

Scilab Textbook Companion for
Electric Power Transmission System
Engineering Analysis And Design
by T. Gonen¹

Created by
Kavan A. B
B.E
Electrical Engineering
Sri Jayachamarajendra College Of Engineering
College Teacher
R. S. Ananda Murthy
Cross-Checked by
K. V. P. Pradeep

July 31, 2019

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: Electric Power Transmission System Engineering Analysis And Design

Author: T. Gonen

Publisher: Crc Press, Florida

Edition: 2

Year: 2009

ISBN: 9781439802540

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

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

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

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

Contents

List of Scilab Codes	4
2 TRANSMISSION LINE STRUCTURES AND EQUIPMENT	5
3 FUNDAMENTAL CONCEPTS	8
4 OVERHEAD POWER TRANSMISSION	13
5 UNDERGROUND POWER TRANSMISSION AND GAS INSULATED TRANSMISSION LINES	46
6 DIRECT CURRENT POWER TRANSMISSION	74
7 TRANSIENT OVERVOLTAGES AND INSULATION CO- ORDINATION	88
8 LIMITING FACTORS FOR EXTRA HIGH AND ULTRA- HIGH VOLTAGE TRANSMISSION	101
9 SYMMETRICAL COMPONENTS AND FAULT ANAL- YSIS	105
10 PROTECTIVE EQUIPMENT AND TRANSMISSION SYS- TEM PROTECTION	135

12 CONSTRUCTION OF OVERHEAD LINES	147
13 SAG AND TENSION ANALYSIS	155
14 APPENDIX C REVIEW OF BASICS	159

List of Scilab Codes

Exa 2.1	calculate tolerable touch step potential	5
Exa 3.1	determine SIL of the line	8
Exa 3.2	determine effective SIL	9
Exa 3.3	calculate RatedCurrent MVARrating Current-Value	11
Exa 4.1	calculate LinetoNeutralVoltage LinetoLineVoltage LoadAngle	13
Exa 4.2	calculate percentage voltage regulation using equation	18
Exa 4.3	mutual impedance between the feeders	19
Exa 4.4	calculate A B C D Vs I pf efficiency	20
Exa 4.5	calculate A B C D Vs I pf efficiency using nominal pi	22
Exa 4.6	calculate A B C D Vs I pf Ploss n VR Is Vr	25
Exa 4.7	find equivalent pi T circuit and Nominal pi T circuit	28
Exa 4.8	calculate attenuation phase change lamda v Vir Vrr Vr Vis Vrs Vs	30
Exa 4.9	calculate SIL	33
Exa 4.10	determine equ A B C D constant	34
Exa 4.11	determine equ A B C D constant	35
Exa 4.12	calculate Is Vs Zin P var	37
Exa 4.13	calculate SIL Pmax Qc Vroc	38
Exa 4.14	calculate SIL Pmax Qc cost Vroc	40
Exa 4.15	calculate La XL Cn Xc	43
Exa 5.1	calculate Emax Emin r	46
Exa 5.2	calculate potential gradient E1	47
Exa 5.3	calculate Ri Power loss	49

Exa 5.4	calculate charging current Ic	50
Exa 5.5	calculate Ic Is pf	51
Exa 5.6	calculate Geometric factor G1 Ic	53
Exa 5.7	calculate Emax C Ic Ri Plc Pdl Pdh	54
Exa 5.8	calculate Rdc Reff percent reduction	56
Exa 5.9	calculate Xm Rs deltaR Ra ratio Ps	57
Exa 5.10	calculate zero sequence impedance Z00 Z0 Z0a	60
Exa 5.11	calculate C0 C1 C2 X0 X1 X2 I0 I1 I2 . . .	63
Exa 5.12	calculate Zabc Z012	65
Exa 5.15	calculate PIOH PIGIL EIOH ELGIL ClOH Elavg-GIL ClavgOH ClavgGIL Csavings breakeven period	67
Exa 5.16	calculate A1 A2 A of OH GIL and submarine transmission line	70
Exa 6.1	determine Vd Id ratio of dc to ac insulation level	74
Exa 6.2	determine Vd ratio of Pdc to Pac and Ploss dc to Ploss ac	76
Exa 6.3	calculate KVA rating Wye side KV rating .	77
Exa 6.4	determine Xc for all 3 possible values of ac system reactance	78
Exa 6.5	calculate u Vdr pf Qr	80
Exa 6.6	determine alpha u pf Qr	82
Exa 6.7	determine u mode Id or Vdr	83
Exa 6.10	determine Vd0 E u pf Qr No of bucks	84
Exa 7.1	determine surge Power surge current	88
Exa 7.2	determine surge Power surge current	89
Exa 7.4	determine Crv Cri vb v Crfv ib i Crfi	90
Exa 7.5	determine if Cr Crf v i vb ib plot of voltage and current surges	92
Exa 7.6	determine Crs Crr lattice diagram volatge plot of receiving end voltage with time	95
Exa 8.1	determine disruptive critical rms V0 and visual critical rms Vv	101
Exa 8.2	determine total fair weather corona loss Pc .	102
Exa 9.1	determine symmetrical components for phase voltages	105
Exa 9.2	determine complex power V012 I012	107

Exa 9.3	determine line impedance and sequence impedance matrix	109
Exa 9.4	determine line impedance and sequence impedance matrix of transposed line	110
Exa 9.5	determine mo m2 for zero negative sequence unbalance	112
Exa 9.6	determine Pabc Cabc C012 d0 d2	114
Exa 9.9	determine Iphase Isequence Vphase Vsequence LinetoLineVoltages at Faultpoints	116
Exa 9.10	determine Isequence Iphase Vsequence at fault G1 G2	118
Exa 9.11	determine Iphase Isequence Vphase Vsequence LinetoLineVoltages at Faultpoints	122
Exa 9.12	determine Iphase Isequence Vphase Vsequence LinetoLineVoltages at Faultpoints	124
Exa 9.13	determine Iphase Isequence Vphase Vsequence LinetoLineVoltages at Faultpoints	127
Exa 9.14	determine admittance matrix	129
Exa 9.15	determine uncoupled positive and negative sequence	131
Exa 9.16	determine Xc0 C0 Ipc Xpc Lpc Spc Vpc . .	132
Exa 10.1	calculate subtransient fault current in pu and ampere	135
Exa 10.2	determine max Idc Imax Imomentary Sinterrupting Smomentary	136
Exa 10.4	determine Rarc Z LineImpedanceAngle with Rarc and without	138
Exa 10.5	determine protection zones and plot of operating time vs impedance	139
Exa 10.6	determine Imax CT VT ZLoad Zr	142
Exa 10.7	determine setting of zone1 zone2 zone3 of mho relay R12	145
Exa 12.1	calculate cost of relocating affordability . .	147
Exa 12.2	calculate pressure of wind on pole and conductors	149
Exa 12.3	calculate min required pole circumference at ground line	150
Exa 12.4	calculate Th beta Tv Tg	152

Exa 13.1	calculate length sag Tmax Tmin Tappr . . .	155
Exa 13.2	calculate Wi Wt P We sag vertical sag . . .	156
Exa 1.C	determine power S12 P12 Q12	159
Exa 2.C	determine reactance Zbhv Zblv Xhv Xlv . .	160
Exa 3.C	determine turns ratio Xlv Xpu	162
Exa 4.C	determine KVA KV Zb Ib I new Zpu V1 V2 V4 S1 S2 S4 table	164
Exa 5.C	determine inductive reactance using equ C135 and tables	168
Exa 6.C	determine shunt capacitive reactance using equ C156 and tables	170

