 pf2=cosd(V2_ang-sigma2);
41 printf( '\n\nThe plant power factors are\npf1=%0.4 f\
        tpf2=%0.4 f',pf1 ,pf2);
42
43 //calculation of loss coefficients
44 B11=(Ma1*Ma1*real (Za)+Mb1*Mb1*real (Zb)+Mc1*Mc1*real (
        Zc)+Md1*Md1*real (Zd))/(V1_mag*V1_mag*pf1*pf1);
45 B22=(Ma2*Ma2*real (Za)+Mb2*Mb2*real (Zb)+Mc2*Mc2*real (
        Zc)+Md2*Md2*real (Zd))/(V2_mag*V2_mag*pf2*pf2);
46 B12=(Ma1*Ma2*real (Za)+Mb1*Mb2*real (Zb)+Mc1*Mc2*real (
        Zc)+Md1*Md2*real (Zd))/(V1_mag*V2_mag*pf1*pf2);
47 printf( '\n\nThe Loss coefficients in PU are \nB11=%0
        .5 f pu\nB22=%0.5 f pu\nB12=%0.5 f pu',B11 ,B22 ,B12);
48 printf( '\n\nThe Loss coefficients in reciprocal
        megawatts are \nB11=%0.8 f MW^-1\nB22=%0.8 f MW^-1\
        nB12=%0.8 f MW^-1',B11/100 ,B22/100 ,B12/100);

```

Scilab code Exa 7.7 Optimal generation schedule for hydrothermal system

```
1 //Chapter 7
2 //Example 7.7
3 //page 281
4 //To find the optimal generation schedule for a
   typical day of the fundamental hydrothermal
   system
5 clear;clc;
6
7 h_b=20; //basic head of the water
8 e=0.005; //head correction factor
9 r=2; //non-effective water discharge
10 Pd_1=7; Pd_2=10; Pd_3=5; //load at three intervals
   of time during a day
11 alpha=0.5; //positive scalar
12 X_0=100; //initial water storage in the reservoir
13 X_3=60; //final water storage in the reservoir
14 //let us assume the initial values of the control
   variables
15 q_2=15;
16 q_3=15;
17 i=0; //iteration count
18 grad_2=1;grad_3=1; //inital value for iterations
19
20 while ((grad_2>0.1)|(grad_3>0.1))
21
22 //water discharge in the first interval
23 q_1=X_0-X_3-(q_2+q_3);
24
25 //water level after the first intervals are
26 X_1=X_0-q_1;
27 X_2=X_1-q_2;
28
29 //hydro generations in the subintervals
30 Pgh_1=9.81*(10^-3)*20*(1+0.5*e*(X_1+X_0))*(q_1-r);
31 Pgh_2=9.81*(10^-3)*20*(1+0.5*e*(X_2+X_1))*(q_2-r);
```

```

32 Pgh_3=9.81*(10^-3)*20*(1+0.5*e*(X_3+X_2))*(q_3-r);
33
34 //thermal generation in the three intervals
35 Pgt_1=Pd_1-Pgh_1;
36 Pgt_2=Pd_2-Pgh_2;
37 Pgt_3=Pd_3-Pgh_3;
38
39 //calculating lamda_1 for three subintervals
40 lamda_1_1=Pgt_1+25;
41 lamda_1_2=Pgt_2+25;
42 lamda_1_3=Pgt_3+25;
43
44 //since we are considering lossless case
45 lamda_3_1=lamda_1_1;
46 lamda_3_2=lamda_1_2;
47 lamda_3_3=lamda_1_3;
48
49 //for calculating lamda_2 for three intervals
50 lamda_2_1=lamda_3_1*9.81*(10^-3)*20*(1+0.5*e*(2*X_0
    -2*q_1+r));
51 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
    ));
52 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
    ));
53
54 //calculation of gradient vector
55 grad_2=lamda_2_2-lamda_3_2*9.81*(10^-3)*20*(1+0.5*e
    *(2*X_1-2*q_2+r));
56 grad_3=lamda_2_3-lamda_3_3*9.81*(10^-3)*20*(1+0.5*e
    *(2*X_2-2*q_3+r));
57
58 q_2=q_2-alpha*grad_2; //updating value of q and
    reiterating
59 q_3=q_3-alpha*grad_3;
60 i=i+1;
61 end

```

```

62
63 //Hydel and thermal generation for the three sub
    intervals are given in tabular format
64 printf('\nResults for Optimal Loading of
    Hydrothermal stations at the end of %d iterations
    ',i);
65 printf('\n
    -----
    n');
66 printf('Interval\t\tLoad\t\tHydro\t\tThermal\t\t
    tWater discharge\n');
67 printf('
    \t\tMW\t\tMW\t\tMW\t\tm^3/s\n');
68 printf('
    -----
    n');
69 printf('    1    \t\t%d\t\t%.4f\t\t%.4f\t\t%.2f\n'
    ,Pd_1,Pgh_1,Pgt_1,q_1);
70 printf('    2    \t\t%d\t\t%.4f\t\t%.4f\t\t%.2f\n'
    ,Pd_2,Pgh_2,Pgt_2,q_2);
71 printf('    3    \t\t%d\t\t%.4f\t\t%.4f\t\t%.2f\n'
    ,Pd_3,Pgh_3,Pgt_3,q_3);
72 printf('
    -----
    n');
    -----

```


Chapter 8

Automatic Generation and Voltage Control

Scilab code Exa 8.1 Frequency change Calculation

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

```

14 printf('\nExcess energy input to rotating parts in
    0.4 sec=%d kW-sec ',EE/1000);
15 printf('\nFrequency at the end of 0.4 sec=%0.2 f Hz\n\
    n',fnew);

```

Scilab code Exa 8.2 Load sharing and System Frequency

```

1 //Chapter 8
2 //Example 8.2
3 //page 301
4 //To determine determine load sharing and system
    frequency
5 clear;clc;
6 f=50; // system frequency
7 x=200; //load on first generator(value is assumed
    first)
8 delta_f=0.01*x; //from the first equation given in
    the book
9 x=(3.75)/(0.01+0.00625); //by substituting (i) in (
    ii)
10 delta_f=0.01*x; //recalculating the value
11 x2=600-x;
12 printf('\nLoad shared by the generator:\n\t
    Generator1=%0.2 f MW\n\tGenerator2=%0.2 f MW\n',x,
    x2);
13 printf('\nSystem Frequency=%0.2 f Hz\n\n',f-delta_f);

```

Chapter 9

Symmetrical Fault Analysis

Scilab code Exa 9.1 Fault Current Calculation

```
1 //Chapter 9
2 //Example 9.1
3 //page 335
4 //To calculate fault current
5 clear;clc;
6 //selecting base KVA and MVA
7 mvab=100;
8 Gmva=10;
9 T1mva=10; T2mva=5;
10 Gkvb=11; //generator kV base
11 OHLkvb=33; //overhead line kV base
12 Ckvb=6.6; // cable kB base
13 xg1=%i*0.15; xg2=%i*0.125; xt1=%i*0.10; xt2=%i*0.08;
14 xOHL=0.27+%i*0.36 ; xcab= 0.135+%i*0.08;
15
16 //clculating PU impedances
17
18 xg1=(xg1*mvab)/Gmva;
19 xg2=(xg2*mvab)/Gmva;
20 xt1=(xt1*mvab)/T1mva;
21 xt2=(xt2*mvab)/T2mva;
```

```

22 xOHL=(30*xOHL*mvab)/(OHLkvb^2);
23 xcab=(3*xcab*mvab)/(Ckvb^2);
24 //displaying results
25 printf('\n Reactance of G1= j%0.1f pu \n',abs(imag(
    xg1)));
26 printf(' Reactance of G2= j%0.1f pu\n',abs(imag(xg2)
    ));
27 printf(' Reactance of T1= j%0.1f pu\n',abs(imag(xt1)
    ));
28 printf(' Reactance of T2= j%0.1f pu\n',abs(imag(xt2)
    ));
29 printf(' Overhead line impedance=(%0.3f + j%0.3f) pu
    \n',real(xOHL),abs(imag(xOHL)));
30 printf(' Cable impedance= (%0.3f + j%0.3f) pu\n',
    real(xcab),abs(imag(xcab)));
31
32 // Impedance diagram is as shown in the figure9.7 in
    the textbook
33 // A XCOS simulation for this proble is done to
    explain the subtransient ,transient and steady
    state periods of a symmetrical short circuit
34 xtotal=((xg1*xg2)/(xg1+xg2)+xt1+xt2+xOHL+xcab);
35 Isc_pu=(1/xtotal);
36 Ibase=(mvab/(sqrt(3)*Ckvb))*1000;
37 Isc=Isc_pu*Ibase;
38 x_F_to_bus=(xt1+xt2+xOHL+xcab);
39 v_11b=x_F_to_bus*Isc_pu*11;
40 //displaying results
41 printf('\nTotal impedance= %0.1f < %0.2f deg pu \n',
    abs(xtotal),atand(imag(xtotal)/real(xtotal)));
42 printf('Short circuit current= %d A\n',abs(Isc));
43 printf('Voltage at 11kV bus=%0.2f kV\n',abs(v_11b));

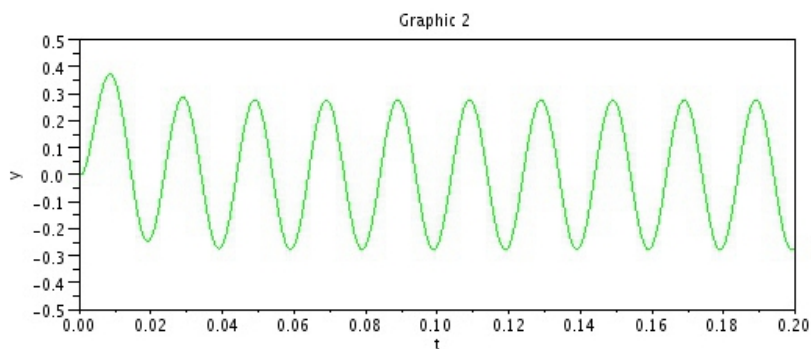
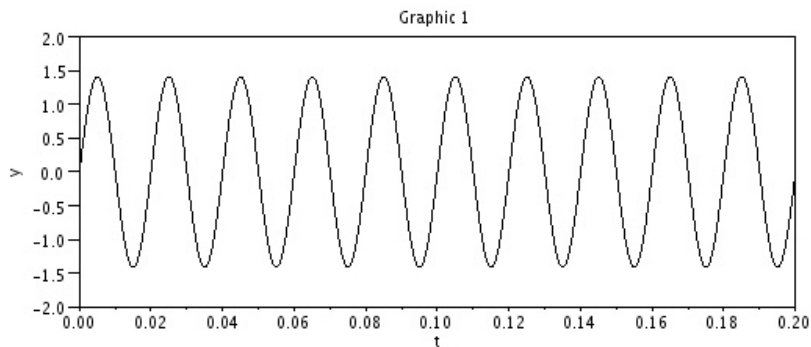
```

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

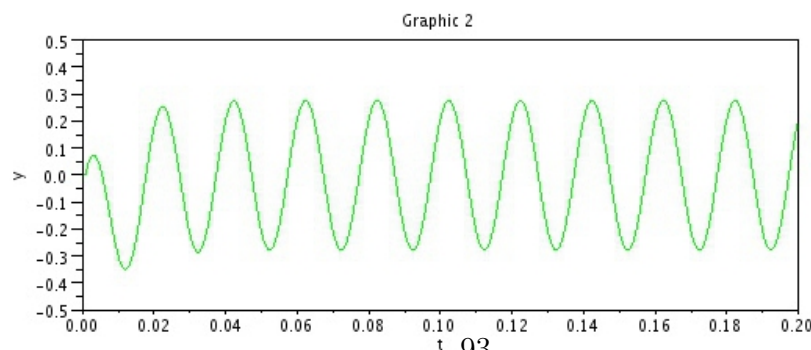
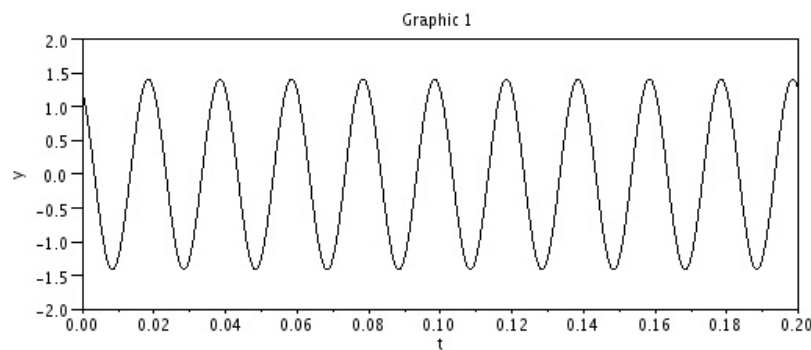
Scilab code Exa 9.2 Subtransient and Momentary current Calculation

```
1 //Chapter 9
2 //Example 9.2
3 //page 337
4 //To calculate subtransient and momentary current
5 clear;clc;
6 mvab=25;
7 Gmva=25;
8 T1mva=25; T2mva=25;
9 Gkvb=11; //generator kV base
10 OHLkvb=66; //overhead line kV base
11 Mkvb=6.6; //motor kV base
12 Mmva=5; //motor mva
13
14 XdG=%i*0.2; //Generator's subtransient reactance
15 XdM=%i*0.25; //Motor's subtransient reactance
16 XdM2=%i*0.3; //Motor's transient reactance
17 Xt1=%i*0.1; // step up transformer's reactance
18 Xt2=%i*0.1; //step down transformer's reactance
19 Xt1=%i*0.15 ;//trnasmission line's reactance
20
21 //per unit calculation
22 XdM=(XdM*mvab)/Mmva ;//perunit impedance of each
   motor
23 printf('\nSubtransient reactance of each motor = j%0
   .2 f pu\n',abs(XdM));
```

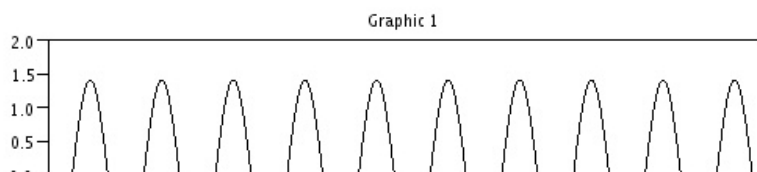
R-Phase Voltage and Fault Current



Y-Phase Voltage and Fault Current



B-Phase Voltage and Fault Current



```

24
25 //(a)subtransient current in the fault
26 Isc=(3*(1/XdM))+(1/(XdG+Xt1+Xt2+Xt1));
27 Ibase=(mvab*1000)/(sqrt(3)*Mkvb);
28 Isc=Isc*Ibase;
29 printf('\nSubtransient current in the fault =%0.1fA\n
      n',abs(Isc));
30
31 //(b)subtransient current in the breaker B
32 IscB=(2*(1/XdM))+(1/(XdG+Xt1+Xt2+Xt1));
33 IscB=IscB*Ibase;
34 printf('\nSubtransient current in breaker B=%0.1fA\n
      ',abs(IscB));
35
36 //(c) to find the momentary current through breaker
      B
37 ImomB=1.6*IscB;
38 printf('\nMomentary current through the breaker B=
      %dA\n',abs(ImomB));
39
40 //(d) to compute current to be interrupted by
      breaker in 5 cycles
41 XdM2=(XdM2*mvab)/Mmva ;//perunit transient impedance
      of each motor
42 IscB=(2*(1/XdM2))+(1/(XdG+Xt1+Xt2+Xt1));
43 IscB=IscB*Ibase;
44 ImomB=1.1*IscB;
45 printf('\nCurrent to be interrupted by breaker B in
      five cycles=%dA\n',abs(ImomB));

```

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

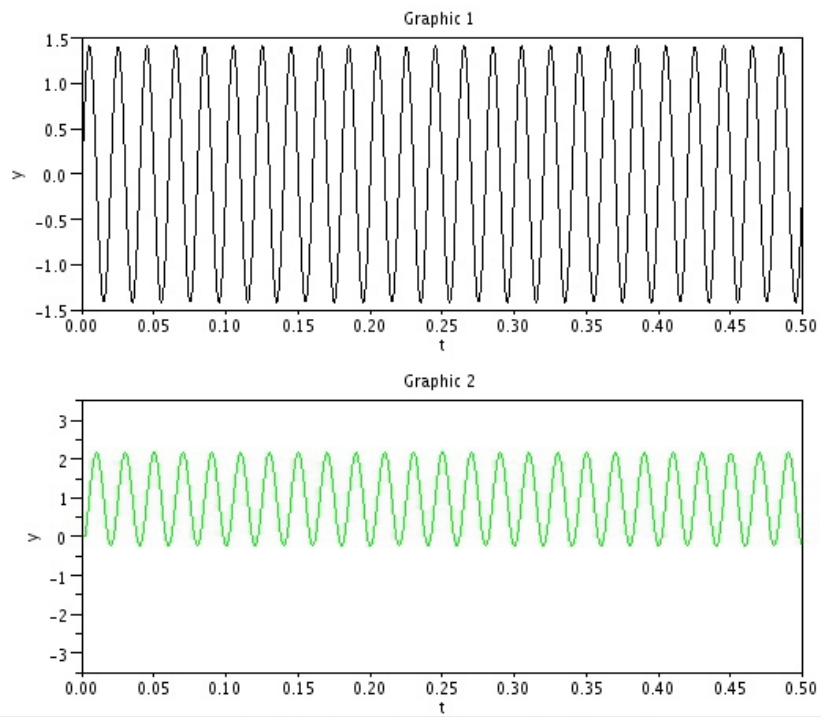


Fig 9.2 : Generator Voltage and Fault current

Figure 9.2: Subtransient and Momentary current Calculation

Scilab code Exa 9.3 Subtransient Current Calculation

```
1 //Chapter 9
2 //Example 9.3
3 //page 340
4 //To calculate subtransient current in Generator,
   Motor and fault
5 clear;clc;
6 mvab=25;
7 kvb=11;
8 Vo=10.6/kvb; //PU Prefault voltage
9 printf('\nPrefault Voltage = %0.4fpu\n',Vo);
10
11 Load=15/mvab; //load PU with 0.8pf leading
12 Io=(Load/(Vo*0.8))*(cosd(36.9)+%i*sind(36.9)); //
   Prefault current
13 printf('\nPrefault current = %0.4f at %0.1f deg PU'
   ,abs(Io),atand(imag(Io)/real(Io)));
14
15 Eg=Vo+(%i*0.45*Io); //voltage behind subtransient
   reactance(generator)
16 printf('\n\nVoltage behind subtransient reactance(
   Generator) = %0.4f+j%0.2f pu\n''',real(Eg),imag(
   Eg));
17
18 Em=Vo-(%i*0.15*Io); //voltage behind subtransient
   reactance(motor)
19 printf('\nVoltage behind subtransient reactance(
   Motor) = %0.4f-j%0.4f pu',real(Em),abs(imag(Em)))
   ;
20
21 Ig=Eg/(%i*0.45); //under fault condition
22 Im=Em/(%i*0.15); //under fault condition
23 printf('\n\nUnder Faulted condition \n Ig'''=%0.4f-
```

```

    j%0.4 f pu ', real(Ig), abs(imag(Ig)));
24 printf('\n Im'"=%0.4 f-j%0.4 f pu ', real(Im), abs(imag(
    Im)));
25 If=Ig+Im; //Current in fault
26 printf('\n\nCurrent in fault= -j%0.4 f pu ', abs(imag(
    If)));
27
28 Ib=(mvab*1000/(sqrt(3)*11)); //Base current
29 //Actual Currents
30 printf("\n\nNow");
31 Ig=Ig*Ib
32 Im=Im*Ib
33 If=If*Ib
34 printf('\nIg'"= %0.1 f-j%0.1 f A', real(Ig), abs(imag(Ig
    )));
35 printf('\nIm'"= %0.1 f-j%0.1 f A', real(Im), abs(imag(Im
    )));
36 printf('\nIf= -j%d A', abs(imag(If)));

```

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

Scilab code Exa 9.4 Maximum MVA Calculation

```

1 //Chapter 9
2 //Example 9.4
3 //page 345
4 //To calculate maximum MVA
5 clear;clc;
6 mvab=50;
7 kvb=6.6;

```

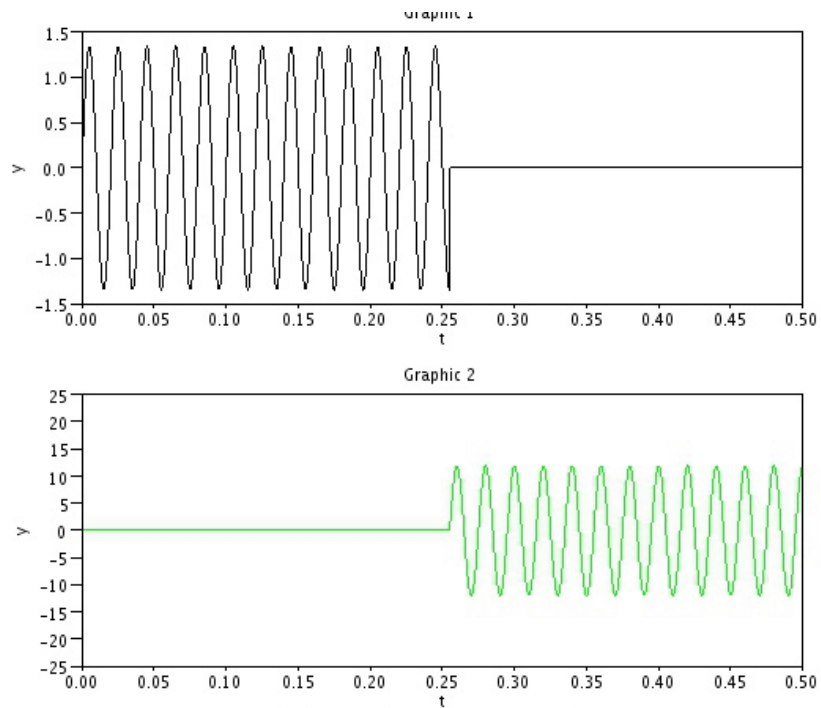


Fig 9.3 : Voltage and Current at the fault

Figure 9.3: Subtransient Current Calculation

```

8 mvaA=40;
9 mvaB=50;
10 mvaC=25;
11 feeder_impedance=((0.06+%i*0.12)*mvab)/(kvb^2)
12
13 Gen_A_reactance=(%i*0.1*mvab/mvaA);
14 Gen_B_reactance=(%i*0.1*mvab/mvaB);
15 Gen_C_reactance=(%i*0.1*mvab/mvaC);
16
17 printf('\nGenerator A reactance = j%0.3f pu',abs(
    Gen_A_reactance));
18 printf('\nGenerator B reactance = j%0.3f pu',abs(
    Gen_B_reactance));
19 printf('\nGenerator C reactance = j%0.3f pu',abs(
    Gen_C_reactance));
20
21 Reactor_A_reactance=(%i*0.12*mvab/mvaA);
22 Reactor_B_reactance=(%i*0.12*mvab/mvaB);
23 Reactor_C_reactance=(%i*0.12*mvab/mvaC);
24
25 printf('\nReactor A reactance = j%0.3f pu',abs(
    Reactor_A_reactance));
26 printf('\nReactor B reactance = j%0.3f pu',abs(
    Reactor_B_reactance));
27 printf('\nReactor C reactance = j%0.3f pu',abs(
    Reactor_C_reactance));
28
29 function resistance=parallel(r1,r2)
30 resistance=(r1*r2/(r1+r2));
31 endfunction
32
33 Z=(feeder_impedance)+parallel(%i*0.125,(%i*0.15 +
    parallel(%i*0.22,%i*0.44)));
34 scmva=(1/abs(Z))*mvab;
35 printf("\n\nSC MVA = %d MVA",scmva);

```

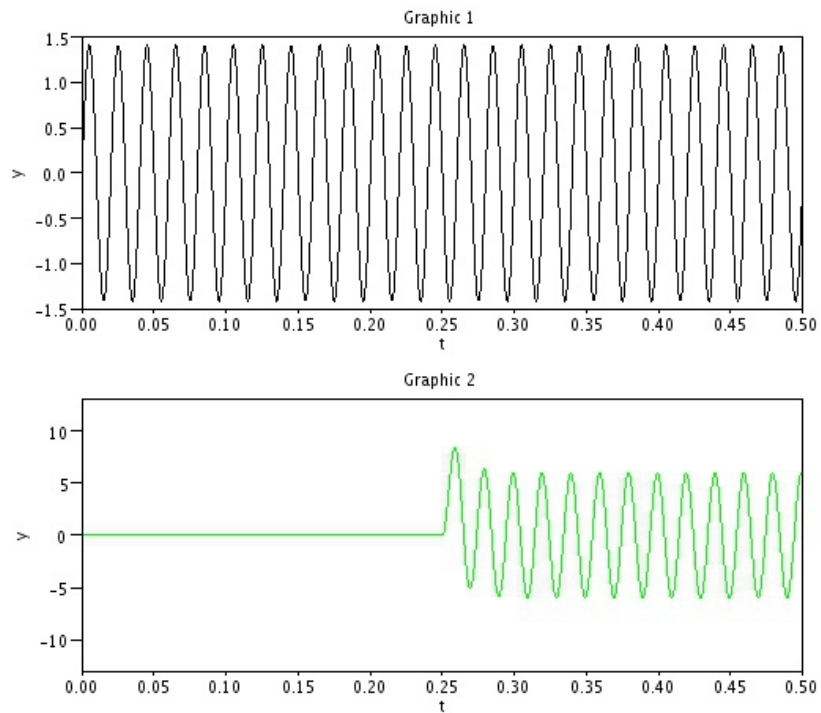


Fig 9.4: Generator Voltage and Fault Current Waveform

Figure 9.4: Maximum MVA Calculation

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

Scilab code Exa 9.5 Short Circuit Solution

```

1 //Chapter 9
2 //Example 9.5
3 //page 347

```

```

4 //To calculate short circuit solution
5 clear;clc;
6 //referring to figures 9.19 in the text book,we get
   directly the fault current
7 V4o=1.0;
8 Zf=%i*0.13560;
9 If=V4o/Zf;
10 printf('\nIf= -j%0.5 f pu\n\n',abs(If));
11
12 //From Fig9.19d
13 I1=If*((%i*0.19583)/(%i*0.37638));
14 I2=If*((%i*0.18055)/(%i*0.37638));
15 printf('I1 = -j%0.5 f pu \n\nI2 = -j%0.5 f pu\n\n',abs
   (I1),abs(I2));
16
17 //voltage changes for bus 1,2 and 3
18 deltaV1=0-(%i*0.15)*I1;
19 deltaV2=0-(%i*0.15)*I2;
20 printf('DeltaV1=%0.5 f pu\n\nDeltaV2=%0.5 f pu\n\n',
   deltaV1,deltaV2);
21
22 //reffering to book
23 V1f=1+deltaV1;
24 V2f=1+deltaV2;
25 printf('V1f= %0.5 f pu\n\nV2f=%0.5 f pu\n\n',V1f,V2f);
26 I13=(V1f-V2f)/(%i*0.15+%i*0.1);
27 printf('I13=j%0.5 f pu\n\n',abs(I13));
28 deltaV3=0-((%i*0.15)*(I1)+(%i*0.15)*(I13));
29 Vf3=1+deltaV3;
30 printf('DeltaV3=%0.5 f pu\n\n',deltaV3);
31 printf('Vf3=%0.5 f pu\n\n',Vf3);
32 Vf4=0;
33 printf('Vf4=%d\n\n',Vf4);
34 //short circuit MVA at bus 4
35 SC_MVA_4=abs(If)*100;
36 printf('Short circuit MVA at bus4 =%0.3 f MVA',
   SC_MVA_4);

```

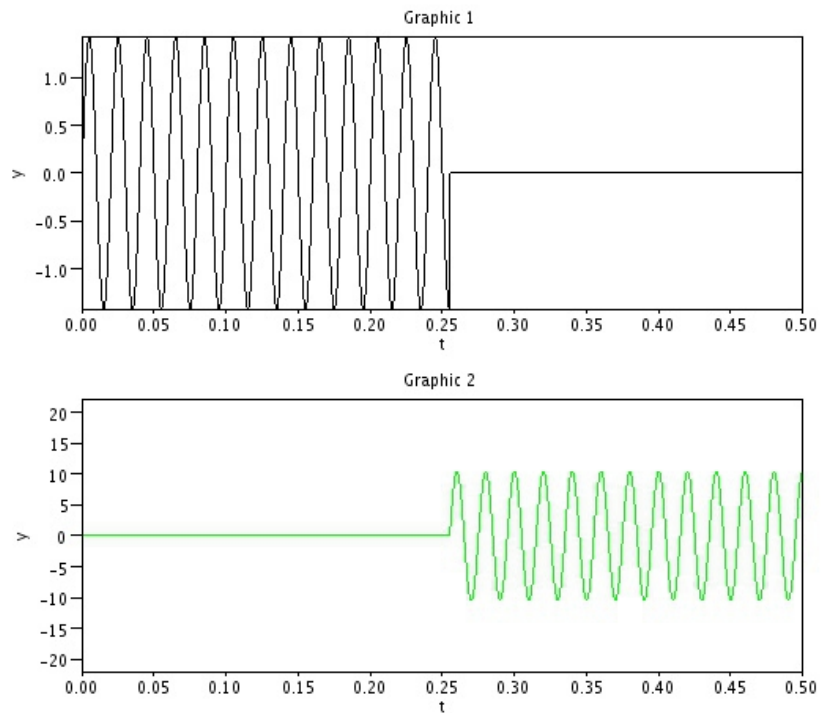


Fig 9.5: Voltage and Current at the Fault

Figure 9.5: Short Circuit Solution

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

Scilab code Exa 9.6 Short Circuit Solution using Algorithm

```

1 //Chapter 9
2 //Example 9.6
3 //page 352

```

```

4 //To calculate short circuit solution using
   algorithm for short circuit studies
5 clear;clc;
6
7 Y11=1/(0.15*%i)+1/(0.15*%i)+1/(0.1*%i)+1/(0.2*%i);
8 Y12=-1/(0.2*%i);
9 Y21=Y12;
10 Y13=-1/(0.15*%i);
11 Y31=Y13;
12 Y14=-1/(0.1*%i);
13 Y41=Y14;
14 Y22=1/(0.15*%i)+1/(0.15*%i)+1/(0.1*%i)+1/(0.2*%i);
15 Y23=-1/(0.1*%i);
16 Y32=Y23;
17 Y24=-1/(0.15*%i);
18 Y42=Y24;
19 Y33=1/(0.15*%i)+1/(0.1*%i);
20 Y34=0;
21 Y43=Y34;
22 Y44=1/(0.15*%i)+1/(0.1*%i);
23
24 //Ybus matrix can be written as
25
26 Ybus=[Y11 Y12 Y13 Y14;Y21 Y22 Y23 Y24;Y31 Y32 Y33
        Y34;Y41 Y42 Y43 Y44];
27
28 Zbus=inv(Ybus);
29
30 //preault voltages
31 V10=1;V20=1;V30=1;V40=1;
32
33 //post fault voltages
34 V1f=V10-(Zbus(1,4)/Zbus(4,4))*V40;
35 V2f=V20-(Zbus(2,4)/Zbus(4,4))*V40;
36 V3f=V30-(Zbus(3,4)/Zbus(4,4))*V40;
37 V4f=V40-(Zbus(4,4)/Zbus(4,4))*V40;
38
39 //to calculate fault current through Zf=0

```



```

40 If=V40/(Zbus(4,4)+0);
41
42 //short circuit current in lines 1-3,1-2,1-4,2-4 and
    2-3
43
44 I13f=(V1f-V3f)/(0.15*%i);
45 I12f=(V1f-V2f)/(0.2*%i);
46 I14f=(V1f-V4f)/(0.1*%i);
47 I24f=(V2f-V4f)/(0.15*%i);
48 I23f=(V2f-V3f)/(0.1*%i);
49
50 //If at all fault occurs on bus1 or bus2
51 If12=1/Zbus(1,1);
52
53 //displaying the results
54 printf('\n Ybus=');
55 disp(Ybus);
56
57 printf('\n Zbus=');
58 disp(Zbus);
59
60 printf('\nV1f= %0.4 f pu',V1f);
61 printf('\nV2f= %0.4 f pu',V2f);
62 printf('\nV3f= %0.4 f pu',V3f);
63 printf('\nV4f= %0.1 f pu\n',V4f);
64
65 printf('\nFault current=-j%0.5 f pu\n',abs(If));
66
67 printf('\nI13f=j%0.3 f pu',abs(I13f));
68 printf('\nI12f=j%0.3 f pu',abs(I12f));
69 printf('\nI14f=-j%0.3 f pu',abs(I14f));
70 printf('\nI24f=-j%0.3 f pu',abs(I24f));
71 printf('\nI23f=-j%0.3 f pu\n',abs(I23f));
72
73 printf('\n Fault current for a fault on bus 1 (or
    bus 2)\n If=-j%0.6 f pu\n\n',abs(If12));

```

Scilab code Exa 9.7 Current Injection Method

```
1 //Chapter 9
2 //Example 9.7
3 //page 355
4 //To evaluate Zbus using Current Injection method
5 clear;clc;
6
7 disp("We can approach this problem using XCOS
      simulation")
8 disp("In this simulation");
9 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");
10 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");
11
12 Z11=7;
13 Z21=4;
14 Z12=Z21;
15 Z22=6;
16
17 Zbus=[Z11 Z12;Z21 Z22]
```

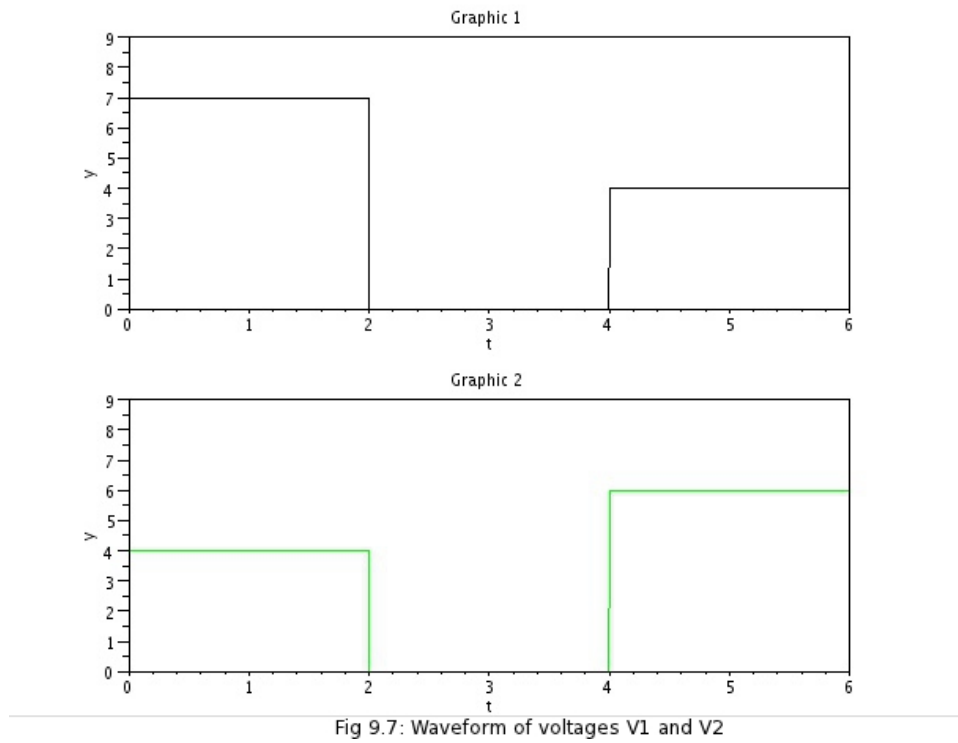


Figure 9.6: Current Injection Method

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

Scilab code Exa 9.8 Zbus matrix building using Algorithm