Chapter 2

TRANSMISSION LINE STRUCTURES AND EQUIPMENT

Scilab code Exa 2.1 calculate tolerable touch step potential

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 2 : TRANSMISSION LINE STRUCTURES AND
7 // EQUIPMENT
8 // EXAMPLE : 2.1 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11 // GIVEN DATA
12 t_s = 0.49 ; // Human body is in contact with 60 Hz
13 // power for 0.49 sec
13 r = 100 ; // Resistivity of soil based on IEEE std
```

```

80-2000

14
15 // CALCULATIONS
16 // For case (a)
17 v_touch50 = 0.116*(1000+1.5*r)/sqrt(t_s) ; // Maximum allowable touch voltage for 50 kg body weight in volts
18
19 // For case (b)
20 v_step50 = 0.116*(1000+6*r)/sqrt(t_s) ; // Maximum allowable step voltage for 50 kg body weight in volts
21 // Above Equations of case (a) & (b) applicable if no protective surface layer is used
22
23 // For metal to metal contact below equation holds good . Hence resistivity is zero
24 r_1 = 0 ; // Resistivity is zero
25
26 // For case (c)
27 v_mm_touch50 = 0.116*(1000)/sqrt(t_s) ; // Maximum allowable touch voltage for 50 kg body weight in volts for metal to metal contact
28
29 // For case (d)
30 v_mm_touch70 = 0.157*(1000)/sqrt(t_s) ; // Maximum allowable touch voltage for 70 kg body weight in volts for metal to metal contact
31
32 // DISPLAY RESULTS
33 disp("EXAMPLE : 2.1 : SOLUTION :-") ;
34 printf("\n (a) Tolerable Touch potential , V_touch50 = %.f V , for 50 kg body weight \n",v_touch50) ;
35 printf("\n (b) Tolerable Step potential , V_step50 = %.f V , for 50 kg body weight \n",v_step50) ;
36 printf("\n (c) Tolerable Touch Voltage for metal-to-metal contact , V_mm_touch50 = %.1f V , for 50 kg body weight \n",v_mm_touch50) ;

```

```
37 printf("\n (d) Tolerable Touch Voltage for metal-to-
metal contact , V_mm_touch70 = %.1f V , for 70 kg
body weight \n",v_mm_touch70) ;
```

Chapter 3

FUNDAMENTAL CONCEPTS

Scilab code Exa 3.1 determine SIL of the line

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING  
2 // ANALYSIS AND DESIGN  
3 // TURAN GONEN  
4 // CRC PRESS  
5 // SECOND EDITION  
6 // CHAPTER : 3 : FUNDAMENTAL CONCEPTS  
7  
8 // EXAMPLE : 3.1 :  
9 clear ; clc ; close ; // Clear the work space and  
// console  
10  
11 // GIVEN DATA  
12 kV = 345 ; // Three phase transmission line voltage  
// in kV  
13 Z_s = 366 ; // Surge impedance of line in  
14 a = 24.6 ; // Spacing between adjacent conductors in  
// feet  
15 d = 1.76 ; // Diameter of conductor in inches  
16  
17 // CALCULATIONS
```

```

18 SIL = (kV)^2/Z_s ; // Surge Impedance loading of
line in MW
19
20 // DISPLAY RESULTS
21 disp("EXAMPLE : 3.1 : SOLUTION :-") ;
22 printf("\n Surge Impedance Loading of line , SIL = %
.f MW \n",SIL) ;
23
24 printf("\n NOTE: Unit of SIL is MW and surge
impedance is " ) ;
25 printf("\n ERROR: Mistake in unit of SIL in textbook
\n") ;

```

Scilab code Exa 3.2 determine effective SIL

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 3 : FUNDAMENTAL CONCEPTS
7
8 // EXAMPLE : 3.2 :
9 clear ; clc ; close ; // Clear the work space and
console
10
11 // GIVEN DATA
12 SIL = 325 ; // Surge impedance Loading in MW . From
exa 3.1
13 kV = 345 ; // Transmission line voltage in kV . From
exa 3.1
14
15 // For case (a)
16 t_shunt1 = 0.5 ; // shunt capacitive compensation is

```

```

      50%
17 t_series1 = 0 ; // no series compensation
18
19 // For case (b)
20 t_shunt2 = 0.5 ; // shunt compensation using shunt
      reactors is 50%
21 t_series2 = 0 ; // no series capacitive compensation
22
23 // For case (c)
24 t_shunt3 = 0 ; // no shunt compensation
25 t_series3 = 0.5 ; // series capacitive compensation
      is 50%
26
27 // For case (d)
28 t_shunt4 = 0.2 ; // shunt capacitive compensation is
      20%
29 t_series4 = 0.5; // series capacitive compensation
      is 50%
30
31 // CALCULATIONS
32 // For case (a)
33 SIL1 = SIL*(sqrt( (1-t_shunt1)/(1-t_series1) )) ; //
      Effective SIL in MW
34
35 // For case (b)
36 SIL2 = SIL*(sqrt( (1+t_shunt2)/(1-t_series2) )) ; //
      Effective SIL in MW
37
38 // For case (c)
39 SIL3 = SIL*(sqrt( (1-t_shunt3)/(1-t_series3) )) ; //
      Effective SIL in MW
40
41 // For case (d)
42 SIL4 = SIL*(sqrt( (1-t_shunt4)/(1-t_series4) )) ; //
      Effective SIL in MW
43
44 // DISPLAY RESULTS
45 disp("EXAMPLE : 3.2 : SOLUTION :-") ;

```

```

46 printf("\n (a) Effective SIL , SIL_comp = %.f MW \n"
        ,SIL1) ;
47 printf("\n (b) Effective SIL , SIL_comp = %.f MW \n"
        ,SIL2) ;
48 printf("\n (c) Effective SIL , SIL_comp = %.f MW \n"
        ,SIL3) ;
49 printf("\n (d) Effective SIL , SIL_comp = %.f MW \n"
        ,SIL4) ;
50
51 printf("\n NOTE: Unit of SIL is MW and surge
           impedance is      ") ;
52 printf("\n ERROR: Mistake in unit of SIL in textbook
           \n") ;

```

Scilab code Exa 3.3 calculate RatedCurrent MVARrating CurrentValue

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 3 : FUNDAMENTAL CONCEPTS
7
8 // EXAMPLE : 3.3 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 // For case (c)
13 I_normal = 1000 ; // Normal full load current in
   Ampere
14
15 // CALCULATIONS
16 // For case (a) equation is (1.5 pu)*I_rated = (2 pu)

```

```

        *I_normal
17 // THEREFORE
18 // I_rated = (1.333pu)*I_normal ; // Rated current
   in terms of per unit value of the normal load
   current
19
20 // For case (b)
21 Mvar = (1.333)^2 ; // Increase in Mvar rating in per
   units
22
23 // For case (c)
24 I_rated = (1.333)*I_normal ; // Rated current value
25
26 // DISPLAY RESULTS
27 disp("EXAMPLE : 3.3 : SOLUTION :-") ;
28 printf("\n (a) Rated current , I_rated = (1.333 pu)*
   I_normal \n") ;
29 printf("\n (b) Mvar rating increase = %.2f pu \n",
   Mvar) ;
30 printf("\n (c) Rated current value , I_rated = %.f A
   \n", I_rated) ;