```

1 //Chapter 9
2 //Example 9.8
3 //page 360

```

```

4 //To build Zbus matrix using Zbus building algorithm
5 clear;clc;
6
7 disp("Let us go on modifying Zbus by including nodes
      and the elements connected to it one by one as
      given in the textbook")
8
9 //step-1 type1 modification
10 Zbus=[0.25];
11 printf('\nstep-1 type1 modification\nZbus=');disp(
      Zbus);
12
13 //step-2 type2 modification
14 Zbus=[Zbus ,0.25;0.25,0.25+0.1];
15 printf('\nstep-2 type2 modification\nZbus=');disp(
      Zbus);
16
17 //step-3 type2 modification
18 Zbus=[Zbus [0.25;0.25]; 0.25 0.25 0.35];
19 printf('\nstep-3 type2 modification\nZbus=');disp(
      Zbus);
20
21 //step-4 type3 modification
22 Zbus=Zbus-(1/(Zbus(3,3)+0.25))*[Zbus(1:3,2:2)]*[Zbus
      (2:2,1:3)];
23 printf('\nstep-4 type3 modification\nZbus=');disp(
      Zbus);
24
25 printf('This is the final Zbus matrix after
      including all the elements\n');
26 //step-5 type4 modification
27 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)];
28 printf('\nstep-5 type4 modification\nZbus=');disp(
      Zbus);
29
30 disp("opening a line between 2-3 is equivalent to

```

```

    connecting (-0.1) between bus3 bus2")
31 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)];
32 printf('Zbus='); disp(Zbus); printf('(same as in step
    4)');

```

Scilab code Exa 9.9 PostFault Currents and Voltages Calculation

```

1 //Chapter 9
2 //Example 9.9
3 //page 362
4 //To find postfault currents and voltages
5 clear;clc;
6
7 disp("The Thevenin passive network for this system
    is drawn in Example_9_8 (or fig 9.28 in the
    textbook)");
8 disp("Using the Zbus matrix from the results of
    example_9_8,we can calculate post fault currents
    and voltages");
9 Zbus=%i*[0.1397059 0.1102941 0.125;0.1102941
    0.1397059 0.125;0.125 0.125 0.175]
10
11 //to find fault current
12 V30=1;V10=1;V20=1;
13 If=(V30/(Zbus(3,3)+0));
14 printf('\nIf=-j%0.2f pu\n',abs(If));
15
16
17 //to find postfault voltages
18 V1f=V10-(Zbus(1,3)/Zbus(3,3));
19 V2f=V20-(Zbus(2,3)/Zbus(3,3));

```

```

20 printf( '\nV1f=%0.3 f ',V1f);
21 printf( '\nV2f=%0.3 f ',V2f);
22
23 //to find fault currents in the TL
24 I12f=(V1f-V2f)/(%i*0.1);
25 I13f=(V1f-0)/(%i*0.1);
26 I23f=(V2f-0)/(%i*0.1);
27 printf( '\n\nI12f=%d', I12f);
28 printf( '\nI13f=-j%0.2 f ',abs(I13f));
29 printf( '\nI23f=-j%0.2 f ',abs(I23f));
30
31 //to find generator currents during faults
32 Eg1=1;Eg2=1;
33 Ig1f=(Eg1-V1f)/(0.2*%i+0.05*%i);
34 Ig2f=(Eg2-V2f)/(0.2*%i+0.05*%i);
35 printf( '\n\nIg1f=-j%0.2 f ',abs(Ig1f));
36 printf( '\nIg2f=-j%0.2 f\n\n',abs(Ig2f));

```

Chapter 10

Symmetrical Components

Scilab code Exa 10.1 Symmetrical components of line currents Calculation

```
1 //Chapter 10
2 //Example 10.1
3 //page 374
4 //To calculate symmetrical components of line
  currents
5 clear;clc;
6 Ia=10*(cosd(30)+%i*sind(30));
7 Ib=15*(cosd(-60)+%i*sind(-60));
8 // from KCL Ia+Ib+Ic=0
9 Ic=-(Ia+Ib);
10 //defining alpha(a)
11 a=cosd(120)+(%i*sind(120));
12 Ip=[Ia;Ib;Ic];
13 A=[1 1 1;a^2 a 1;a a^2 1];
14 IA=inv(A)*Ip;
15 IB=diag([a^2,a,1])*IA;
16 IC=diag([a,a^2,1])*IA;
17
18 function [r,theta]=phasorform(x)
19     r=abs(x);
```

```

20     theta=atand(imag(x),real(x));
21 endfunction
22
23 [IAr,IAth]=phasorform(IA);
24 [IBr,IBth]=phasorform(IB);
25 [ICr,ICth]=phasorform(IC);
26
27 //to display the results of symmetrical components
    of line currents
28
29 printf('\n\nIA1=%0.2 f @ %d deg A',IAr(1,1),IAth(1,1)
    );
30 printf('\n\nIA2=%0.2 f @ %d deg A',IAr(2,1),IAth(2,1));
31 printf('\n\nIA0=%0.2 f A',IAr(3,1));
32
33
34 printf('\n\nIB1=%0.2 f @ %d deg A',IBr(1,1),IBth(1,1)
    );
35 printf('\n\nIB2=%0.2 f @ %d deg A',IBr(2,1),IBth(2,1));
36 printf('\n\nIB0=%0.2 f A',IBr(3,1));
37
38
39 printf('\n\nIC1=%0.2 f @ %d deg A',ICr(1,1),ICth(1,1)
    );
40 printf('\n\nIC2=%0.2 f @ %d deg A',ICr(2,1),ICth(2,1));
41 printf('\n\nIC0=%0.2 f A',ICr(3,1));
42
43 //to calculate Delta currents
44
45 IAB=(Ia-Ib)/3;
46 IBC=(Ib-Ic)/3;
47 ICA=(Ic-Ia)/3;
48
49 //to get the results in phasor notation
50 [IABr,IABth]=phasorform(IAB);
51 [IBCr,IBCth]=phasorform(IBC);
52 [ICAr,ICAth]=phasorform(ICA);
53

```



```

54 printf( '\n\nIAB=%0.2 f @ %d deg A', IABr, IABth);
55 printf( '\nIBC=%0.2 f @ %d deg A', IBCr, IBCth);
56 printf( '\nICA=%0.2 f @ %d deg A', ICAr, ICAtH);
57
58 //to calculate the symmetrical components of delta
    currents by reusing the variable Ip
59 Ip=[IAB; IBC; ICA];
60 IAB=inv(A)*Ip;
61 IBC=diag([a^2, a, 1])*IAB;
62 ICA=diag([a, a^2, 1])*IAB;
63
64 [IABr, IABth]=phasorform(IAB);
65 [IBCr, IBCth]=phasorform(IBC);
66 [ICAr, ICAtH]=phasorform(ICA);
67
68 //to display the results of symmetrical components
    of Delta currents
69
70 printf( '\n\nIAB1=%0.2 f @ %d deg A', IABr(1,1), IABth
    (1,1));
71 printf( '\nIAB2=%0.2 f @ %d deg A', IABr(2,1), IABth
    (2,1));
72 printf( '\nIAB0=%0.2 f A', IABr(3,1));
73
74
75 printf( '\n\nIBC1=%0.2 f @ %d deg A', IBCr(1,1), IBCth
    (1,1));
76 printf( '\nIBC2=%0.2 f @ %d deg A', IBCr(2,1), IBCth
    (2,1));
77 printf( '\nIBC0=%0.2 f A', IBCr(3,1));
78
79
80 printf( '\n\nICA1=%0.2 f @ %d deg A', ICAr(1,1), ICAtH
    (1,1));
81 printf( '\nICA2=%0.2 f @ %d deg A', ICAr(2,1), ICAtH
    (2,1));
82 printf( '\nICA0=%0.2 f A\n\n', ICAr(3,1));

```

Scilab code Exa 10.2 Sequence Network of the System

```
1 //Chapter 10
2 //Example 10.2
3 //page no 390
4 //To draw sequence networks of the system
5 clear;clc;
6
7 //selecting generator rating as base in generator
  circuit
8
9 mvab=25;
10 kvGb=11; //base voltage for generator
11 kvTLb=kvGb*(121/10.8); //base voltage for TL
12 kvMb=kvTLb*(10.8/121); //base voltage for motors
13
14 xG=%i*0.2;
15 xT=%i*0.1;
16 xTL=100;
17 xM=%i*0.25;
18
19 mvaG=25;
20 mvaT=30;
21 mvaM1=15;
22 mvaM2=7.5;
23
24 kvM=10;
25
26 //converting all the reactances to PUs
27
28 xT=xT*(mvab/mvaT)*(10.8/kvGb)^2;
29 xTL=xTL*(mvab/(kvTLb)^2);
```

```

30 xM1=xM*(mvab/mvaM1)*(kvM/kvMb)^2;
31 xM2=xM*(mvab/mvaM2)*(kvM/kvMb)^2;
32
33 //displaying the results
34
35 printf('\n\nTransmission line voltage base = %0.1f
      kV',kvTLb);
36 printf('\n\nMotor voltage base = %d kV',kvMb);
37 printf('\n\nTransformer reactance = %0.4f pu',abs(
      imag(xT)));
38 printf('\nLine reactance = %0.3f pu',abs(xTL));
39 printf('\nReactance of motor 1 = %0.3f pu',abs(imag(
      xM1)));
40 printf('\nReactance of motor 2 = %0.3f pu\n\n',abs(
      imag(xM2)));
41
42 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 www.scilab.in

Scilab code Exa 10.3 Zero sequence Network

```

1 //Chapter 10
2 //Example 10.3
3 //page no 392
4 //To draw the zero sequence networks of the system
5 clear;clc;
6
7 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 www.scilab.in

Scilab code Exa 10.4 Zero Sequence Network

```
1 //Chapter 10
2 //Example 10.4
3 //page no 392
4 //To draw the zero sequence networks of the system
   given in example 10.2
5 clear;clc;
6
7 //selecting generator rating as base in generator
   circuit
8
9 mvab=25;
10 kvGb=11; //base voltage for generator
11 kvTLb=kvGb*(121/10.8); //base voltage for TL
12 kvMb=kvTLb*(10.8/121); //base voltage for motors
13
14 //Calculation of zero sequence reactance
15
16 xT0=0.0805; //zero sequence reactance of transformer
17 xG0=0.06; //zero sequence reactance of generator
18
19 //zero sequence reactanc eof motors
20 xM1_0=0.06*(mvab/15)*(10/kvMb)^2;
21 xM2_0=0.06*(mvab/7.5)*(10/kvMb)^2;
22
23 x_clr_0=3*2.5*(mvab/kvGb^2); // Reactance of current
   limiting reactors to be icluded in the zero
```

```

sequence network
24 x_TL_0=300*(mvab/kvTLb^2); //Zero sequence reactance
    of TL
25
26 printf('\n\nTransformer zero sequence reactance = %0
    .4f pu',xT0);
27 printf('\nGenerator zero sequence reactances = %0.2f
    pu',xG0);
28 printf('\nZero sequence reactance of motor 1 = %0.3f
    pu',xM1_0);
29 printf('\nZero sequence reactance of motor 2 = %0.3f
    pu',xM2_0);
30 printf('\nReactance of current limiting reactors =
    %0.3f pu',x_clr_0);
31 printf('\nZero sequence reactance of transmission
    line = %0.3f pu\n\n',x_TL_0);
32
33 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 www.scilab.in

Chapter 11

Unsymmetrical Fault Analysis

Scilab code Exa 11.1 LG and 3Phase faults Comparision

```
1 //Chapter 11
2 //Example 11.1
3 //page 406
4 //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
5 clear;clc;
6
7 disp("Sequence networks of synchronous generator
   grounded through neutral impedance has been drawn
   using XCOS ");
8
9 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");
10
11 disp("X1=j0.18, X2=j0.15, X0=j0.10 pu and Ea=1");
12
13 disp("From the figs 11.13 and 11.14 in the textbook,
   we can find Ilg and I3L");
```

```

14
15 Ea=1; X1=0.18*%i; X2=0.15*%i; X0=0.10*%i;
16
17 IaLG=3*Ea/(2*X1+X0)
18 Ia3L=3*Ea/(3*X1)
19
20 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 www.scilab.in

Scilab code Exa 11.2 Grounding Resistor voltage and Fault Current

```

1 //Chapter 11
2 //Example 11.2
3 //page 408
4 //To find fault current and voltage across the
   grounding resistor
5 clear;clc;
6
7 X1eq=(%i*0.18)/2;
8 X2eq=(%i*0.15)/2;
9 Z0eq=(%i*0.10)+3*(2*20/(11^2));
10
11 Ea=1;
12
13 //calculation of fault current
14 printf('\nFault current is given by ');
15 If=(3*Ea)/(X1eq+X2eq+Z0eq)

```

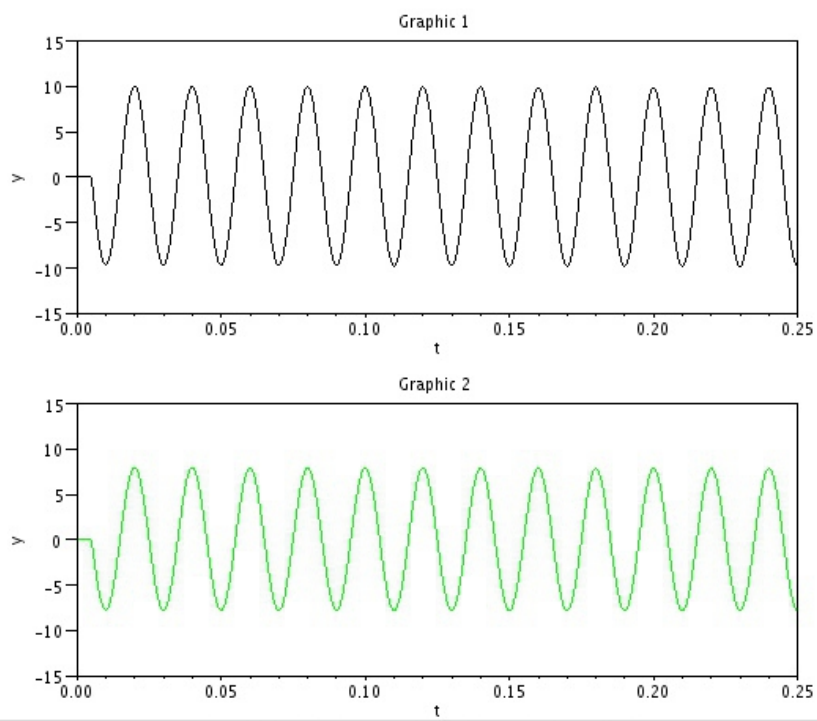


Fig 11.1: Fault current in LG and 3phase Faults

Figure 11.1: LG and 3Phase faults Comparison


```

16
17 //current in grounding resistor
18 Ifg=abs(If)*(20/(11*sqrt(3)));
19 printf('\n\nCurrent through grounding resistor Ifg=
    %0.2fkA ',Ifg);
20
21 //voltage across grounding resistor
22 Vgr=abs(If*(2*20/(11^2))*(11/sqrt(3)));
23 printf('\n\nVoltage across grounding resistor Vgr=%0
    .2fkV\n\n',Vgr);

```

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

Scilab code Exa 11.3 Fault and subtransient currents of the system

```

1 //Chapter 11
2 //Example 11.3
3 //page 409
4 //To find fault current and subtransient current in
    all parts of the system
5 clear;clc;
6
7 a=-0.5+(sqrt(3)/2)*%i;
8
9 //neglecting prefault currents
10 Vf0=10/11;
11 Eg=Vf0; Em1=Vf0 ;Em2=Vf0;
12
13 //positive sequence network when it is replaced by
    its thevenin's equivalent as shown in fig11.18

```

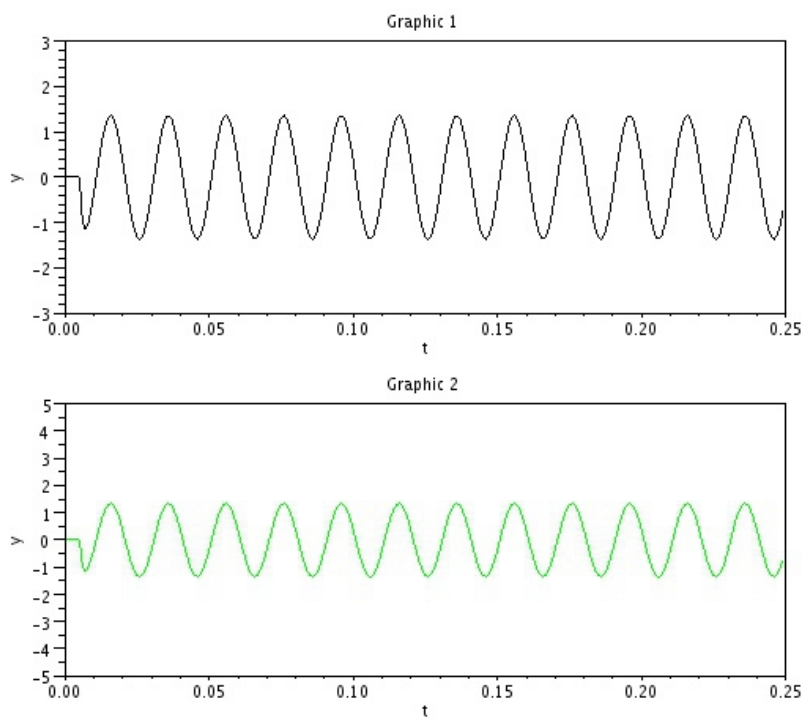


Fig 11.2: Fault current and Voltage across the grounding resistor

Figure 11.2: Grounding Resistor voltage and Fault Current

```

14 printf('\nsequence impedances are given by \n');
15 Z1=(%i*0.525*%i*0.23)/(%i*0.755);
16 Z2=Z1;
17 Z0=%i*1.712;
18 printf('Z1=j%0.4f \nZ2=j%0.4f \nZ0=j%0.4f',abs(imag(
    Z1)),abs(imag(Z2)),abs(imag(Z0)));
19 //to find sequence current
20 Ia1=Vf0/(Z1+Z2+Z0);
21 Ia2=Ia1;
22 Ia0=Ia1;
23
24 //to find fault current
25 If=3*Ia0;
26 printf('\n\nFault Current= -j%0.4f',abs(imag(If)));
27
28
29 //component current flowing from generator and motor
30 printf('\n\nComponents currents flowing from
    Generator and motor are \n');
31 Ig1=Ia1*(0.23/0.755) ;
32 Ig2=Ig1;
33 Ig0=0;
34 printf('Ig1= -j%0.4f \nIg2= -j%0.4f \nIg0=%d',abs(
    Ig1),abs(Ig2),abs(Ig0));
35 printf('\n');
36 Im1=Ia1*(0.525/0.755);
37 Im2=Im1;
38 Im0=Ia0;
39 printf('\nIm1= -j%0.4f \nIm2= -j%0.4f \nIm0= -j%0.4f
    ',abs(Im1),abs(Im2),abs(Im0));
40
41 //fault currents from the generator and motor
    towards g are
42 printf('\n\nFault current from the generator towards
    g are ');
43 Ig=[1 1 1;a^2 a 1;a a^2 1]*[Ig1;Ig2;Ig0];
44 disp(Ig);
45 printf('and to g from motors are');

```

```

46 Im=[1 1 1;a^2 a 1;a a^2 1]*[Im1;Im2;Im0];
47 disp(Im);
48
49 printf('\nPositive sequence current =%0.3f pu',(-%i*
      Ig1));
50 printf('\nNegative sequence current =%0.3f pu',(%i*
      Ig2));
51 printf('\nZero sequence current=%d\n',Ig0);
52
53 //under loaded condition,PU motor currents are
54 Im1o=(15/(25*0.909*0.8))*(0.800103636+%i
      *0.5998617938);
55 Im2o=(7.5/(25*0.909*0.8))*(0.800103636+%i
      *0.5998617938);
56 printf('\nThe per unit motor currents are:\n');
57 printf('Motor1:%0.2f +j%0.3f pu',real(Im1o),imag(
      Im1o));
58 printf('Motor2:%0.2f +j%0.3f pu',real(Im2o),imag(
      Im2o));
59
60 //the voltages behind subtransient reactances are
      calculated below
61 printf('\n\nVoltage behind subtransient reactances:\n
      n');
62 printf('Motor1:');
63 Em1=Em1-(%i*0.345*Im1o);
64 printf('Em1= %0.4f-j%0.4f',real(Em1),abs(imag(Em1)))
      ;
65
66 printf('\nMotor2:');
67 Em2=Em2-(%i*0.69*Im2o);
68 printf('Em2= %0.4f-j%0.4f',real(Em2),abs(imag(Em2)))
      ;
69
70 printf('\nGenerator:');
71 Eg=Eg+(%i*0.525*(Im2o+Im1o));
72 printf('Eg= %0.4fj+%0.4f',real(Eg),abs(imag(Eg)));
73

```

```

74 //actual value of positive sequence current from
    generator and motor
75 printf('\n\nThe actual value of positive sequence
    current from the generator towards fault is = %0
    .2 f+j%0.3 f ',real(Im1o+Im2o+Ig1),imag(Im1o+Im2o+
    Ig1));
76 printf('\n\nThe actual value of positive sequence
    current from the motors towards fault is = %0.2 f-
    j%0.3 f ',real(-Im1o-Im2o+Im1),abs(imag(-Im1o-Im2o+
    Im1)));

```

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

Scilab code Exa 11.4 LL Fault Current

```

1 //Chapter 11
2 //Example 11.4
3 //page 412
4 //To find L-L fault current and voltage of healthy
    phase
5 clc;clear;
6 X1eq=0.09*%i;
7 X2eq=0.075*%i;
8 Z0=0.99+(%i*0.1);
9 Ea=1;Ia0=0;
10
11 //to calculate Ia1
12 Ia1=Ea/(X1eq+X2eq);
13
14 //to calculate fault current

```

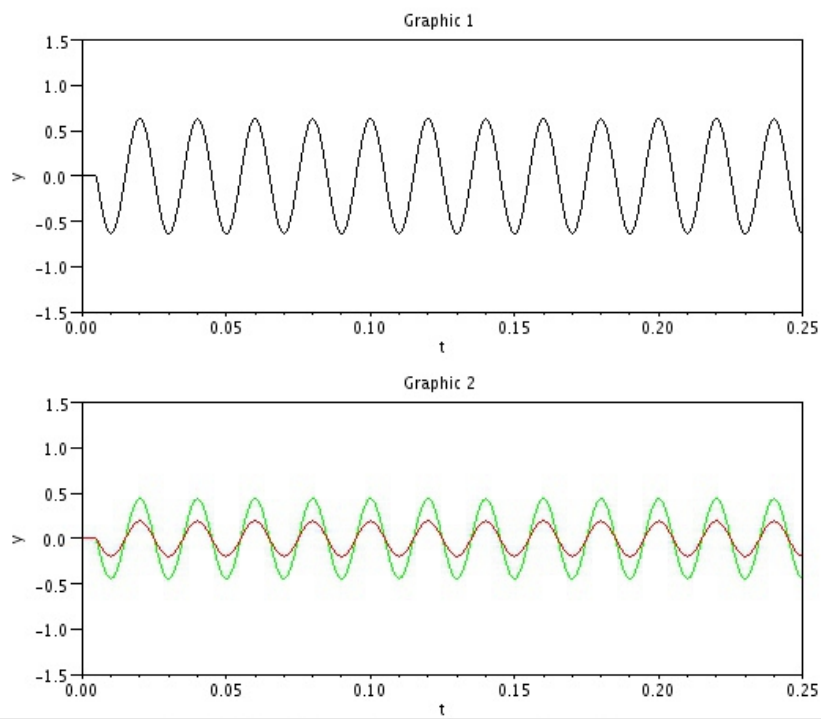


Fig 11.3: Fault Current.Generator and Motor side components

Figure 11.3: Fault and subtransient currents of the system

```

15 If=(-%i*sqrt(3))*(-%i*6.06);
16 Va1=Ea-(Ia1*X1eq);
17 Va0=(-Ia0*Z0);
18 Va2=Va1;
19
20 //voltage in healthy phase
21 Va=Va1+Va2+Va0;
22
23 //displaying the result
24 printf('\nIa1=-j%0.2f',abs(Ia1));
25 printf('\nIf=%0.3f',If);
26 printf('\nVa1=Va2=%0.3f',Va1);
27 printf('\nVa0=%d',Va0);
28 printf('\nVa=Va1+Va2+Va0=%0.2f\n\n',Va);

```

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

Scilab code Exa 11.5 Double line to ground Fault

```

1 //Chapter 11
2 //Example 11.5
3 //page 413
4 //To find Double line to ground fault current and
   voltage of healthy phase
5 clc;clear;
6
7 Z1eq=0.09*%i;
8 Z2eq=0.075*%i;
9 Z0=(%i*0.1);
10 Ea=1;

```

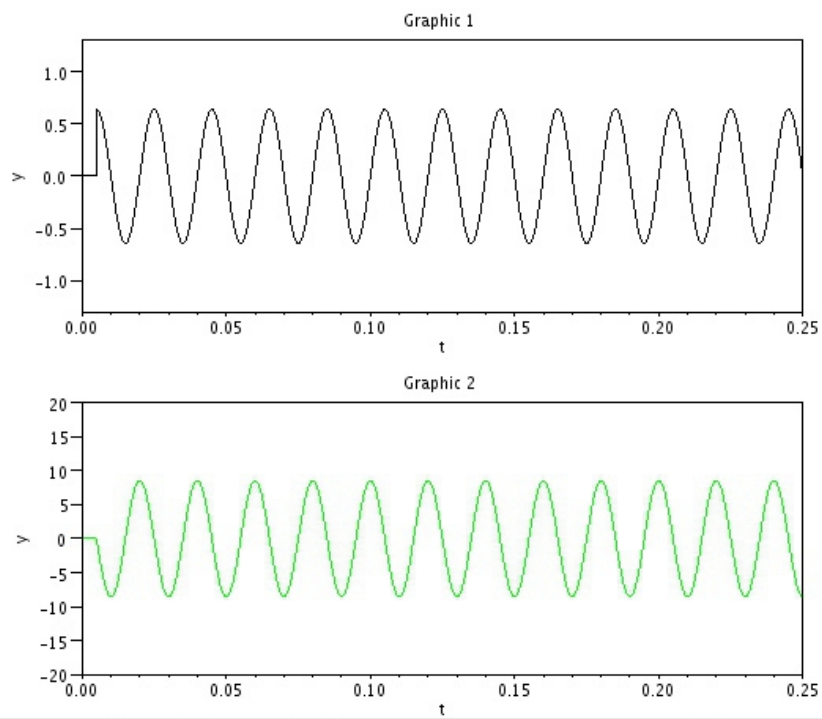


Fig 11.4: Measure of Positive sequence voltage and Fault current

Figure 11.4: LL Fault Current


```

11 a=(-0.5+%i*sqrt(3)/2);
12
13 //to find the sequence components of healthy phase
14 Ia1=Ea/(Z1eq+(Z2eq*Z0/(Z2eq+Z0)));
15 Va1=Ea-(Ia1*Z1eq);
16 Va2=Va1;
17 Va0=Va1;
18
19 Ia2=-(Va2/Z2eq);
20 Ia0=-(Va0/Z0);
21
22 I=[1 1 1;a^2 a 1;a a^2 1]*[Ia1; Ia2; Ia0];
23
24 //voltage of the healthy phase
25 Va=3*Va1;
26
27 //displaying the results
28 printf(' Ia1=-j%0.3 f\n',abs(Ia1));
29 printf(' Ia2=j%0.3 f\n',abs(Ia2));
30 printf(' Ia0=j%0.3 f\n\n',abs(Ia0));
31
32 printf(' Ia=%0.3 f + j%0.3 f\n',real(I(1,1)),imag(I
    (1,1)));
33 printf(' Ib=%0.3 f + j%0.3 f\n',real(I(2,1)),imag(I
    (2,1)));
34 printf(' Ic=%0.3 f + j%0.3 f\n\n',real(I(3,1)),imag(I
    (3,1)));
35
36 printf(' Voltage of the healthy phase Va=3Va1=%0.3 f'
    ,Va);

```

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

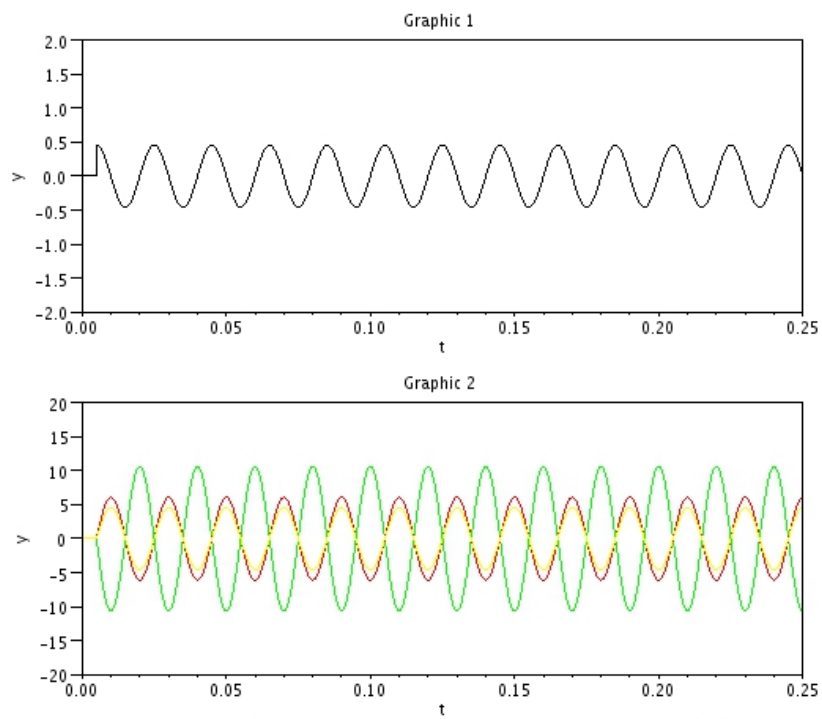


Fig 11.5: Positive sequence voltage and sequence currents

Figure 11.5: Double line to ground Fault

Scilab code Exa 11.6 Bus Voltages and Currents Calculations

```
1 //Chapter 11
2 //Example 11.6
3 //page 420
4 //To find bus voltages and currents
5
6 clc;clear;
7 v_pf=1; //prefault voltage
8 //according to the fig.11.26
9 Y1dd=((%i*0.2)^-1)+((%i*0.0805)^-1);
10 Y1fg=-(%i*0.0805)^-1;
11 Y1de=Y1fg;
12 Y1ff=((%i*0.0805)^-1)+((%i*0.164)^-1);
13 Y1ee=Y1ff;
14 Y1ef=-(%i*0.164)^-1;
15 Y1gg=((%i*0.0805)^-1)+((%i*0.345)^-1)+((%i*0.69)^-1)
    ;
16 Y1df=0;
17 Y1dg=0;
18 Y1ed=Y1de;
19 Y1eg=0;
20 Y1fd=0;
21 Y1fe=Y1ef;
22 Y1gd=0;
23 Y1ge=0;
24 Y1gf=Y1fg;
25 printf('\nY-Bus and Z-Bus matrix can be written as:\n
    n');
26 Y1_bus=[Y1dd Y1de Y1df Y1dg;Y1ed Y1ee Y1ef Y1eg;Y1fd
    Y1fe Y1ff Y1fg;Y1gd Y1ge Y1gf Y1gg];
27 Y2_bus=Y1_bus;
28 printf('\nY1_bus=');disp(Y1_bus);
29 printf('\nY2_bus=');disp(Y2_bus);
```

```

30 Y0dd=(%i*1.608)^-1;Y0de=0;Y0df=0;Y0dg=0;
31 Y0ed=0;Y0ee=((%i*0.0805)^-1)+((%i*0.494)^-1);Y0ef=-(%i*0.494)^-1;Y0eg=0;
32 Y0fd=0;Y0fe=Y0ef;Y0ff=Y0ee;Y0fg=0;
33 Y0gd=0;Y0ge=0;Y0gf=0;Y0gg=(%i*1.712)^-1;
34
35 Y0_bus=[Y0dd Y0de Y0df Y0dg;Y0ed Y0ee Y0ef Y0eg;Y0fd
          Y0fe Y0ff Y0fg;Y0gd Y0ge Y0gf Y0gg];
36 printf('\nY0_bus=');disp(Y0_bus);
37
38 //finding Z-bus matrix
39 Z1_bus=inv(Y1_bus);
40 Z2_bus=inv(Y2_bus);
41 Z0_bus=inv(Y0_bus);
42 printf('\n\nZ1bus=');disp(Z1_bus);
43 printf('\n\nZ2_bus=');disp(Z2_bus);
44 printf('\n\nZ0_bus=');disp(Z0_bus);
45
46 //to find fault current with LG fault on bus e ---
   case(i)
47 If_e=(3*v_pf)/(Z1_bus(2,2)+Z2_bus(2,2)+Z0_bus(2,2));
48 printf('\n\n\nFault current with LG fault on bus e
   is If_e= -j%0.5f\n',abs(imag(If_e)));
49
50 //to find fault current with LG fault on bus f ---
   case(ii)
51 If_f=(3*v_pf)/(Z1_bus(3,3)+Z2_bus(3,3)+Z0_bus(3,3));
52 printf('Fault current with LG fault on bus f is If_f
   = -j%0.5f\n',abs(imag(If_f)));
53
54 //to find bus voltages and line currents in case(i)
55 printf('\n\n\nBus voltages and currents are given
   below:\n\n');
56 Vf1_d=1-(Z1_bus(1,2)*If_e/3);
57 Vf1_e=1-(Z1_bus(2,2)*If_e/3);
58 Vf1_f=1-(Z1_bus(3,2)*If_e/3);
59 Vf1_g=1-(Z1_bus(4,2)*If_e/3);
60 disp('Vf1_d=');disp(Vf1_d);

```