```

Chapter 4

OVERHEAD POWER TRANSMISSION

Scilab code Exa 4.1 calculate LinetoNeutralVoltage LinetoLineVoltage LoadAngle

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.1 :
9 clear ; clc ; close ; // Clear the work space and
10 console
11 // GIVEN DATA
12 V_RL_L = 23*10^3 ; // line to line voltage in volts
13 z_t = 2.48+%i*6.57 ; // Total impedance in ohm/phase
14 p = 9*10^6 ; // load in watts
15 pf = 0.85 ; // lagging power factor
16
17 // CALCULATIONS
```

```

18 // METHOD I : USING COMPLEX ALGEBRA
19
20 V_RL_N = (V_RL_L)/sqrt(3) ; // line-to-neutral
   reference voltage in V
21 I = (p/(sqrt(3)*V_RL_L*pf))*( pf - %i*sindacosd(pf)
   )) ; // Line current in amperes
22 IZ = I*z_t ;
23 V_SL_N = V_RL_N + IZ // Line to neutral voltage at
   sending end in volts
24 V_SL_L = sqrt(3)*V_SL_N ; // Line to line voltage at
   sending end in volts
25
26 // DISPLAY RESULTS
27 disp("EXAMPLE : 4.1 : SOLUTION :-") ;
28 disp("METHOD I : USING COMPLEX ALGEBRA") ;
29 printf("\n (a) Line-to-neutral voltage at sending
   end , V_SL_N = %.f<%i f V \n",abs(V_SL_N),atand(
   imag(V_SL_N),real(V_SL_N)) ) ;
30 printf("\n i.e Line-to-neutral voltage at sending
   end , V_SL_N = %.f V \n",abs(V_SL_N)) ;
31 printf("\n Line-to-line voltage at sending end ,
   V_SL_L = %.f<%i f V \n",abs(V_SL_L),atand( imag(
   V_SL_L),real(V_SL_L)) ) ;
32 printf("\n i.e Line-to-line voltage at sending end ,
   V_SL_L = %.f V \n",abs(V_SL_L)) ;
33 printf("\n (b) load angle , = %.1f degree \n",
   atand( imag(V_SL_L),real(V_SL_L)) ) ;
34 printf("\n") ;
35
36
37 // CALCULATIONS
38 // METHOD II : USING THE CURRENT AS REFERENCE PHASOR
39 theta_R = acosd(pf) ;
40 V1 = V_RL_N*cosd(theta_R) + abs(I)*real(z_t) ; // 
   unit is volts
41 V2 = V_RL_N*sind(theta_R) + abs(I)*imag(z_t) ; // 
   unit is volts
42 V_SL_N2 = sqrt( (V1^2) + (V2^2) ) ; // Line to

```

```

        neutral voltage at sending end in volts/phase
43 V_SL_L2 = sqrt(3) * V_SL_N2 ; // Line to line
        voltage at sending end in volts
44 theta_s = atand(V2/V1) ;
45 delta = theta_s - theta_R ;
46
47 // DISPLAY RESULTS
48 disp("METHOD II : USING THE CURRENT AS REFERENCE
        PHASOR");
49 printf("\n (a) Line-to-neutral voltage at sending
        end , V_SL_N = %.f V \n",V_SL_N2) ;
50 printf("\n         Line-to-line voltage at sending end ,
        V_SL_L = %.f V \n",V_SL_L2) ;
51 printf("\n (b) load angle ,      = %.1f degree \n",
        delta) ;
52 printf("\n") ;
53
54 // CALCULATIONS
55 // METHOD III : USING THE RECEIVING-END VOLTAGE AS
        REFERENCE PHASOR
56 // for case (a)
57 V_SL_N3 = sqrt( (V_RL_N + abs(I) * real(z_t) * cosd(
        theta_R) + abs(I) * imag(z_t) * sind(theta_R))^2
        + (abs(I)*imag(z_t) * cosd(theta_R) - abs(I) *
        real(z_t) * sind(theta_R))^2) ;
58 V_SL_L3 = sqrt(3)*V_SL_N3 ;
59
60 // for case (b)
61 delta_3 = atand( (abs(I)*imag(z_t) * cosd(theta_R) -
        abs(I) * real(z_t) * sind(theta_R))/(V_RL_N +
        abs(I) * real(z_t) * cosd(theta_R) + abs(I) *
        imag(z_t) * sind(theta_R)) ) ;
62
63 // DISPLAY RESULTS
64 disp("METHOD III : USING THE RECEIVING END VOLTAGE
        AS REFERENCE PHASOR") ;
65 printf("\n (a) Line-to-neutral voltage at sending
        end , V_SL_N = %.f V \n",V_SL_N3) ;

```

```

66 printf("\n      Line-to-line voltage at sending end ,\n      V_SL_L = %.f V \n",V_SL_L3) ;
67 printf("\n (b) load angle ,    = %.1f degree \n",
       delta_3) ;
68 printf("\n") ;
69
70 // CALCULATIONS
71 // METHOD IV : USING POWER RELATIONSHIPS
72 P_4 = 9 ; // load in MW (Given)
73 P_loss = 3 * (abs(I))^2 * real(z_t) * 10^-6 ; // Power loss in line in MW
74 P_T = P_4 + P_loss ; // Total input power to line in MW
75 Q_loss = 3 * (abs(I))^2 * imag(z_t) * 10^-6 ; // Var loss of line in Mvar lagging
76 Q_T = ( (P_4*sind(theta_R))/cosd(theta_R) ) + Q_loss ; // Total megavar input to line in Mvar lagging
77 S_T = sqrt( (P_T^2)+(Q_T^2) ) ; // Total megavoltampere input to line
78 // for case (a)
79 V_SL_L4 = S_T*10^6/(sqrt(3) * abs(I)) ; // line to line voltage in volts
80 V_SL_N4 = V_SL_L4/sqrt(3) ; // Line to line neutral in volts
81
82 // for case (b)
83 theta_S4 = acosd(P_T/S_T) ; // Lagging
84 delta_4 = theta_s - theta_R ;
85
86 // DISPLAY RESULTS
87 disp("METHOD IV : USING POWER RELATIONSHIPS");
88 printf("\n (a) Line-to-neutral voltage at sending end , V_SL_N = %.f V \n",V_SL_N4) ;
89 printf("\n (a) Line-to-line voltage at sending end , V_SL_L = %.f V \n",V_SL_L4) ;
90 printf("\n (b) load angle ,    = %.1f degree \n",
       delta_4) ;
91 printf("\n");

```