```

61 disp('Vf1_e='); disp(Vf1_e);
62 disp('Vf1_f='); disp(Vf1_f);
63 disp('Vf1_g='); disp(Vf1_g);
64
65 printf('\n\n\n');
66 Vf2_d=-(Z2_bus(1,2)*If_e/3);
67 Vf2_e=-(Z2_bus(2,2)*If_e/3);
68 Vf2_f=-(Z2_bus(3,2)*If_e/3);
69 Vf2_g=-(Z2_bus(4,2)*If_e/3);
70 disp('Vf2_d='); disp(Vf2_d);
71 disp('Vf2_e='); disp(Vf2_e);
72 disp('Vf2_f='); disp(Vf2_f);
73 disp('Vf2_g='); disp(Vf2_g);
74
75 printf('\n\n\n');
76 Vf0_d=-(Z0_bus(1,2)*If_e/3);
77 Vf0_e=-(Z0_bus(2,2)*If_e/3);
78 Vf0_f=-(Z0_bus(3,2)*If_e/3);
79 Vf0_g=-(Z0_bus(4,2)*If_e/3);
80 disp('Vf0_d='); disp(Vf0_d);
81 disp('Vf0_e='); disp(Vf0_e);
82 disp('Vf0_f='); disp(Vf0_f);
83 disp('Vf0_g='); disp(Vf0_g);
84
85 printf('\n\n\n');
86 If1_fe=-Y1fe*(Vf1_f-Vf1_e); disp('If1_fe='); disp(
    If1_fe);
87 If1_de=-Y1de*(Vf1_d-Vf1_e); disp('If1_de='); disp(
    If1_de);
88 Ia1=If1_fe+If1_de; disp('Ia1='); disp(Ia1);
89
90 printf('\n\n\n');
91 If1_gf=-Y1gf*(Vf2_g-Vf2_f); disp('If1_gf='); disp(
    If1_gf);
92
93 printf('\n\n\n');
94 If2_fe=-Y1fe*(Vf2_f-Vf2_e); disp('If2_fe='); disp(
    If2_fe); //Y2fe=Y1fe

```

```

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

```

Scilab code Exa 11.7 Short Circuit Current Calculations

```

1 //Chapter 11
2 //Example 11.7
3 //page 423
4 //To find short circuit currents
5
6 clc;clear;
7 v_pf=1; //prefault voltage
8 a=0.5+0.8660254*%i;
9 //according to the fig.11.28 we can write Z-bus
    matrix for positive and negative phase sequence
10 printf('\nstep by step for finding Z1_bus\n')
11
12 //Bus1 to referance bus
13 Z1_bus=[0.15];
14 printf('Bus1 to reference\nZ1_bus=');disp(Z1_bus);
15
16 //Bus2 to Bus1
17 Z1_bus=[Z1_bus 0.15;0.15 0.15+0.2];
18 printf('\nBus2 to Bus1\nZ1_bus=');disp(Z1_bus);
19
20 //Bus2 to reference bus
21 Z1_bus=Z1_bus-(1/(Z1_bus(2,2)+0.15))*[Z1_bus
    (1:2,2:2)]*[Z1_bus(2:2,1:2)];
22 Z1_bus=(%i*Z1_bus);
23 Z2_bus=Z1_bus;
24 printf('\nBus2 to Reference\nZ1_bus=');disp(Z1_bus);

```

```

    printf( '\nZ2_bus=' ); disp( Z2_bus );
25
26 //according to the fig.11.29 we can write Z-bus
    matrix for zero phase sequence
27 printf( '\nstep by step for finding Z0_bus\n' )
28 //Bus1 to reference bus
29 Z0_bus=[0.05];
30 printf( '\nBus1 to reference \nZ0_bus=' ); disp( Z0_bus )
    ;
31
32 //Bus2 to Bus1
33 Z0_bus=[Z0_bus 0.05;0.05 0.05+0.4];
34 printf( '\nBus1 to Bus1 \nZ0_bus=' ); disp( Z0_bus );
35
36 //Bus2 to reference bus
37 Z0_bus=Z0_bus-(1/(Z0_bus(2,2)+0.05))*[Z0_bus
    (1:2,2:2)]*[Z0_bus(2:2,1:2)];
38 Z0_bus=(%i*Z0_bus);
39 printf( '\nBus2 to reference \nZ0_bus=' ); disp( Z0_bus )
    ;
40
41 //to find positive sequence of fault current
42 printf( '\n\n\nFault current calculation\n' )
43 If1_1=v_pf/(Z1_bus(1,1)+Z2_bus(1,1)+Z0_bus(1,1));
    printf( 'If1_1 = -j%0.5 f ', abs( imag( If1_1 ) ) );
44 printf( '\nFault current=If1=3If1_1=-j%0.1 f\n\n', abs(
    imag( 3*If1_1 ) ) );
45
46 Vf1_1=1-Z1_bus(1,1)*If1_1;
47 Vf1_2=1-Z1_bus(2,1)*If1_1;
48
49 Vf2_1=-Z2_bus(1,1)*If1_1;
50 Vf2_2=-Z2_bus(2,1)*If1_1;
51
52 Vf0_1=-Z0_bus(1,1)*If1_1;
53 Vf0_2=-Z0_bus(2,1)*If1_1;
54
55 If1_12=((%i*0.2)^-1)*(Vf1_1-Vf1_2);

```

```

56 If2_12=((%i*0.2)^-1)*(Vf2_1-Vf2_2);
57 If0_12=((%i*0.4)^-1)*(Vf0_1-Vf0_2);
58
59 If=[1 1 1;a^2 a 1;a a^2 1]*[If1_12;If2_12;If0_12];
60
61 printf('\n\nShort circuit current on the
    transmission line in all the three phases\n')
62 printf('\nIf_a_12=');
63 disp(If(1,1));
64
65 printf('\nIf_b_12=');
66 disp(If(2,1));
67
68 printf('\nIf_c_12=');
69 disp(If(3,1));
70
71 //short circuit current phase(a) of the generator
72 If1_G=((0.15*%i)^-1)*(1-Vf1_1)*(cosd(-30)+%i*sind
    (-30));
73 If2_G=((0.15*%i)^-1)*(0-Vf2_1)*(cosd(30)+%i*sind(30)
    );
74 If0_G=0;
75 printf('\n\nshort circuit current phase(a) of the
    generator\n')
76 Ifa_G=If1_G+If2_G+If0_G; printf('Ifa_G = -j%0.5 f',
    abs(imag(Ifa_G)));
77
78 //Voltage of the healthy phases of the bus 1.
79 printf('\n\nVoltage of the healthy phases of the
    bus 1\n')
80 Vf_b_1=Vf1_1*(cosd(240)+%i*sind(240))+Vf2_1*(cosd
    (120)+%i*sind(120))+Vf0_1; printf('Vf_b_1=%0.4 f -
    j%0.5 f',real(Vf_b_1),abs(imag(Vf_b_1)));
81 Vf_c_1=Vf1_1*(cosd(120)+%i*sind(120))+Vf2_1*(cosd
    (240)+%i*sind(240))+Vf0_1; printf('\nVf_c_1=%0.4 f
    + j%0.5 f',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

```
1 //Chapter 12
2 //Example 12.1
3 //page 439
4 //To find stored kinetic energy, rotor acceleration,
   change in torque angle and rotor speed
5 clear;clc;
6 G=100; //base machine rating
7 H=8.0; //inertia constant
8 P=4; //no of poles
9 //(a)To find stored energy in rotor at synchronous
   speed
10 stored_energy=G*H;
11 printf('\nStored energy = %d MJ',stored_energy);
12
13 //(b)To find rotor acceleration when mechanical
   input is raised 80MW for an electrical load of 50
   MW
14 Pa=30; //nett power
15 f=50; //frequency
16 M=stored_energy/(180*f);
```

```

17 alpha=Pa/M; //rotor acceleration
18 printf('\n\nRotor acceleration = %0.1f elect deg/s^2
      ',alpha);
19
20 //(c)To calculate change in torque angle and rotor
      speed when the above acceleration is maintained
      for 10 cycles
21 change_angle=0.5*alpha*(10*20*10^(-3));
22 printf('\n\nChange in torque angle = %0.2f elect
      degrees ',change_angle);
23 change_angle=60*alpha/(2*360);
24 printf('\n\nChange in torque angle = %0.3f rpm/s',
      change_angle);
25 speed=(120*f/P)+(change_angle*0.2);
26 printf('\n\nRoor speed at the end of 10 cycles = %0
      .3f rpm ',speed);

```

Scilab code Exa 12.2 steady state power limit

```

1 //Chapter 12
2 //Example 12.1
3 //page 448
4 //To calculate steady state power limit
5 clear;clc;
6
7 Xdg=1%i; //generator's
8 Xdm=1%i; //motor's
9 Xt=0.1%i; //transformers
10 Xl=0.25%i; //transmission line's
11 Xc=-1%i; //static capacitor's
12 Xi=1%i; //inductive reactor
13 Eg=1.2; //generator's internal voltage
14 Em=1; //motor's internal voltage

```

```

15
16 //case(i) steady state power limit without reactor
17 P1=(abs(Eg)*abs(Em))/(abs(Xdg+Xt+Xl+Xt+Xdm));
18 printf('\n\n Steady state power limit without
    reactor = %0.5f pu',P1);
19
20 //case(ii) steady state power limit with capacitive
    reactor
21 //three arms of star connected reactances are
22 Xa=Xdg+Xt+Xl; //from generator side
23 Xb=Xdm+Xt; //from load side
24 Xc=Xc; //from reactor side
25
26 //converting star to delta
27 //reactance between generator side to load side is
28 Xab=(Xa*Xb+Xb*Xc+Xc*Xa)/Xc;
29 //power limit is
30 P2=(abs(Eg)*abs(Em))/(abs(Xab));
31 printf('\n\n Steady state power limit with
    capacitive reactor = %0.5f pu',P2);
32
33 //case(iii) steady state power limit with inductive
    reactor
34 //three arms of star connected reactances are
35 Xa=Xdg+Xt+Xl; //from generator side
36 Xb=Xdm+Xt; //from load side
37 Xc=Xi; //from reactor side
38
39 //converting star to delta
40 //reactance between generator side to load side is
41 Xab=(Xa*Xb+Xb*Xc+Xc*Xa)/Xc;
42 //power limit is
43 P3=(abs(Eg)*abs(Em))/(abs(Xab));
44 printf('\n\n Steady state power limit with inductive
    reactor = %0.5f pu',P3);

```

Scilab code Exa 12.3 Maximum Power Transferred

```
1 //Chapter 12
2 //Example 12.3
3 //page 450
4 //To calculate maximum power transferred
5 clear;clc;
6
7 Vt=1.0; //generator terminal voltage
8 V=1.0 ; //infinite bus voltage
9 Pe=1.0 ; //power delivered
10 Xd=0.25*%i ; //generator's transient reactance
11 Xl=0.5*%i ; //transmission line's reactance
12 Xt=0.1*%i; //transformer's reactance
13
14 //to calculate alpha
15 alpha=asind(Pe*abs(Xt+Xl/2)/(abs(Vt)*abs(V)));
16 printf('\n\nAlpha=%0.1f deg',alpha);
17
18 //current to infinite bus
19 I=(Vt*(cosd(alpha)+%i*sind(alpha))-V)/(Xt+Xl/2);
20 printf('\nCurrent to infinite bus=%d+j%0.3f pu',real(I),imag(I));
21
22 //voltage behind transient reactance
23 E=Vt+I*(Xd+Xt+Xl/2);
24 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)));
25
26 delta=0:0.001:180;
27
```

```

28 //case(a) Maximum power when system is healthy
29 X12=Xd+Xt+Xl/2;
30 Pmax=abs(V)*abs(E)/abs(X12);
31 Pe1=Pmax*sind(delta);
32 printf('Maximum power that can be transferred under
    the following condition is')
33 printf('\n\n(a)System Healthy:');
34 printf('\nPmax=%0.2f pu',Pmax);
35 printf('\nPe=%0.2f sin(delta) pu',Pmax);
36
37 //case(b) One line short in the middle
38 //converting bus3 to delta40
39 Xa=Xd+Xt; //generator side
40 Xb=Xl; //healthy transmission line side
41 Xc=Xl/2; //unhealthy line side
42 X12=(Xa*Xb+Xb*Xc+Xc*Xa)/(Xc);
43 Pmax=abs(V)*abs(E)/abs(X12);
44 Pe2=Pmax*sind(delta);
45 printf('\n\n(b)One line shorted in the middle:');
46 printf('\nPmax=%0.4f pu',Pmax);
47 printf('\nPe=%0.4f sin(delta) pu',Pmax);
48
49 //case(c) One line open
50 X12=Xd+Xt+Xl;
51 Pmax=abs(V)*abs(E)/abs(X12);
52 Pe3=Pmax*sind(delta);
53 printf('\n\n(c)One line open :');
54 printf('\nPmax=%0.4f pu',Pmax);
55 printf('\nPe=%0.4f sin(delta) pu',Pmax);
56
57 //plotting Power angle curves
58 plot(delta,Pe1,delta,Pe2,delta,Pe3);
59 legend(['1.79 sin(delta)';'0.694 sin(delta)';'1.265 sin
    (delta)']);
60 title("Power angle curves");
61 xlabel("Delta");
62 ylabel("Pe");

```

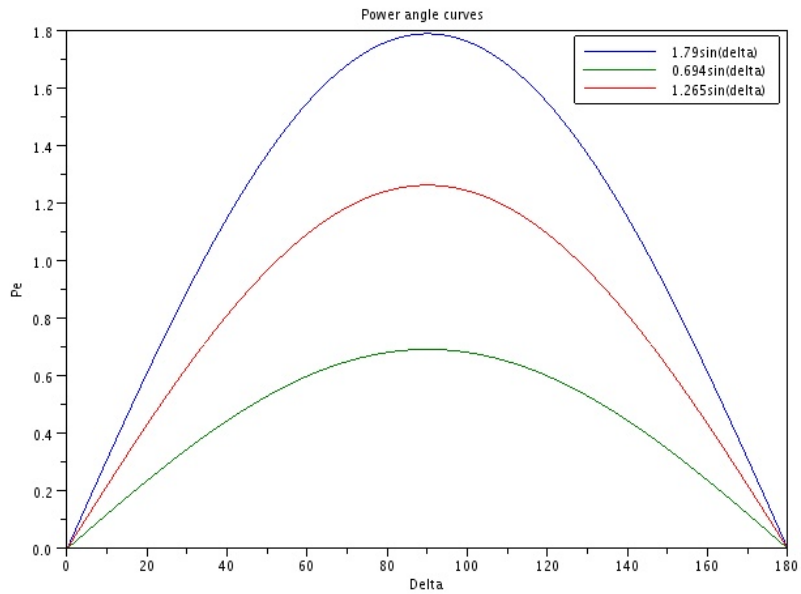


Figure 12.1: Maximum Power Transferred

Scilab code Exa 12.4 Acceleration and Rotor angle

```

1 //Chapter 12
2 //Example 12.4
3 //page 453
4 //To calculate acceleration and rotor angle
5 clear;clc;
6
7 delta0=33.9; //initial rotor angle

```

```

8 H=4; //inertia constant
9 f=50; //frequency
10 Pm=1; //mechanical power input
11 t=0.05; //time interval
12 angular_acceleration=(Pm-0.694*sind(delta0))*180*f/H
    ;
13 delta_change=0.5*angular_acceleration*t^2;
14 delta_new=delta0+delta_change;
15 new_angular_acceleration=(Pm-0.694*sind(delta_new))
    *180*f/H;
16
17 printf('\n\nInitial rotor angular acceleration = %d
    elect deg/s^2',angular_acceleration);
18 printf('\nDelta_change=%0.1f deg',delta_change);
19 printf('\nNew delta =delta1=%0.1f deg',delta_new);
20 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

```

1 //Chapter 12
2 //Example 12.5
3 //page 456
4 //To calculate frequency of natural oscilations
5 clear;clc;
6
7 E=1.2; //no load voltage
8 V=1; //infinite bus voltage
9 Xg=1.2; // synchronous generator reactance
10 Xt1=0.6 //transformer anf transmission line
    reactance
11 H=4; //inertia constant
12

```



```

13 //case(i) 50% loading
14 delta0=asind(0.5);
15 synchronizing_coefficien=(abs(E)*abs(V)*cosd(delta0)
    )/(Xg+Xt1);
16 M=H/(%pi*50);
17 p=%i*sqrt(synchronizing_coefficien/M);
18 f=abs(p)/(2*%pi);
19 printf('\n\ncase(i) For 50%% loading ');
20 printf('\nDelta_0=%d deg',delta0);
21 printf('\nsynchronizing_coefficient=%0.3f MW(pu)/
    elect rad',synchronizing_coefficien);
22 printf('\nM=%0.4f s^2/elect rad',M);
23 printf('\nFrequency of oscillations=%0.2f rad/sec =
    %0.3f Hz\n',abs(p),f);
24
25 //case(i) 80% loading
26 delta0=asind(0.8);
27 synchronizing_coefficien=(abs(E)*abs(V)*cosd(delta0)
    )/(Xg+Xt1);
28 M=H/(%pi*50);
29 p=%i*sqrt(synchronizing_coefficien/M);
30 f=abs(p)/(2*%pi);
31 printf('\n\ncase(ii) For 80%% loading ');
32 printf('\nDelta_0=%d deg',delta0);
33 printf('\nSynchronizing_coefficient=%0.3f MW(pu)/
    elect rad',synchronizing_coefficien);
34 printf('\nM=%0.4f s^2/elect rad',M);
35 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

```
1 //Chapter 12
```

```

2 //Example 12.6
3 //page 457
4 //To find steady state power limit
5 clear;clc;
6
7 V=1.0; //infinite bus volatge
8 Vt=1.2; //terminal volatge
9 Xd=0.5*i; //synchronous generator reactance
10 X=i; //series reactance
11 //by solving the expressions given in the textbook
12 theta=acosd(0.5/1.8);
13 printf('\n\theta=%0.3f deg',theta);
14 Vt=Vt*(cosd(theta)+i*sind(theta));
15 printf('\nVt=%0.3f+j%0.3f pu',real(Vt),imag(Vt));
16 I=(Vt-V)/X;
17 printf('\nI=%0.3f+j%0.3f pu',real(I),imag(I));
18 E=Vt+Xd*I;
19 printf('\nE=%0.3f @ %d deg pu',abs(E),atand(imag(E)/
    real(E)));
20 Pmax=(abs(E)*abs(V))/abs(X+Xd);
21 printf('\n\nSteady state power limit is given by:\n
    tPmax=%0.3f pu',Pmax);
22 E=1.2;Pmax=(abs(E)*abs(V))/abs(X+Xd);
23 printf('\n\nIf the generator emf is held fixed at a
    value 1.2pu,steady state power limit would be :\n
    Pmax=%0.2f pu\n\n',Pmax);

```

Scilab code Exa 12.7 Critical Clearing Angle

```

1 //Chapter 12
2 //Example 12.7
3 //page 475
4 //To calculate critcal clearing angle

```

```

5 clear;clc;
6
7 Xd=0.25; //direct axis transient reactance of the
    generator
8 Xl1=0.5; Xl2=0.4; //reactances of transmission line
9 E=1.2; //voltage behind transient reactance
10 Xinf=0.05; //reactnce before infinite bus
11 V=1; //infinite bus voltage
12 Pm=1; //mechanical input to the generator
13 delta=0:1:180;
14
15 //Normal operation (prefault)
16 X1=Xd+(Xl1*Xl2/(Xl1+Xl2))+Xinf; //equivalent
    reactance between sending ened and receiving end
17 //Power angle equation before the fault is
18 Pe1=(E*V/X1)*sind(delta);
19 //prefault operating power =1.0pu
20 delta0=asin(1/max(Pe1));
21 printf('Normal Operation (prefault):\n');
22 printf('X1=%0.3f PU\n',X1);
23 printf('Pe1=%0.1 fsin(delta)\n\n',max(Pe1));
24
25 //during fault there will be no power transfer
26 Pe2=0;
27 printf('During Fault:\n');
28 printf('Pe2=%d\n\n',Pe2);
29
30 //Post fault operation(fault cleared by opening the
    faulted line)
31 X3=Xd+Xl1+Xinf;
32 Pe3=(E*V/X3)*sind(delta);
33 delta_max=%pi-asin(Pm/max(Pe3));
34 //from A1 and A2, we solve A1=A2
35 deff(' [y]=fx(delta_cr)', 'y=1.5*cos(delta_cr)+
    delta_cr-1.293-Pm*(delta_cr-delta0)');
36 delta_cr=fsolve(0.45,fx);
37 printf('Post fault operation(fault cleared by
    opening the faulted line):\n');

```

```

38 printf('X3=%0.1fPU\n',X3);
39 printf('Pe3=%0.1fsin(delta)\n',max(Pe3));
40 printf('Delta_cr=%0.4f rad =%0.2f deg',delta_cr,
        delta_cr*180/%pi);
41 plot(delta,Pe1,delta,Pe3,delta,Pm);
42 legend('Pe1=2.3 sin(delta)', 'Pe3=1.5 sin(delta)', 'Pm=1
        ');
43 title('Power angle Diagram for example 12.7');
44 xlabel('delta (in degrees)----->');
45 ylabel('Electrical output (Pe)----->');

```

Scilab code Exa 12.8 Critical Clearing Angle 2

```

1 //Chapter 12
2 //Example 12.8
3 //page 477
4 //To calculate critical clearing angle
5 clear;clc;
6
7 Xd=0.25; //direct axis transient reactance of the
        generator
8 Xl1=0.28; Xl2_1=0.14;Xl2_2=0.14; //reactances of
        transmission line
9 E=1.2; //voltage behind transient reactance
10 Xinf=0.17; //reactnce before infinite bus
11 V=1; //infinite bus voltage
12 Pm=1; //mechanical input to the generator
13 Xtr=0.15; //transformer reactance
14 delta=0:1:180;
15
16 //prefault operation

```

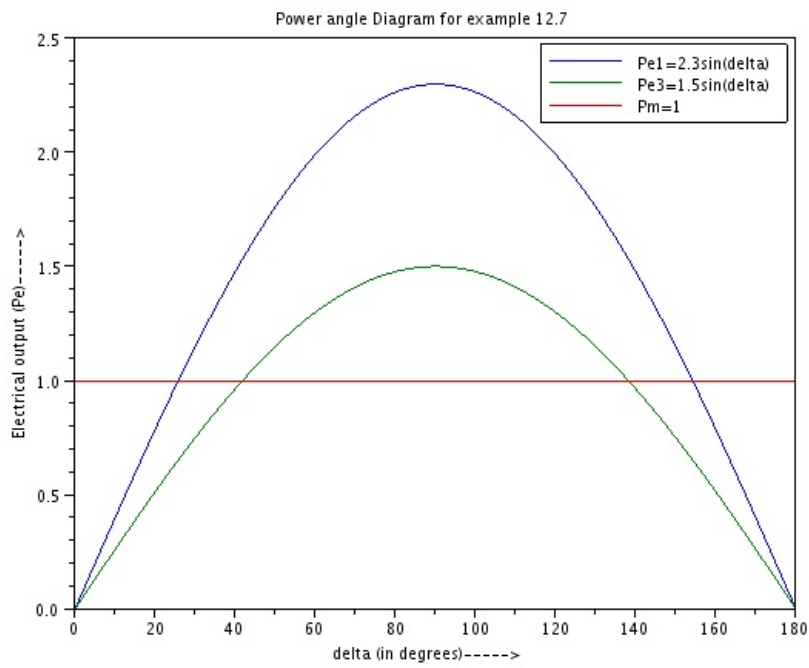


Figure 12.2: Critical Clearing Angle

```

17 X1=Xd+Xinf+(Xtr+Xl1+Xtr)/2; //transfer reactance
    between generator and infinte bus
18 Pe1=E*V*sind(delta)/X1;
19 delta0=asin(1/max(Pe1));
20 printf('Normal Operation (prefault):\n');
21 printf('X1=%0.3f PU\n',X1);
22 printf('Pe1=%0.2 fsin(delta)\n',max(Pe1));
23 printf('delta0=%0.3fPU\n\n',delta0);
24 //during fault there will be no power transfer
25 //using star delta transformation given in the
    textbook
26 X2=2.424;
27 Pe2=E*V*sind(delta)/X2;
28 printf('During Fault:\n');
29 printf('X2=%0.3f PU\n',X2);
30 printf('Pe2=%0.3 fsin(delta)\n\n',max(Pe2));
31
32 //Post fault operation(faulty line switched off)
33 X3=Xd+Xinf+(Xtr+Xl1+Xtr);
34 Pe3=E*V*sind(delta)/X3;
35 delta_max=%pi-asin(Pm/max(Pe3));
36 //from A1 and A2, we solve A1=A2
37 deff(' [y]=fx(delta_cr)', 'y=-delta0+max(Pe2)*cos(
    delta_cr)-0.399-0.661-max(Pe3)*cos(delta_cr)+
    delta_max");
38 delta_cr=fsolve(0.45,fx);
39 printf('Post fault operation(faulty line switched
    off):\n');
40 printf('X3=%0.1fPU\n',X3);
41 printf('Pe3=%0.1 fsin(delta)\n',max(Pe3));
42 printf('Delta_cr=%0.4f rad =%0.2f deg',delta_cr,
    delta_cr*180/%pi);
43 plot(delta,Pe1,delta,Pe2,delta,Pe3,delta,Pm);
44 legend('Pe1=1.69 sin(delta)', 'Pe2=0.495 sin(delta)', '
    Pe3=1.2 sin(delta)', 'Pm=1');
45 title('Power angle Diagram for example 12.8');
46 xlabel('delta (in degrees)----->');
47 ylabel('Electrical output (Pe)----->');

```

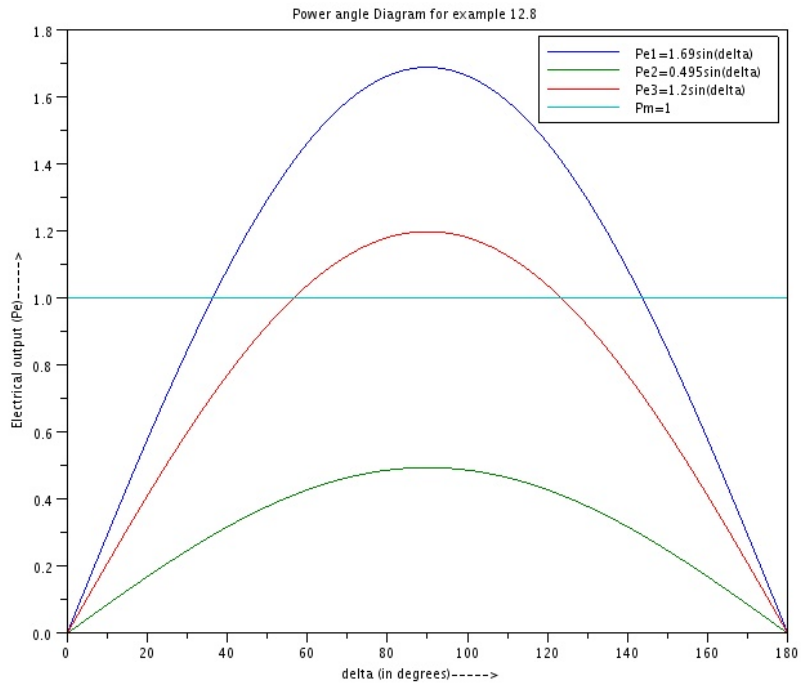


Figure 12.3: Critical Clearing Angle 2

```

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

```

Scilab code Exa 12.9 Critical Clearing Angle 3

```

1 //Chapter 12

```

```

2 //Example 12.9
3 //page 479
4 //To calculate critical clearing angle
5 clear;clc;
6 Pmax1=2; // pre-fault(2 lines)
7 Pmax2=0.5; //during fault
8 Pmax3=1.5; //post fault(1 line)
9 Pm=1; //initial loading
10
11 delta0=asin(Pm/Pmax1);
12 delta_max=%pi-asin(Pm/Pmax3);
13
14 //to find critical angle, using eq.12.67
15 delta_cr=acos((Pm*(delta_max-delta0)-Pmax2*cos(
    delta0)+Pmax3*cos(delta_max))/(Pmax3-Pmax2));
16 printf('Pmax1=%0.1f PU\t Pmax2=%0.2f PU\t Pmax3=%0.2
    f PU\n\n',Pmax1,Pmax2,Pmax3);
17 printf('Delta0=%0.3f rad\n\n',delta0);
18 printf('Delta_max=%0.3f rad\n\n',delta_max);
19 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

```

1 //Chapter 12
2 //Example 12.10
3 //page 482
4 //To plot swing curves for sustained fault and fault
    cleared at 2.5 and 6.25 cycles
5 clear;clc;
6 P_delivered=18;
7 MVA_base=20;

```



```

8 Xd=0.35;E=1.1;
9 Xl=0.2;
10 V=1;
11
12 H=2.52;
13 f=50;
14 M=H/(180*f);
15
16 //////////Prefault////////////////////
17 X1=Xd+Xl/2;
18 delta=0:0.1:180;
19 Pe1=E*V*sind(delta)/X1;
20 P_initial=P_delivered/MVA_base;Pm=P_initial;
21 delta0=asind(P_initial/max(Pe1));
22
23
24 //////////during fault////////
25 X2=1.25; //from delta to star conversion
26 Pe2=E*V*sind(delta)/X2;
27
28 //////////postfault:with faulted line switched off
   //////////
29 X3=Xd+Xl;
30 Pe3=E*V*sind(delta)/X3;
31
32 Pa_0minus=0;
33 Pa_0plus=Pm-max(Pe2)*sind(delta0);
34 Pa_avg=(Pa_0minus+Pa_0plus)/2;
35
36
37 //////////for a sustained fault////////
38 P_max=max(Pe2);
39 delta_delta=0; //initially
40 delta=21.64; //initially
41 delta_old=21.64;
42 delta_t=0.05;
43 z1=21.64
44 n=10;

```

```

45 T=0;
46 printf('Point-by-point calculation of swing curve
    for sustained fault delta_t=0.05sec\n');
47 printf('
    -----
    \n');
48 printf('t\t\tPmax\t\t sin(delta)\t\tPa\t\t y\t\t
    tdelta\n');
49 printf('
    -----
    \n');
50 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);
51 for i=1:n
52     t=i*delta_t;
53     if i==1 then
54         Pa=(0.9-P_max*sind(delta_old))/2;
55     else
56         Pa=0.9-P_max*sind(delta_old);
57     end
58
59     y=(delta_t^2)*Pa/M;
60     delta_delta=delta_delta+y;
61     delta=delta+delta_delta;
62     z1=[z1,delta];T=[T,t];
63     printf('%0.3f sec\t%0.3f\t\t %0.3f\t\t\t%0.3f\t\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);
64     delta_old=delta;
65 end
66
67
68 // Fault cleared in 2.5 cycles(time to clear
    fault=0.05sec)/////
69

```

```

70 P_max1=max(Pe2);
71 P_max2=max(Pe3);
72 delta_delta=0; //initially
73 delta=21.64; //initially
74 delta_old=21.64;
75 delta_t=0.05;
76 z2=21.64
77 n=10;
78 T=0;
79 printf('\n\nComputations of swing curves for fault
      cleared at 2.5 cycles(0.05 sec)\n');
80 printf('
      -----
      \n');
81 printf('t\t\tPmax\t\t sin(delta)\t\tPa\t\t y\t\t
      tdelta\n');
82 printf('
      -----
      \n');
83 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);
84 for i=1:n
85     t=i*delta_t;
86     if i==1 then
87         Pa=(0.9-P_max*sind(delta_old))/2;
88         P_max=P_max1;
89     elseif i==2 then
90         Pa=((0.9-P_max2*sind(delta_old))+((0.9-
          P_max1*sind(delta_old))))/2;
91         P_max=P_max2;
92     else
93         Pa=0.9-P_max2*sind(delta_old);
94         P_max=P_max2;
95     end
96
97     y=(delta_t^2)*Pa/M;

```

```

98     delta_delta=delta_delta+y;
99     delta=delta+delta_delta;
100    z2=[z2,delta];T=[T,t];
101
102    if i==1 then
103        delta_old=delta;
104        printf('%.3f sec\t%.3f\t\t %.3f\t\t\t%.3f\t\t\t%.3f\t\t\t%.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);
105    else
106        printf('%.3f sec\t%.3f\t\t %.3f\t\t\t%.3f\t\t\t%.3f\t\t\t%.3f\n',t,P_max,sind(delta)
            ),0.9-P_max*sind(delta),8.929*(0.9-P_max*
            sind(delta)),delta);
107        delta_old=delta;
108    end
109
110 end
111
112
113
114 // Fault cleared in 6.25 cycles(time to clear
    fault=0.125sec)/////
115
116 P_max1=max(Pe2);
117 P_max2=max(Pe3);
118 P_max=P_max1;
119 delta_delta=0; //initially
120 delta=21.64; //initially
121 delta_old=21.64;
122 delta_t=0.05;
123 z3=21.64
124 n=10;
125 T=0;
126 printf('\n\nComputations of swing curves for fault

```

```

        cleared at 6.25 cycles(0.125sec)\n');
127 printf('
        -----
        \n');
128 printf('t\t\tPmax\t\t sin(delta)\t\tPa\t\t y\t\t
        tdelta\n');
129 printf('
        -----
        \n');
130 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);
131 for i=1:n
132     t=i*delta_t;
133     if i==1 then
134         Pa=(0.9-P_max1*sind(delta_old))/2;
135         P_max=P_max1;
136     elseif i==2 then
137         Pa=(0.9-P_max1*sind(delta_old));
138         P_max=P_max1;
139     elseif i==3 then
140         Pa=(0.9-P_max1*sind(delta_old));
141         P_max=P_max2;
142     else
143         Pa=0.9-P_max2*sind(delta_old);
144         P_max=P_max2;
145     end
146
147     y=(delta_t^2)*Pa/M;
148     delta_delta=delta_delta+y;
149     delta=delta+delta_delta;
150     z3=[z3,delta];
151     T=[T,t];
152     printf('%0.3f sec\t%0.3f\t\t %0.3f\t\t\t%0.3f\t\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);

```

```

153     delta_old=delta;
154
155 end
156
157 plot(T,z1,T,z2,T,z3);
158 set(gca(),"grid",[1 1]);
159 legend('Sustained Fault','Fault cleared at 2.5
        cycles','Fault cleared at 6.25 cycles',[,2]);
160
161 title('Swing Curves for Example 12.10 for a
        sustained fault and for clearing in 2.5 and 6.25
        cycles','fontsize',2.4);
162 xlabel('Time (in seconds)----->');
163 ylabel('Torque Angle (delta ,deg)----->');
164 f=get("current_figure")
165 f.figure_position=[0,15]
166 f.figure_size=[645,1000]

```

Scilab code Exa 12.11 Swing Curves For Multimachines