```

92
93 // CALCULATIONS
94 // METHOD V : Treating 3- line as 1- line having
    having V_S and V_R represent line-to-line
    voltages not line-to-neutral voltages
95 // for case (a)
96 I_line = (p/2)/(V_RL_L * pf) ; // Power delivered is
    4.5 MW
97 R_loop = 2*real(z_t) ;
98 X_loop = 2*imag(z_t) ;
99 V_SL_L5 = sqrt( (V_RL_L * cosd(theta_R) + I_line*
    R_loop)^2 + (V_RL_L * sind(theta_R) + I_line *
    X_loop)^2) ; // line to line voltage in V
100 V_SL_N5 = V_SL_L5/sqrt(3) ; // line to neutral
    voltage in V
101
102 // for case (b)
103 theta_S5 = atand((V_RL_L * sind(theta_R) + I_line *
    X_loop)/(V_RL_L * cosd(theta_R) + I_line*R_loop))
    ;
104 delta_5 = theta_S5 - theta_R ;
105
106 // DISPLAY RESULTS
107 disp("METHOD V : TREATING 3- LINE AS 1- LINE") ;
108 printf("\n (a) Line to neutral voltage at sending
    end , V_SL_N = %.f V \n",V_SL_N5) ;
109 printf("\n (a) Line to line voltage at sending end ,
    V_SL_L = %.f V \n",V_SL_L5) ;
110 printf("\n (b) load angle ,     = %.1f degree \n",
    delta_5) ;
111 printf("\n") ;
112
113 printf("\n NOTE : ERROR : Change in answer because
    root(3) = 1.73 is considered in Textbook ") ;
114 printf("\n But here sqrt(3) = 1.7320508 is
    considered \n") ;

```

Scilab code Exa 4.2 calculate percentage voltage regulation using equation

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6
7 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
8 // EXAMPLE : 4.2 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11
12 // GIVEN DATA
13 // for case (a)
14 V_S = 14803 ; // sending end phase voltage at no
15 // load in volts . From exa 4.1
16 V_R = 13279.056 ; // receiving end phase voltage at
17 // full load in volts . From exa 4.1
18
19 // for case (b)
20 I_R = 265.78785 ; // Line current in amperes . From
21 // exa 4.1
22 z_t = 2.48+%i*6.57 ; // Total impedance in ohm/phase
23 pf = 0.85 ; // power factor
24 theta_R = acosd(pf) ;
25
26 // CALCULATIONS
27 // for case (a)
28 V_reg1 = ( (V_S - V_R)/V_R )*100 ; // percentage
29 // voltage regulation using equ 4.29
30
31 // for case (b)
32 V_reg2 = (I_R * ( real(z_t) * cosd(theta_R) + imag(
```

```

27
28 // DISPLAY RESULTS
29 disp("EXAMPLE : 4.2 : SOLUTION :-") ;
30 printf("\n (a) Percentage of voltage regulation
      using equ 4.29 = %.1f \n",V_reg1) ;
31 printf("\n (b) Percentage of voltage regulation
      using equ 4.31 = %.1f \n",V_reg2) ;
32
33 printf("\n NOTE : ERROR : The question is with
      respect to values given in Exa 4.1 not 4.5 \n") ;

```

Scilab code Exa 4.3 mutual impedance between the feeders

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.3 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 Z_xy = 0.09 + %i*0.3 ; // Mutual impedance between
   two parallel feeders in /mi per phase
13 Z_xx = 0.604*exp(%i*50.4*pi/180) ; // Self
   impedance of feeders in /mi per phase
14 Z_yy = 0.567*exp(%i*52.9*pi/180) ; // Self
   impedance of feeders in /mi per phase
15

```

```

16 // SOLUTION
17 Z_2 = Z_xx - Z_xy ; // mutual impedance between
   feeders
18 Z_4 = Z_yy - Z_xy ; // mutual impedance between
   feeders
19
20 // DISPLAY RESULTS
21 disp("EXAMPLE : 4.3 : SOLUTION :-") ;
22 printf("\n Mutual impedance at node 2 , Z_2 = %.3f
   + j%.3f \n", real(Z_2), imag(Z_2)) ;
23 printf("\n Mutual impedance at node 4 , Z_4 = %.3f
   + j%.3f \n", real(Z_4), imag(Z_4)) ;

```

Scilab code Exa 4.4 calculate A B C D Vs I pf efficiency

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.4 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 V = 138*10^3 ; // transmission line voltage in V
13 P = 49*10^6 ; // load power in Watts
14 pf = 0.85 ; // lagging power factor
15 Z = 95 * exp(%i*78*pi/180) ; // line constants in
   siemens
16 Y = 0.001 * exp(%i*90*pi/180) ; // line constants
   in siemens

```

```

17
18 // CALCULATIONS
19 V_RL_N = V/sqrt(3) ;
20 theta_R = acosd(pf) ;
21 I_R = P/(sqrt(3)*V*pf)*( cosd(theta_R) - %i*sind(
    theta_R) ) ; // receiving end current in ampere
22
23 // for case (a)
24 // A,B,C,D constants for nominal-T circuit
    representation
25 A = 1 + (1/2)*Y*Z ;
26 B = Z + (1/4)*Y*Z^2 ;
27 C = Y ;
28 D = A ;
29
30 // for case (b)
31 P = [A B ; C D] * [V_RL_N ; I_R] ;
32 V_SL_N = P(1,1) ; // Line-to-neutral Sending end
    voltage in V
33 V_SL_L = sqrt(3) * abs(V_SL_N) * exp(%i* (atand(
    imag(V_SL_N),real(V_SL_N) ) + 30 )* %pi/180) ; //
    Line-to-line voltage in V
34 // NOTE that an additional 30 degree is added to the
    angle since line to line voltage is 30 degree
    ahead of its line to neutral voltage
35
36
37 // for case (c)
38 I_S = P(2,1); // Sending end current in A
39
40 // for case (d)
41 theta_s = atand( imag(V_SL_N),real(V_SL_N) ) - atand(
    imag(I_S),real(I_S) ) ;
42
43 // for case (e)
44 n = (sqrt(3) * V * abs(I_R) * cosd(theta_R)/(sqrt(3)
    * abs(I_S) * abs(V_SL_L) * cosd(theta_s) ))*100
    ; // Efficiency

```

```

45
46 // DISPLAY RESULTS
47 disp("EXAMPLE : 4.4 : SOLUTION :-") ;
48 printf("\n (a) A constant of line , A = %.4f<%1f \n
        ",abs(A),atand( imag(A),real(A)) ) ;
49 printf("\n      B constant of line , B = %.2f<%1f
        \n",abs(B),atand( imag(B),real(B)) ) ;
50 printf("\n      C constant of line , C = %.3f<%1f S
        \n",abs(C),atand( imag(C),real(C)) ) ;
51 printf("\n      D constant of line , D = %.4f<%1f \n
        ",abs(D),atand( imag(D),real(D)) ) ;
52 printf("\n (b) Sending end line-to-neutral voltage ,
        V_SL_N = %.1f<%1f V \n",abs(V_SL_N),atand( imag(
        V_SL_N),real(V_SL_N)) ) ;
53 printf("\n      Sending end line-to-line voltage ,
        V_SL_L = %.1f<%1f V \n",abs(V_SL_L),atand( imag(
        V_SL_L),real(V_SL_L)) ) ;
54 printf("\n (c) sending end current , I_S = %.2f<%1f
        A \n",abs(I_S),atand( imag(I_S),real(I_S)) ) ;
55 printf("\n (d) sending end power factor , cos _s =
        %.3f \n",cosd(theta_s)) ;
56 printf("\n (e) Efficiency of transmission ,      = %.2
        f Percentage \n",n) ;
57
58 printf("\n NOTE : From A = 0.9536<0.6 , magnitude is
        0.9536 & angle is 0.6 degree") ;
59 printf("\n ERROR : Change in answer because root(3)
        = 1.73 is considered in Textbook ") ;
60 printf("\n But here sqrt(3) = 1.7320508 is
        considered \n") ;

```

Scilab code Exa 4.5 calculate A B C D Vs I pf efficiency using nominal pi

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN

```

```

2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.5 :
9 clear ; clc ; close ; // Clear the work space and
  console
10
11 // GIVEN DATA
12 V = 138*10^3 ; // Transmission line voltage in V
13 P = 49*10^6 ; // load power in Watts
14 pf = 0.85 ; // lagging power factor
15 Z = 95 * exp(%i*78*pi/180) ; // line constants in
16 Y = 0.001 * exp(%i*90*pi/180) ; // line constants
  in siemens
17
18 // CALCULATIONS
19 V_RL_N = V/sqrt(3) ;
20 theta_R = acosd(pf) ;
21 I_R = P/(sqrt(3)*V*pf) * ( cosd(theta_R) - %i*sind(
  theta_R) ) ; // Receiving end current in A
22
23 // for case (a)
24 // A,B,C,D constants for nominal- circuit
  representation
25 A = 1 + (1/2)*Y*Z ;
26 B = Z ;
27 C = Y + (1/4)*(Y^2)*Z ;
28 D = 1 + (1/2)*Y*Z ;
29
30 // for case (b)
31 P = [A B ; C D] * [V_RL_N ; I_R] ;
32 V_SL_N = P(1,1) ; // Line-to-neutral Sending end
  voltage in V
33 V_SL_L = sqrt(3) * abs(V_SL_N) * exp(%i* ( atand(

```

```

        imag(V_SL_N),real(V_SL_N) ) + 30 )* %pi/180) ; //  

        Line-to-line voltage in V  

34 // NOTE that an additional 30 degree is added to the  

    angle since line-to-line voltage is 30 degree  

    ahead of its line-to-neutral voltage  

35  

36  

37 // for case (c)  

38 I_S = P(2,1); // Sending end current in A  

39  

40 // for case (d)  

41 theta_s = atand( imag(V_SL_N),real(V_SL_N) ) - atand  

    ( imag(I_S),real(I_S) ) ;  

42  

43 // for case (e)  

44 n = (sqrt(3) * V * abs(I_R) * cosd(theta_R)/(sqrt(3)  

    * abs(I_S) * abs(V_SL_L) * cosd(theta_s) ))*100  

    ; // Efficiency  

45  

46 // DISPLAY RESULTS  

47 disp("EXAMPLE : 4.5 : SOLUTION :-") ;  

48 printf("\\n (a) A constant of line , A = %.4f<%1f \\n  

    ",abs(A),atand( imag(A),real(A) ) );  

49 printf("\\n      B constant of line , B = %.2f<%1f  

    \\n",abs(B),atand( imag(B),real(B) ) );  

50 printf("\\n      C constant of line , C = %.3f<%1f S  

    \\n",abs(C),atand( imag(C),real(C) ) );  

51 printf("\\n      D constant of line , D = %.4f<%1f \\n  

    ",abs(D),atand( imag(D),real(D) ) );  

52 printf("\\n (b) Sending end line-to-neutral voltage ,  

    V_SL_N = %.1f<%1f V \\n",abs(V_SL_N),atand( imag  

    (V_SL_N),real(V_SL_N) ) );  

53 printf("\\n      Sending end line-to-line voltage ,  

    V_SL_L = %.1f<%1f V \\n",abs(V_SL_L),atand( imag  

    (V_SL_L),real(V_SL_L) ) );  

54 printf("\\n (c) sending end current , I_S = %.2f<%1f  

    A \\n",abs(I_S),atand( imag(I_S),real(I_S) ) );  

55 printf("\\n (d) sending end power factor , cos _s =

```

```

    %.3f \n", cosd(theta_s)) ;
56 printf("\n (e) Efficiency of transmission ,      = %.2
      f Percentage \n",n) ;
57
58 printf("\n NOTE : ERROR : Change in answer because
      root(3) = 1.73 is considered in Textbook ") ;
59 printf("\n But here sqrt(3) = 1.7320508 is
      considered \n") ;

```

Scilab code Exa 4.6 calculate A B C D Vs I pf P Ploss n VR Is Vr

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.6 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 V_RL_L = 138*10^3 ; // transmission line voltage in
   V
13 R = 0.1858 // Line constant in /mi
14 f = 60 // frequency in Hertz
15 L = 2.60*10^-3 // Line constant in H/mi
16 C = 0.012*10^-6 // Line constant in F/mi
17 pf = 0.85 // Lagging power factor
18 P = 50*10^6 // load in VA
19 l = 150 // length of 3- transmission line in mi
20
21 // CALCULATIONS

```

```

22 z = R + %i*2*pi*f*L ; // Impedance per unit length
   in /mi
23 y = %i*2*pi*C*f ; // Admittance per unit length in
   S/mi
24 g = sqrt(y*z) ; // Propagation constant of line per
   unit length
25 g_l = real(g) * l + %i * imag(g) * l ; //
   Propagation constant of line
26 Z_c = sqrt(z/y) ; // Characteristic impedance of
   line
27 V_RL_N = V_RL_L/sqrt(3) ;
28 theta_R = acosd(pf) ;
29 I_R = P/(sqrt(3)*V_RL_L)*( cosd(theta_R) - %i*sind(
   theta_R) ) ; // Receiving end current in A
30
31 // for case (a)
32 // A,B,C,D constants of line
33 A = cosh(g_l) ;
34 B = Z_c * sinh(g_l) ;
35 C = (1/Z_c) * sinh(g_l) ;
36 D = A ;
37
38 // for case (b)
39 P = [A B ; C D] * [V_RL_N ; I_R] ;
40 V_SL_N = P(1,1) ; // Line-to-neutral Sending end
   voltage in V
41 V_SL_L = sqrt(3) * abs(V_SL_N) * exp(%i* ( atand(
   imag(V_SL_N),real(V_SL_N) ) + 30 )* %pi/180) ; //
   Line-to-line voltage in V
42 // NOTE that an additional 30 degree is added to the
   angle since line-to-line voltage is 30 degree
   ahead of its line-to-neutral voltage
43
44 // for case (c)
45 I_S = P(2,1); // Sending end current in A
46
47 // for case (d)
48 theta_s = atand( imag(V_SL_N),real(V_SL_N) ) - atand