```

1 //Chapter 12
2 //Example 12.11
3 //page 488
4 //To plot swing curves for fault cleared at 0.275s
   and 0.08s of a multimachine system
5 clear;clc;
6
7 xd1=%i*0.067;xd2=%i*0.1;
8
9 //primitive admittances of the lines
10 y45=1/(0.018+%i*0.11);    B45=%i*0.113;

```

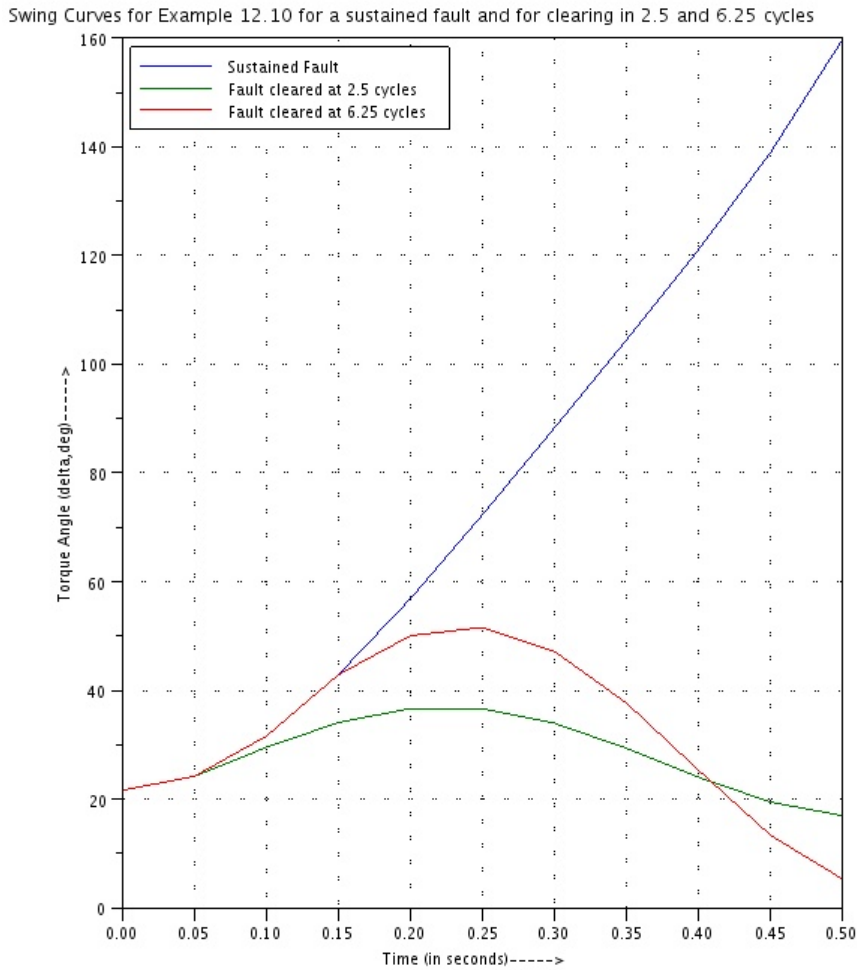


Figure 12.4: Swing Curves For Sustained Fault and Cleared Fault at the Specified Time

```

11 y51=1/(0.004+%i*0.0235);    B51=%i*0.098;
12 y41=1/(0.007+%i*0.04);    B41=%i*0.041;
13 z24=(%i*0.022);
14 z35=(%i*0.04);
15
16 //Bus data and pre-fault load-flow values in PU
17 V1=1.0;                      P1=-3.8083;    Q1
    =-0.2799;    P11=0;    Q12=0;
18 V2=1.0194+%i*0.1475;    P2=3.25;    Q2=0.6986;
    P12=0;    Q12=0;
19 V3=1.0121+%i*0.1271;    P3=2.10;    Q3=0.3110;
    P13=0;    Q13=0;
20 V4=1.0146+%i*0.0767;    P4=0;    Q4=1.0;
    P14=1.0;    Q14=0.44;
21 V5=1.0102+%i*0.0439;    P5=0;    Q5=0;
    P15=0.5;    Q15=0.16;
22
23
24 // To find voltage behind transient reactances
    before the occurrence of fault
25
26 I2=(P2-%i*Q2)/(V2');
27 E2=V2+I2*xd1;
28 E1=V1;
29 I3=(P3-%i*Q3)/(V3');
30 E3=V3+I3*xd2;
31
32 //converting loads into their admittances
33 Y14=(P14-%i*Q14)/(V4*V4');
34 Y15=(P15-%i*Q15)/(V5*V5');
35
36 //forming augmented Bus admittance matrix before the
    occurrence of fault
37 Y11=y41+y51; Y12=0; Y13=0; Y14=-y41; Y15=-y51;
38 Y21=Y12; Y22=1/(xd1+z24); Y23=0; Y24=-(1/(z24+xd1)); Y25
    =0;
39 Y31=0; Y32=0; Y33=1/(z35+xd2); Y34=0; Y35=-1/(z35+xd2);
40 Y41=Y14; Y42=Y24; Y43=Y34; Y44=y41+Y14+y45+B45+B41-Y24;

```



```

        Y45=-y45;
41  Y51=Y15;Y52=Y25;Y53=Y35;Y54=Y45;Y55=Y15+y45+y51+B45+
        B51-Y35;
42
43  Ybus=[Y11 Y12 Y13 Y14 Y15;
44         Y21 Y22 Y23 Y24 Y25;
45         Y31 Y32 Y33 Y34 Y35;
46         Y41 Y42 Y43 Y44 Y45;
47         Y51 Y52 Y53 Y54 Y55];
48
49  printf('\n Augmented prefault bus admittance matrix
        (in PU) is given by\n\n Ybus=\n');
50  disp(Ybus);
51  /////to find the Ybus during fault
52  Ybus_1=Ybus([1:3,5],[1:3,5]);
53  n=4
54  for k=1:n-1
55      for j=1:n-1
56          Ybus_during_fault(k,j)=Ybus_1(k,j)-(Ybus_1(k
                ,n)*Ybus_1(n,j))/Ybus_1(n,n);
57      end
58  end
59  printf('\n\n\n Bus admittance matrix during fault (
        in PU) is given by\n\n Ybus_during_fault=\n');
60  disp(Ybus_during_fault);
61
62  //to find Ybus after the fault has been cleared
63  Y45=0;Y54=0;Y44=Y44-y45-B45;Y55=Y55-y45-B45;
64  Ybus_2=[Y11 Y12 Y13 Y14 Y15;
65         Y21 Y22 Y23 Y24 Y25;
66         Y31 Y32 Y33 Y34 Y35;
67         Y41 Y42 Y43 Y44 Y45;
68         Y51 Y52 Y53 Y54 Y55];
69
70  //eliminating node 5 from Ybus_2
71  n=5
72  for k=1:n-1
73      for j=1:n-1

```

```

74         Ybus_3(k,j)=Ybus_2(k,j)-(Ybus_2(k,n)*Ybus_2(
           n,j))/Ybus_2(n,n);
75     end
76 end
77
78 //eliminating node 4 to get post fault Ybus
79 n=4
80 for k=1:n-1
81     for j=1:n-1
82         Ybus_post_fault(k,j)=Ybus_3(k,j)-(Ybus_3(k,n)
           *Ybus_3(n,j))/Ybus_3(n,n);
83     end
84 end
85 printf('\n\n\n Bus admittance matrix postfault (in
           PU) is given by\n\n Ybus_post_fault=\n');
86 disp(Ybus_post_fault);
87 printf('\n\n\n');
88 //During fault power angle equation
89 delta3=0:0.1:180;
90 Pe2f=0;
91 Pe3f=(abs(E3'))^2*real(Ybus_during_fault(3,3))+abs(E1
           ')*abs(E3')*abs(Ybus_during_fault(3,1))*cosd(
           delta3-atan2(imag(Ybus_during_fault(1,3))/real(
           Ybus_during_fault(1,3))));
92
93 //Postfault power angle equations
94 delta2=0:0.1:180;
95 Pe2pf=(abs(E2'))^2*real(Ybus_post_fault(2,2))+abs(E1
           ')*abs(E2')*abs(Ybus_post_fault(2,1))*cosd(delta2
           -atan2(imag(Ybus_post_fault(1,2))/real(
           Ybus_post_fault(1,2))));
96 Pe3pf=(abs(E3'))^2*real(Ybus_post_fault(3,3))+abs(E1
           ')*abs(E3')*abs(Ybus_post_fault(3,1))*cosd(delta3
           -atan2(imag(Ybus_post_fault(1,3))/real(
           Ybus_post_fault(1,3))));
97
98 //mechanical inputs which are assumed to be constant
           are given by

```

```

99 Pm2=max(real(E2*I2'));
100 Pm3=max(real(E3*I3'));
101
102 //xdot function defining the swing equations of each
    of the machines
103 function xdot=mac2(t,x,tc)
104     xdot(1)=x(2);
105     if t>tc then
106         xdot(2)=180*50*(Pm2-(0.6012+8.365*sind(x(1)
            -1.662)))/12; //swing equation after
            clearing the fault
107     else
108         xdot(2)=180*50*(Pm2-(0))/12; //swing
            equation before clearing the fault
109     end
110
111 endfunction
112
113 function xdot=mac3(t,x,tc)
114     xdot(1)=x(2);
115     if t>tc then
116         xdot(2)=180*50*(Pm3-(0.1823+6.5282*sind(x(1)
            -0.8466)))/9; //swing equation after
            clearing the fault
117     else
118         xdot(2)=180*50*(Pm3-(0.1561+5.531*sind(x(1)
            -0.755)))/9; //swing equation before
            clearing the fault
119     end
120
121 endfunction
122
123 //to find the solution of swing equation to draw the
    swing curves
124
125 //to draw the swing curves for machines 2 and 3 for
    example12.11 for clearing at 0.275 sec
126 subplot(2,1,1)

```

```

127 x_1_0=[19.354398,0]';t0=0; T=0:0.01:1;T=T';
128 x_2_0=[18.2459,0]';tc=0.275;
129 sol1=ode(x_1_0,t0,T,mac2);
130 sol2=ode(x_2_0,t0,T,mac3);
131
132 plot(T(1:20),sol1(1,1:20)',T,sol2(1,:)');
133 set(gca(),"grid",[1 1]);
134 legend('Machine 2','Machine 3',[,1]);
135 title('Swing Curves for machines 2 and 3 of Example
12.11 for a clearing at '+string(tc)+' s');
136 xstring(0.55,59,'Machine 1 is reference (infinte bus
)');
137 xlabel('Time (in seconds)----->');
138 ylabel('Torque Angle (delta ,deg)----->');
139
140
141 //to draw the swing curves for machines 2 and 3 for
example12.11 for clearing at 0.08 sec
142 subplot(2,1,2)
143 x_1_0=[19.354398,0]';t0=0; T=0:0.01:1;T=T';
144 x_2_0=[18.2459,0]';tc=0.08;
145 sol1=ode(x_1_0,t0,T,mac2);
146 sol2=ode(x_2_0,t0,T,mac3);
147
148 plot(T,sol1(1,:) ',T,sol2(1,:) ');
149 set(gca(),"grid",[1 1]);
150 legend('Machine 2','Machine 3',[,4]);
151 title('Swing Curves for machines 2 and 3 of Example
12.11 for a clearing at '+string(tc)+' s');
152 xstring(0.44,43,'Machine 1 is reference (infinte bus
)');
153 xlabel('Time (in seconds)----->');
154 ylabel('Torque Angle (delta ,deg)----->');
155
156 f=get("current_figure");
157 f.figure_position=[0,15];
158 f.figure_size=[565,1000];

```

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

```
1 //Chapter 12
2 //Example 12.12
3 //page 500
4 //To plot swing curves for single pole and three
   pole switching
5 clear;clc;
6
7 Xg0=0.1;Xg1=0.3;Xg2=0.15;E=1.2;H=4.167;
8 Xt=0.1;
9 Xl0=1.0;Xl1=0.3;Xl2=0.3;V=1;
10
11 //transfer reactance during LG fault(fault not
   cleared) by star delta transformation is given by
12 X12_fault=1.45;
13
14 //transfer reactance after LG faulted line open is
   given by
15 X12_fault_open=1.22;
16
17 //transfer reactance when all the lines are healthy
   is given by
18 X12_healthy=0.8;
19
20 //power angle equations
21 delta=0:0.1:180;
22
23 //Prefault condition
```

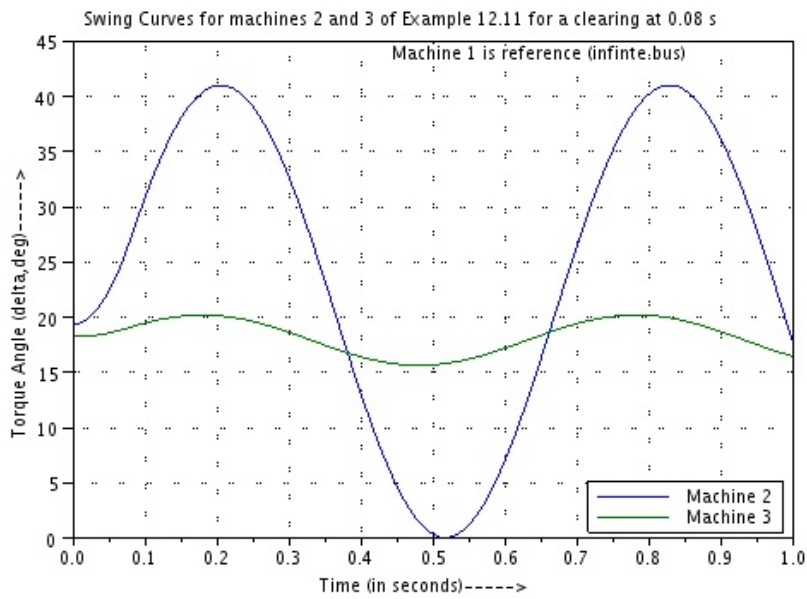
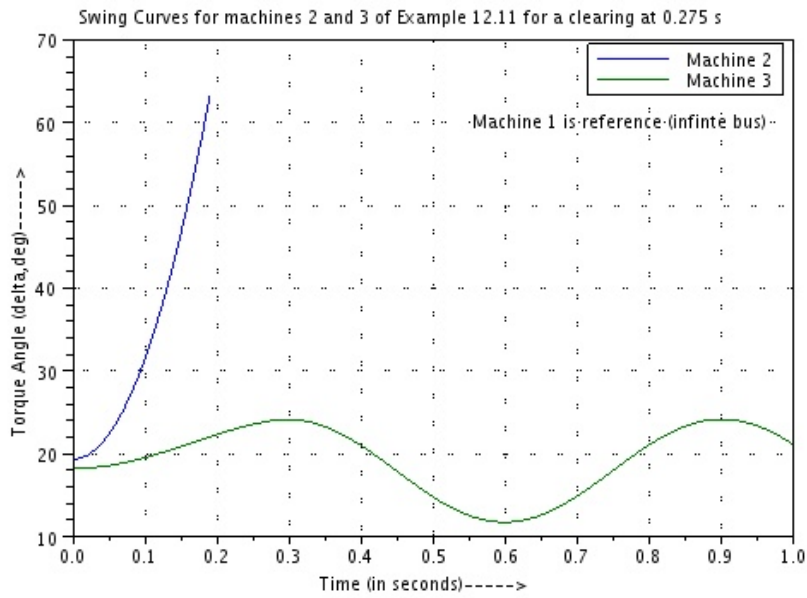


Figure 12.5: Swing Curves For Multimachines

```

24 Pe1=(E*V)*sind(delta)/X12_healthy;
25 //for an initial load of 1PU
26 delta0=asind(1/1.5);
27
28 //during fault
29 Pe2=(E*V)*sind(delta)/X12_fault;
30
31 //during single pole switching
32 Pe3=(E*V)*sind(delta)/X12_fault_open;
33
34 //during three pole switching
35 Pe4=0;
36
37 //after reclosure
38 Pe5=Pe1;
39
40 Pm=1.0;
41
42 //xdot function defining the swing equations of
   machine during single poling
43 function xdot=mac_1_pole(t,x,tc,tr)
44     xdot(1)=x(2);
45     if (t<=tc) then
46         xdot(2)=180*50*(Pm-(0.827*sind(x(1))))/12; //
           swing equation before clearing the faulted
           line
47     elseif (t>tc)&(t<tr) then
48         xdot(2)=180*50*(Pm-(0.985*sind(x(1))))/12;//
           swing equation during single pole switching
49     elseif (t>=tr) then
50         xdot(2)=180*50*(Pm-(1.5*sind(x(1))))/12; //
           after reclosure
51     end
52 endfunction
53
54 //xdot function defining the swing equations of
   machine during three poling
55 function xdot=mac_3_pole(t,x,tc,tr)

```

```

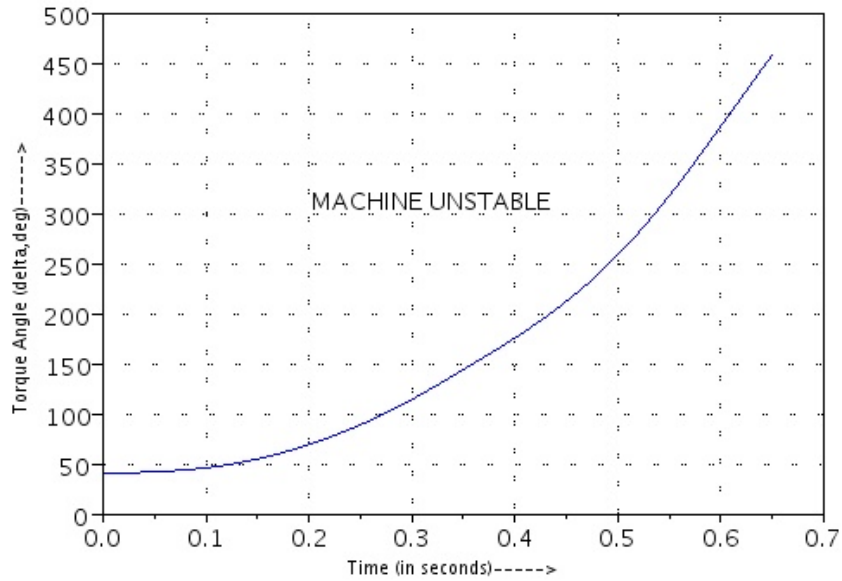
56     xdot(1)=x(2);
57     if (t>tc)&(t<tr) then
58         xdot(2)=180*50*(Pm-0)/4.167; //swing equation
           during three pole switching
59     elseif (t<=tc) then
60         xdot(2)=180*50*(Pm-(0.827*sind(x(1))))/4.167;
           //swing equation before clearing the
           faulted line
61     elseif (t>=tr) then
62         xdot(2)=180*50*(Pm-(1.5*sind(x(1))))/4.167; //
           after reclosure
63     end
64 endfunction
65
66 //to find the solution of swing equation to draw the
           swing curves
67
68 //to draw the swing curves for three pole switching
           with reclosure
69 subplot(2,1,1)
70 x_1_0=[41.8,0]';t0=0; T=0:0.001:0.65;T=T';
71 tc=0.075;tr=0.325;
72 sol1=ode(x_1_0,t0,T,mac_3_pole);
73 plot(T,sol1(1,:));
74 set(gca(),'grid',[1 1]);
75 title('Swing Curve for three pole switching at '+
           string(tc)+' s'+ ' and reclosure at '+string(tr)+'
           s','fontsize',3);
76 xset('font size',3)
77 xstring(0.2,300,'MACHINE UNSTABLE');
78 xlabel('Time (in seconds)----->');
79 ylabel('Torque Angle (delta ,deg)----->');
80
81 //to draw the swing curves for single pole switching
           with reclosure
82 subplot(2,1,2)
83 x_1_0=[41.8,0]';t0=0; T=0:0.001:2.2;T=T';
84 tc=0.075;tr=0.325;

```