```

```

        ( imag(I_S),real(I_S) ) ; // Sending-end pf
49
50 // For case (e)
51 P_S = sqrt(3) * abs(V_SL_L) * abs(I_S) * cosd(theta_s) ; // Sending end power
52
53 // For case (f)
54 P_R = sqrt(3)*abs(V_RL_L)*abs(I_R)*cosd(theta_R) ;
      // Receiving end power
55 P_L = P_S - P_R ; // Power loss in line
56
57 // For case (g)
58 n = (P_R/P_S)*100 ; // Transmission line efficiency
59
60 // For case (h)
61 reg = (( abs(V_SL_N) - V_RL_N )/V_RL_N )*100 ; // Percentage of voltage regulation
62
63 // For case (i)
64 Y = y * l ; // unit is S
65 I_C = (1/2) * Y * V_SL_N ; // Sending end charging current in A
66
67 // For case (j)
68 Z = z * l ;
69 V_RL_N0 = V_SL_N - I_C*Z ;
70 V_RL_L0 = sqrt(3) * abs(V_RL_N0) * exp(%i* ( atand(  

      imag(V_RL_N0),real(V_RL_N0) ) + 30 )* %pi/180) ;
      // Line-to-line voltage at receiving end in V
71
72 // DISPLAY RESULTS
73 disp("EXAMPLE : 4.6 :SOLUTION :-") ;
74 printf("\\n (a) A constant of line , A = %.4f<%2f \\n  

      ",abs(A),atand( imag(A),real(A) )) ;
75 printf("\\n      B constant of line , B = %.2f<%2f  

      ",abs(B),atand( imag(B),real(B) )) ;
76 printf("\\n      C constant of line , C = %.5f<%2f S  

      ",abs(C),atand( imag(C),real(C) )) ;

```

```

77 printf("\n      D constant of line , D = %.4f<%.2f \n
    ", abs(D), atan( imag(D), real(D) ) );
78 printf("\n (b) Sending end line-to-neutral voltage ,
    V_SL_N = %.2f<%.2f V \n", abs(V_SL_N), atan( imag(
    V_SL_N), real(V_SL_N) ) );
79 printf("\n      Sending end line-to-line voltage ,
    V_SL_L = %.2f<%.2f V \n", abs(V_SL_L), atan( imag(
    V_SL_L), real(V_SL_L) ) );
80 printf("\n (c) sending-end current , I_S = %.2f<%.2f
    A \n", abs(I_S), atan( imag(I_S), real(I_S) ) );
81 printf("\n (d) sending-end power factor , cos_s =
    %.4f \n", cosd(theta_s));
82 printf("\n (e) sending-end power , P_S = %.5e W \n",
    P_S);
83 printf("\n (f) Power loss in line , P_L = %.5e W \n"
    , P_L);
84 printf("\n (g) Transmission line Efficiency ,      =
    %.1f Percentage\n", n);
85 printf("\n (h) Percentage of voltage regulation = %
    .1f Percentage \n", reg);
86 printf("\n (i) Sending-end charging current at no
    load , I_C = %.2f A \n", abs(I_C));
87 printf("\n (j) Receiving-end voltage rise at no load
    , V_RL_N = %.2f<%.2f V \n", abs(V_RL_N0), atan(
    imag(V_RL_N0), real(V_RL_N0)));
88 printf("\n      Line-to-line voltage at receiving end
    at no load , V_RL_L = %.2f<%.2f V \n", abs(V_RL_L0)
    ), atan( imag(V_RL_L0), real(V_RL_L0)));
89
90 printf("\n NOTE : ERROR : Change in answer because
    root(3) = 1.73 is considered in Textbook & change
    in     &     values ");
91 printf("\n But here sqrt(3) = 1.7320508 is
    considered \n");

```

Scilab code Exa 4.7 find equivalent pi T circuit and Nominal pi T circuit

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6
7 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
8
9 // EXAMPLE : 4.7 :
10 clear ; clc ; close ; // Clear the work space and
   console
11
12 // GIVEN DATA
13 R = 0.1858 // Line constant in /mi
14 f = 60 // frequency in Hertz
15 L = 2.60*10^-3 // Line constant in H/mi
16 C = 0.012*10^-6 // Line constant in F/mi
17 l = 150 // length of 3- transmission line in mi
18
19 // CALCULATIONS
20 z = R + %i*2*%pi*f*L ; // Impedance per unit length
   in /mi
21 y = %i*2*%pi*C*f ; // Admittance per unit length in
   S/mi
22 g = sqrt(y*z) ; // Propagation constant of line per
   unit length
23 g_1 = real(g) * l + %i * imag(g) * l ; // Propagation
   constant of line
24 Z_c = sqrt(z/y) ; // Characteristic impedance of
   line
25
26 A = cosh(g_1) ;
27 B = Z_c * sinh(g_1) ;
28 C = (1/Z_c) * sinh(g_1) ;
29 D = A ;
```

```

30 Z_pi = B ;
31 Y_pi_by2 = (A-1)/B ; // Unit in Siemens
32 Z = l * z ; // unit in ohms
33 Y = y * l ;
34 Y_T = C ;
35 Z_T_by2 = (A-1)/C ; // Unit in
36
37 // DISPLAY RESULTS
38 disp("EXAMPLE : 4.7 : SOLUTION :-") ;
39 printf("\n FOR EQUIVALENT- CIRCUIT ") ;
40 printf("\n Z_ = B = %.2f<%i f \n", abs(Z_pi),
        atan( imag(Z_pi), real(Z_pi) ) ) ;
41 printf("\n Y_ /2 = %.6f<%i f S \n", abs(Y_pi_by2),
        atan( imag(Y_pi_by2), real(Y_pi_by2) ) ) ;
42 printf("\n FOR NOMINAL- CIRCUIT ") ;
43 printf("\n Z = %.3f<%i f \n", abs(Z), atan( imag
        (Z), real(Z) ) ) ;
44 printf("\n Y/2 = %.6f<%i f S \n", abs(Y/2), atan(
        imag(Y/2), real(Y/2) ) ) ;
45 printf("\n FOR EQUIVALENT-T CIRCUIT ") ;
46 printf("\n Z_T/2 = %.2f<%i f \n", abs(Z_T_by2),
        atan( imag(Z_T_by2), real(Z_T_by2) ) ) ;
47 printf("\n Y_T = C = %.5f<%i f S \n", abs(Y_T),
        atan( imag(Y_T), real(Y_T) ) ) ;
48 printf("\n FOR NOMINAL-T CIRCUIT ") ;
49 printf("\n Z/2 = %.2f<%i f \n", abs(Z/2), atan(
        imag(Z/2), real(Z/2) ) ) ;
50 printf("\n Y = %.6f<%i f S \n", abs(Y), atan( imag(
        Y), real(Y) ) ) ;

```

Scilab code Exa 4.8 calculate attenuation phase change lamda v Vir Vrr Vr Vis Vrs

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
2 // TURAN GONEN

```

```

3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.8 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 V_RL_L = 138*10^3 ; // transmission line voltage in
    V
13 R = 0.1858 // Line constant in /mi
14 f = 60 // frequency in Hertz
15 L = 2.60*10^-3 // Line constant in H/mi
16 C = 0.012*10^-6 // Line constant in F/mi
17 pf = 0.85 // Lagging power factor
18 P = 50*10^6 // load in VA
19 l = 150 // length of 3- transmission line in mi
20
21 // CALCULATIONS
22 // For case (a)
23 z = R + %i*2*pi*f*L ; // Impedance per unit length
    in /mi
24 y = %i*2*pi*C*f ; // Admittance per unit length in
    S/mi
25 g = sqrt(y*z) ; // Propagation constant of line per
    unit length
26
27 // For case (b)
28 lamda = (2 * %pi)/imag(g) ; // Wavelength of
    propagation in mi
29 V = lamda * f ; // Velocity of propagation in mi/sec
30
31 // For case (c)
32 Z_C = sqrt(z/y) ;
33 V_R = V_RL_L/sqrt(3) ;
34 theta_R = acosd(pf) ;

```

```

35 I_R = P/(sqrt(3)*V_RL_L) * ( cosd(theta_R) - %i*sind
    (theta_R) ) ; // Receiving end current in A
36 V_R_incident = (1/2)*(V_R + I_R*Z_C) ; // Incident
    voltage at receiving end in V
37 V_R_reflected = (1/2)*(V_R - I_R*Z_C) ; // Reflected
    voltage at receiving end in V
38
39 // For case (d)
40 V_RL_N = V_R_incident + V_R_reflected ; // Line-to-
    neutral voltage at receiving end in V
41 V_RL_L = sqrt(3)*V_RL_N // Receiving end Line
    voltage in V
42
43 // For case (e)
44 g_l = real(g) * l + %i * imag(g) * l ; //
    Propagation constant of line
45 a = real(g) ; // a = is the attenuation constant
46 b = imag(g) ; // b = is the phase constant
47 V_S_incident = (1/2) * (V_R+I_R*Z_C) * exp(a*l) *
    exp(%i*b*l) ; // Incident voltage at sending end
    in V
48 V_S_reflected = (1/2) * (V_R-I_R*Z_C) * exp(-a*l) *
    exp(%i*(-b)*l) ; // Reflected voltage at sending
    end in V
49
50 // For case (f)
51 V_SL_N = V_S_incident + V_S_reflected ; // Line-to-
    neutral voltage at sending end in V
52 V_SL_L = sqrt(3)*V_SL_N ; // sending end Line
    voltage in V
53
54 // DISPLAY RESULTS
55 disp("EXAMPLE : 4.8 : SOLUTION :-") ;
56 printf("\n (a) Attenuation constant ,      = %.4f Np/
    mi \n",real(g)) ;
57 printf("\n     Phase change constant ,      = %.4f rad/
    mi \n",imag(g)) ;
58 printf("\n (b) Wavelength of propagation = %.2f mi \

```

```

        n",lamda) ;
59 printf("\n      velocity of propagation = %.2f mi/s \
        n",V) ;
60 printf("\n (c) Incident voltage receiving end , V_R(
    incident) = %.2f<%.2f V \n",abs(V_R_incident),
    atan(imag(V_R_incident),real(V_R_incident))*(180/
    %pi));
61 printf("\n      Receiving end reflected voltage , V_R(
    reflected) = %.2f<%.2f V \n",abs(V_R_reflected),
    atan(imag(V_R_reflected),real(V_R_reflected))
    *(180/%pi)) ;
62 printf("\n (d) Line voltage at receiving end ,
    V_RL_L = %d V \n",V_RL_L) ;
63 printf("\n (e) Incident voltage at sending end , V_S(
    incident) = %.2f<%.2f V \n",abs(V_S_incident),
    atan(imag(V_S_incident),real(V_S_incident))*(180/
    %pi)) ;
64 printf("\n      Reflected voltage at sending end ,
    V_S(reflected) = %.2f<%.2f V \n",abs(
    V_S_reflected),atan(imag(V_S_reflected),real(
    V_S_reflected))*(180/%pi)) ;
65 printf("\n (f) Line voltage at sending end , V_SL_L
    = %.2f V \n",abs(V_SL_L)) ;

```

Scilab code Exa 4.9 calculate SIL

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.9 :

```

```

9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 L = 2.60 * 10^-3 ; // Inductance of line in H/mi
13 R = 0.1858 ; // Resistance of line in /mi
14 C = 0.012 * 10^-6 ; // Capacitance in F/mi
15 kV = 138 ; // Transmission line voltage in kV
16 Z_c1 = 469.60085 // Characteristic impedance of line
    in . Obtained from example 4.6
17
18 // CALCULATIONS
19 Z_c = sqrt(L/C) ; // Approximate value of surge
    Impedance of line in ohm
20 SIL = kV^2/Z_c ; // Approximate Surge impedance
    loading in MW
21 SIL1 = kV^2/Z_c1 ; // Exact value of SIL in MW
22
23 // DISPLAY RESULTS
24 disp("EXAMPLE : 4.9 : SOLUTION :-") ;
25 printf("\n Approximate value of SIL of transmission
    line , SIL_app = %.3f MW\n",SIL) ;
26 printf("\n Exact value of SIL of transmission line ,
    SIL_exact = %.3f MW\n",SIL1) ;

```

Scilab code Exa 4.10 determine equ A B C D constant

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7

```

```

8 // EXAMPLE : 4.10 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 Z_1 = 10 * exp(%i*(30)*pi/180) ; // Impedance in
13 Z_2 = 40 * exp(%i*(-45)*pi/180) ; // Impedance in

14
15 // CALCULATIONS
16 P = [1 ,Z_1 ; 0 , 1]; // For network 1
17 Y_2 = 1/Z_2 ; // unit is S
18 Q = [1 0 ; Y_2 1]; // For network 2
19 EQ = P * Q ;
20
21 // DISPLAY RESULTS
22 disp("EXAMPLE : 4.10 : SOLUTION :-") ;
23 printf("\n Equivalent A , B , C , D constants are \n
   ") ;
24 printf("\n A_eq = %.3f<%i f \n" ,abs( EQ(1,1) ),atand
   ( imag(EQ(1,1)),real(EQ(1,1)) ) );
25 printf("\n B_eq = %.3f<%i f \n" ,abs( EQ(1,2) ),atand
   ( imag(EQ(1,2)),real(EQ(1,2)) ) );
26 printf("\n C_eq = %.3f<%i f \n" ,abs( EQ(2,1) ),atand
   ( imag(EQ(2,1)),real(EQ(2,1)) ) );
27 printf("\n D_eq = %.3f<%i f \n" ,abs( EQ(2,2) ),atand
   ( imag(EQ(2,2)),real(EQ(2,2)) ) );

```

Scilab code Exa 4.11 determine equ A B C D constant

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION

```

```

5
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.11 :
9 clear ; clc ; close ; // Clear the work space and
  console
10
11 // GIVEN DATA
12 Z_1 = 10*exp(%i*(30)*%pi/180) ; // Impedance in
13 Z_2 = 40*exp(%i*(-45)*%pi/180) ; // Impedance in
14 Y_2 = 1/Z_2 ;
15 A_1 = 1 ;
16 B_1 = Z_1 ;
17 C_1 = 0 ;
18 D_1 = 1 ;
19 A_2 = 1 ;
20 B_2 = 0 ;
21 C_2 = Y_2 ;
22 D_2 = 1 ;
23
24 // CALCULATIONS
25 P = [A_1 B_1 ; C_1 D_1] ; // For network 1
26 Q = [A_2 B_2 ; C_2 D_2] ; // For network 2
27 A_eq = ( A_1*B_2 + A_2*B_1 )/( B_1 + B_2 ) ; //
  Constant A
28 B_eq = ( B_1*B_2 )/(B_1 + B_2) ; // Constant B
29 C_eq = C_1 + C_2 + ( (A_1 - A_2) * (D_2 - D_1)/(B_1 +
  B_2) ) ; // Constant C
30 D_eq = ( D_1*B_2 + D_2*B_1 )/(B_1+B_2) ; // Constant
  D
31
32 // DISPLAY RESULTS
33 disp("EXAMPLE : 4.11 : SOLUTION :-") ;
34 printf("\n Equivalent A , B , C , D constants are \n
  ") ;
35 printf("\n A_eq = %.2f<%.f \n" , abs(A_eq),atand( imag
  (A_eq),real(A_eq))) ;
36 printf("\n B_eq = %.2f<%.f \n" , abs(B_eq),atand( imag

```