```
85 sol2=ode(x_1_0,t0,T,mac_1_pole);
86 plot(T,sol2(1,:));
87 set(gca(),"grid",[1 1]);
88 title('Swing Curve for single pole switching at '+
      string(tc)+' s'+ ' and reclosure at '+string(tr)+'
      s','fontsize',3);
89 xset("font size",3)
90 xstring(1.2,50,'MACHINE STABLE');
91 xlabel('Time (in seconds)----->');
92 ylabel('Torque Angle (delta ,deg)----->');
93
94 f=get("current_figure");
95 f.figure_position=[0,15];
96 f.figure_size=[560,1000];
```

Swing Curve for three pole switching at 0.075 s and reclosure at 0.325 s



Swing Curve for single pole switching at 0.075 s and reclosure at 0.325 s

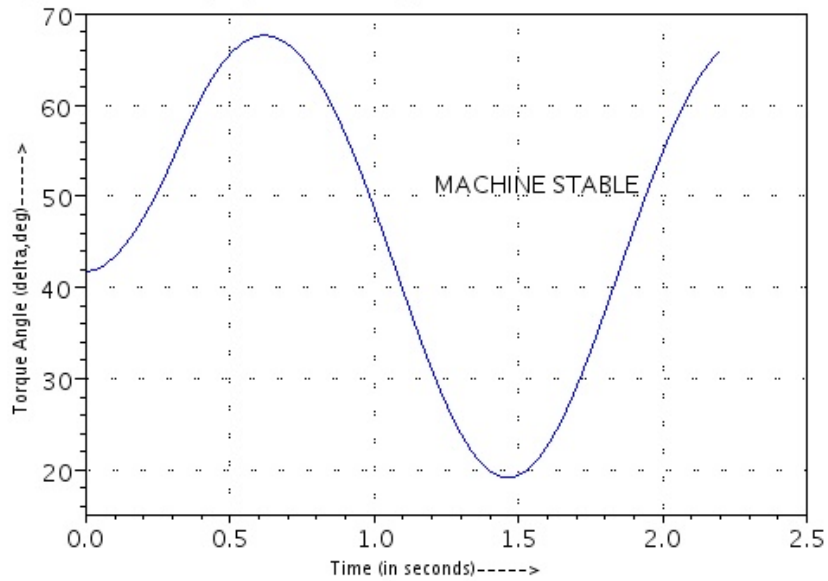


Figure 12.6: Swing Curves For Three Pole and Single Pole Switching

Chapter 13

Power System Security

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

```
1 //Chapter 13
2 //Example 13.1
3 //page 522
4 //To find the generation shift factors and the line
   outage distribution factors
5 clear;clc;
6
7 //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"
8 //
```

```

9
10 // Function to form the Ybus for primitve admittance
    values and line charging admittance values
11 function Ybus=formYbus(y_l,y_l c)
12     Ybus=[y_l(1)+y_l(2)+y_l c(1)+y_l c(2)    -y_l(1)
            -y_l(2)    0    0;
13     -y_l(1)    y_l(1)+y_l(3)+y_l(4)+y_l(5)+
            y_l c(1)+y_l c(3)+y_l c(4)+y_l c(5)    -y_l
            (3)    -y_l(4)    -y_l(5);
14     -y_l(2)    -y_l(3)    y_l(2)+y_l(3)+y_l(6)
            +y_l c(2)+y_l c(3)+y_l c(6)    -y_l(6)
            0;
15     0    -y_l(4)    -y_l(6)    y_l(6)+y_l(4)+
            y_l(7)+y_l c(6)+y_l c(4)+y_l c(7)    -y_l
            (7);
16     0    -y_l(5)    0    -y_l(7)    y_l(5)+y_l
            (7)+y_l c(5)+y_l c(7)];
17 endfunction
18
19 //Function to incorporate load flow analysis for a
    given system
20 function P_line=load_flow(E,Pg,Qg,Pl,Ql,y_l,y_l c)
21
22     //to retrieve Ybus for the given network
        parameters
23     Y=formYbus(y_l,y_l c);
24
25     //to form primitive admittance matrix and
        primitive line charging admittances that
        required later in the program
26     y1=[0 y_l(1) y_l(2) 0 0;
27         y_l(1) 0 y_l(3) y_l(4) y_l(5);
28         y_l(2) y_l(3) 0 y_l(6) 0;
29         0 y_l(4) y_l(6) 0 y_l(7);
30         0 y_l(5) 0 y_l(7) 0];
31     y c=[0 y_l c(1) y_l c(2) 0 0;
32         y_l c(1) 0 y_l c(3) y_l c(4) y_l c(5);

```

```

33     y_1c(2) y_1c(3) 0 y_1c(6) 0;
34     0 y_1c(4) y_1c(6) 0 y_1c(7);
35     0 y_1c(5) 0 y_1c(7) 0];
36
37     // to optimize the evaluation, constants like
        KLS and YLS are evaluated only once outside
        the loop
38     KL2=((Pg(2)-P1(2))-(Qg(2)-Q1(2)))/Y(2,2);
39     KL3=((Pg(3)-P1(3))-(Qg(3)-Q1(3)))/Y(3,3);
40     KL4=((Pg(4)-P1(4))-(Qg(4)-Q1(4)))/Y(4,4);
41     KL5=((Pg(5)-P1(5))-(Qg(5)-Q1(5)))/Y(5,5);
42
43     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);
44     YL31=Y(3,1)/Y(3,3);      YL32=Y(3,2)/Y(3,3);
        YL34=Y(3,4)/Y(3,3);
45     YL42=Y(4,2)/Y(4,4);      YL43=Y(4,3)/Y(4,4);
        YL45=Y(4,5)/Y(4,4);
46     YL52=Y(5,2)/Y(5,5);      YL54=Y(5,4)/Y(5,5);
47
48     //to calculate bus voltages (Refer [1] stagg ,pg
        285)
49     n=100;
50     for i=1:n
51         E(1)=E(1);
52         E(2)=(KL2/E(2)')-YL21*E(1)-YL23*E(3)-YL24*E
            (4)-YL25*E(5);
53         E(3)=(KL3/E(3)')-YL31*E(1)-YL32*E(2)-YL34*E
            (4);
54         E(4)=(KL4/E(4)')-YL42*E(2)-YL43*E(3)-YL45*E
            (5);
55         E(5)=(KL5/E(5)')-YL52*E(2)-YL54*E(4);
56     end
57     // to calculate line flows(Refer [1] stagg ,pg
        291)
58     for i=1:5
59         for j=1:5
60             S(i,j)=E(i)'*(E(i)-E(j))*yl(i,j)+E(i) '*E

```

```

        (i)*yc(i,j);
61         end
62     end
63     P_line=conj(S); //since P_line=P-jQ=conj(S)
64
65 endfunction
66 //
        ///////////////////////////////////////////////////////////////////
67 //First we will calculate the line flows for the
        system which operating under normal condition (
        without any congincy)[taken as Base system for
        comparision]//
68 //
        ///////////////////////////////////////////////////////////////////
69 //ypq          Buscode(p-q)          y'pq/2          line no
70 y11=1/(0.02+%i*0.06);    ylc_1=%i*0.030;    //l=1
        line 1-2
71 y12=1/(0.08+%i*0.24);    ylc_2=%i*0.025;    //l=2
        line 1-3
72 y13=1/(0.06+%i*0.18);    ylc_3=%i*0.020;    //l=3
        line 2-3
73 y14=1/(0.06+%i*0.18);    ylc_4=%i*0.020;    //l=4
        line 2-4
74 y15=1/(0.04+%i*0.12);    ylc_5=%i*0.015;    //l=5
        line 2-5
75 y16=1/(0.01+%i*0.03);    ylc_6=%i*0.010;    //l=6
        line 3-4
76 y17=1/(0.08+%i*0.24);    ylc_7=%i*0.025;    //l=7
        line 4-5
77
78 y_1_vector=[y11 y12 y13 y14 y15 y16 y17];
79 y_lc_vector=[ylc_1 ylc_2 ylc_3 ylc_4 ylc_5 ylc_6
        ylc_7];
80
81

```

```

82 //Assumed voltage          Generation
                                load
                                Buscode
83 //
                                MW          MVAR
                                MVAR
84 E1=1.06+%i*0;                Pg1=0;          Qg1=%i
    *0;          P11=0;          Q11=%i*0;          //1
85 E2=1+%i*0;                  Pg2=0.4;        Qg2=%i
    *0.3;          P12=0.2;        Q12=%i*0.1;        //2
86 E3=1+%i*0;                  Pg3=0;          Qg3=%i
    *0;          P13=0.45;         Q13=%i*0.15;       //3
87 E4=1+%i*0;                  Pg4=0;          Qg4=%i
    *0;          P14=0.40;         Q14=%i*0.05;       //4
88 E5=1+%i*0;                  Pg5=0;          Qg5=%i
    *0;          P15=0.60;         Q15=%i*0.10;       //5
89
90 E=[E1 E2 E3 E4 E5];          Pg=[Pg1 Pg2 Pg3 Pg4 Pg5];
    Qg=[Qg1 Qg2 Qg3 Qg4 Qg5];
91 P1=[P11 P12 P13 P14 P15];    Q1=[Q11 Q12 Q13 Q14 Q15
    ];
92
93 P_base=load_flow(E,Pg,Qg,P1,Q1,y_l_vector,
    y_lc_vector);
94 P_base=P_base*100; //converting back to MW and MVARs
95
96
97
98
99 //
    //////////////////////////////////////
100 //To find generation shift factor let us remove the
    generator at each of PV buses and calculate line
    flows//
101 //
    //////////////////////////////////////
102

```

```

103 //(i)when generator at slack bus trips
104 Pg1_old=Pg1; //required for the calculation of
      change in MWs
105 Pg1=0;Qg1=0; //generation remains same
106 Pg=[Pg1 Pg2 Pg3 Pg4 Pg5]; Qg=[Qg1 Qg2 Qg3 Qg4
      Qg5]; //updating the changed values
107
108 //conducting load flow studies
109 P_G_1=load_flow(E,Pg,Qg,Pl,Ql,y_l_vector,
      y_lc_vector);
110 P_G_1=P_G_1*100; //converting back to MW and MVARs
111 alpha1=(real(P_G_1)-real(P_base))/((Pg1_old-Pg1
      +0.001)*100); //0.001 is added to eliminate
      divide by zero error
112 alpha1=tril(alpha1); //only lower triangular
      matrix is required
113 l1=[alpha1(2,1) alpha1(3,1) alpha1(3,2) alpha1
      (4,2) alpha1(5,2) alpha1(4,3) alpha1(5,4)];
114 //(ii)When generator at Bus2 trips
115 Pg2_old=Pg2; //required for the calculation of
      change in MWs
116 Pg2=0;Qg2=0; Pg1=0;Qg1=0;
117 Pg=[Pg1 Pg2 Pg3 Pg4 Pg5]; Qg=[Qg1 Qg2 Qg3 Qg4
      Qg5]; //updating the changed values
118
119 //conducting load flow studies
120 P_G_2=load_flow(E,Pg,Qg,Pl,Ql,y_l_vector,
      y_lc_vector);
121 P_G_2=P_G_2*100; //converting back to MW and MVARs
122 alpha2=(real(P_G_2)-real(P_base))/((Pg2_old-Pg2)
      *100);
123 alpha2=tril(alpha2); //only lower triangular
      matrix is required
124 l2=[alpha2(2,1) alpha2(3,1) alpha2(3,2) alpha2
      (4,2) alpha2(5,2) alpha2(4,3) alpha2(5,4)];
125 //To print the results of generator shift factors
126
127 printf('Generator Shift Factor for Five-bus System\n

```



```

    ');
128 printf('
    _____\
    n');
129 printf('Lines\t\t\t Bus 1 \t\t\tBus 2\n');
130 printf('
    _____\
    n');
131 for i=1:7
132     printf('l = %d\t\t\t %d\t\t\t\t%0.4f\n',i,l1(
        i),l2(i));
133 end
134 printf('
    _____\
    n');
135 //
    //////////////////////////////////////
136 //To find Line Outage Distribution Factors let us
    remove each line and calculate the line flows//
137 //
    //////////////////////////////////////

138
139 //changing the network back to normal system
140 Pg2=0.4;          Qg2=%i*0.3;
141
142 //copying the original values of the network
    parameters
143 y_l_vector_normal=y_l_vector;
    y_lc_vector_normal=y_lc_vector;
144 Pg=[Pg1 Pg2 Pg3 Pg4 Pg5];    Qg=[Qg1 Qg2 Qg3 Qg4
    Qg5]; //updating the changed values
145 //when jth line trips the load flow analysis is done
    as follows
146 for j=1:7
147     y_l_vector(j)=0;y_lc_vector(j)=0;
148     P_L=load_flow(E,Pg,Qg,Pl,Ql,y_l_vector,

```

```

        y_lc_vector); //load flow analysis
149     P_L=P_L*100;
150     select j,
151         case 1 then fi0=P_base(2,1),
152         case 2 then fi0=P_base(3,1),
153         case 3 then fi0=P_base(3,2),
154         case 4 then fi0=P_base(4,2),
155         case 5 then fi0=P_base(5,2),
156         case 6 then fi0=P_base(4,3),
157         case 7 then fi0=P_base(5,4),
158     end
159     d0=(real(P_L)-real(P_base))/real(fi0);
160     d(:,j)=[d0(2,1); d0(3,1); d0(3,2); d0(4,2); d0
        (5,2); d0(4,3); d0(5,4)];
161     y_l_vector=y_l_vector_normal; y_lc_vector=
        y_lc_vector_normal; //changing the system back
        to normal
162 end
163
164 //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
165 for i=1:7
166     d(i,i)=0;
167 end
168
169 //To print the results of line outage distribution
        factors
170 printf('\n\n\n\nLine Outage Distribution Factor for
        Five-bus System\n');
171 printf('
        _____
        n');
172 printf('Lines \t\tj=1\t\tj=2\t\tj=3\t\tj=4\t\tj=5\t
        \tj=6\t\tj=7\n');
173 printf('
        _____
        n');

```

```
174 for l=1:7
175     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(1,1),d(
        1,2),d(1,3),d(1,4),d(1,5),d(1,6),d(1,7));
176 end
177 printf('
n\n\n');
```

Chapter 14

An Introduction to State Estimation of Power Systems

Scilab code Exa 14.1 Estimation of random variables

```
1 //Chapter 14
2 //Example 14.1
3 //page 533
4 //To estimate the values of the random variables x1
   and x2
5 clear;clc;
6
7 H=[1 0;0 1;1 1]; //given matrix
8 k=inv(H'*H)*H'; // from eq 14.9
9 y=['y1';'y2';'y3'];
10 k=string(k);
11 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)
    ];
12 printf('Estimate of x =\n');
13 disp(x);
```

Scilab code Exa 14.2 Estimation of random variables using WLSE

```
1 //Chapter 14
2 //Example 14.2
3 //page 534
4 //To estimate the values of the random variables x1
   and x2 using WLSE
5 clear;clc;
6
7 w=diag([0.1;1;0.1]); //assumed matrix
8 H=[1 0;0 1;1 1]; //given matrix
9 k=inv(H'*w*H)*H'*w; // from eq 14.12b
10 y=['y1 ','y2 ','y3 '];
11 Px=k*k';
12 k=string(k);
13 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)
   ];
14 printf('The weighted least square s estimate of the
   vector x =\n');
15 disp(x);
16 printf('\n\nThe matrix k is in this case found to be
   \n');
17 disp(k);
18 //covariance of measurement is assumed is assumed to
   be unit matrix
19 printf('\n\nThe covariance of the estimation error
   is obtained as Px=\n');
20 disp(Px);
21
22 printf('\n\n\n Now choosing W=1\n');
23 w=diag([1;1;1]); //assumed matrix
```


Chapter 17

Voltage Stability

Scilab code Exa 17.1 Reactive power sensitivity

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

Scilab code Exa 17.2 Capacity of static VAR compensator

```
1 //Chapter 17
2 //Example 17.2
```

```
3 //page 602
4 //To find capacity of static VAR compensator
5 clear;clc;
6
7 delta_V=5/100; //allowable voltage fluctuation
8 S_sc=5000; //system short circuit capacity in MVA
9 delta_Q=delta_V*S_sc; //size of the compensator
10 printf('The capacity of the static VAR compensator
        is +%d MVAR',delta_Q);
```