```

(B_eq),real(B_eq) )) ;
37 printf("\n C_eq = %.3f<%f \n",abs(C_eq),atand( imag
(C_eq),real(C_eq) )) ;
38 printf("\n D_eq = %.2f<%f \n",abs(D_eq),atand( imag
(D_eq),real(D_eq) )) ;

```

Scilab code Exa 4.12 calculate Is Vs Zin P var

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.12 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 Z = 2.07 + 0.661 * %i ; // Line impedance in
13 V_L = 2.4 * 10^3 ; // Line voltage in V
14 p = 200 * 10^3; // Load in VA
15 pf = 0.866 ; // Lagging power factor
16
17 // CALCULATIONS
18 // for case (a)
19 A = 1 ;
20 B = Z ;
21 C = 0 ;
22 D = A ;
23 theta = acosd(pf) ;
24 S_R = p * ( cosd(theta) + %i * sind(theta) ) ; // 
   Receiving end power in VA

```

```

25 I_L1 = S_R/V_L ;
26 I_L = conj(I_L1) ;
27 I_S = I_L ; // sending end current in A
28 I_R = I_S ; // Receiving end current in A
29
30 // for case (b)
31 Z_L = V_L/I_L ; // Impedance in
32 V_R = Z_L * I_R ;
33 V_S = A * V_R + B * I_R ; // sending end voltage in
   V
34 P = [A B ; C D] * [V_R ; I_R] ;
35
36 // for case (c)
37 V_S = P(1,1) ;
38 I_S = P(2,1) ;
39 Z_in = V_S/I_S ; // Input impedance in
40
41 // for case (d)
42 S_S = V_S * conj(I_S) ;
43 S_L = S_S - S_R ; // Power loss of line in VA
44
45 // DISPLAY RESULTS
46 disp("EXAMPLE : 4.12 : SOLUTION :-") ;
47 printf("\n (a) Sending-end current , I_S = %.2f<% .2f f
          A \n", abs(I_S), atand( imag(I_S), real(I_S) )) ;
48 printf("\n (b) Sending-end voltage , V_S = %.2f<% .2f f
          V \n", abs(V_S), atand( imag(V_S), real(V_S) )) ;
49 printf("\n (c) Input impedance , Z_in = %.2f<% .2f f
          \n", abs(Z_in), atand( imag(Z_in), real(Z_in) )) ;
50 printf("\n (d) Real power loss in line , S_L = %.2f
          W \n", real(S_L)) ;
51 printf("\n      Reactive power loss in line , S_L = %
          .2 f var \n", imag(S_L)) ;

```

Scilab code Exa 4.13 calculate SIL Pmax Qc Vroc

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6
7 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
8
9 // EXAMPLE : 4.13 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 KV = 345 ; // Transmission line voltage in kV
13 V_R = KV ;
14 V_S = KV ;
15 x_L = 0.588 ; // Inductive reactance in /mi/phase
16 b_c = 7.20*10^-6 ; // susceptance S phase to neutral
    per phase
17 l = 200 ; // Total line length in mi
18
19 // CALCULATIONS
20 // for case (a)
21 x_C = 1/b_c ; // /mi/phase
22 Z_C = sqrt(x_C * x_L) ;
23 SIL = KV^2/Z_C ; // Surge impedance loading in MVA/
    mi . [1MVA = 1MW]
24 SIL1 = (KV^2/Z_C) * l ; // Surge impedance loading
    of line in MVA . [1MVA = 1MW]
25
26 // for case (b)
27 delta = 90 ; // Max 3- theoretical steady-state
    power flow limit occurs for = 90 degree
28 X_L = x_L * l ; // Inductive reactance /phase
29 P_max = V_S * V_R * sind(delta)/(X_L) ;
30
31 // for case (c)
32 Q_C = V_S^2 * (b_c * 1/2) + V_R^2 *( b_c * 1/2) ; //

```

```

    Total 3- magnetizing var in Mvar
33
34 // for case (d)
35 g = %i * sqrt(x_L/x_C) ; // rad/mi
36 g_l = g * l ; // rad
37 V_R_oc = V_S / cosh(g_l) ; // Open-circuit receiving
   -end voltage in kV
38 X_C = x_C * 2 / l ;
39 V_R_oc1 = V_S * ( - %i * X_C/( - %i * X_C + %i * X_L
   ) ) ; // Alternative method to find Open-circuit
   receiving-end voltage in kV
40
41 // DISPLAY RESULTS
42 disp("EXAMPLE : 4.13 : SOLUTION :-") ;
43 printf("\n (a) Total 3- SIL of line , SIL = %.2f
   MVA/mi \n",SIL) ;
44 printf("\n      Total 3- SIL of line for total line
   length , SIL = %.2f MVA \n",SIL1) ;
45 printf("\n (b) Maximum 3- theortical steady-state
   power flow limit , P_max = %.2f MW \n",P_max) ;
46 printf("\n (c) Total 3- magnetizing var generation
   by line capacitance , Q_C = %.2f Mvar \n",Q_C) ;
47 printf("\n (d) Open-circuit receiving-end voltage if
   line is open at receiving end , V_R_oc = %.2f kV
   \n",V_R_oc) ;
48 printf("\n      From alternative method ,") ;
49 printf("\n      Open-circuit receiving-end voltage if
   line is open at receiving end , V_R_oc = %.2f kV
   \n",V_R_oc1) ;

```

Scilab code Exa 4.14 calculate SIL Pmax QC cost Vroc

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN

```

```

3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.14 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 KV = 345 ; // Transmission line voltage in kV
13 V_R = KV ; // Sending end voltage in kV
14 x_L = 0.588 ; // Inductive reactance in /mi/phase
15 b_c = 7.20*10^-6 ; // susceptance S phase to neutral
    per phase
16 l = 200 ; // Total line length in mi
17 per = 60/100 ; // 2 shunt reactors absorb 60% of
    total 3- magnetizing var
18 cost = 10 ; // cost of each reactor is $10/kVA
19
20 // CALCULATIONS
21 // For case (a)
22 x_C = 1/b_c ; // /mi/phase
23 Z_C = sqrt(x_C * x_L) ;
24 SIL = KV^2/Z_C ; // Surge impedance loading in MVA/
    mi
25 SIL1 = (KV^2/Z_C) * l ; // Surge impedance loading
    of line in MVA . [1MVA = 1MW]
26
27 // For case (b)
28 delta = 90 ; // Max 3- theoretical steady-state
    power flow limit occurs for = 90 degree
29 V_S = V_R ; // sending end voltage in kV
30 X_L = x_L * l ; // Inductive reactance /phase
31 P_max = V_S * V_R * sind(delta)/(X_L) ;
32
33 // For case (c)
34 Q_C = V_S^2 * (b_c * 1/2) + V_R^2 *( b_c * 1/2) ; //

```

```

        Total 3- magnetizing var in Mvar
35 Q = (1/2) * per * Q_C ; // 3- megavoltampere
                           rating of each reactor . Q = (1/2)*Q_L
36
37 // For case (d)
38 Q_L1 = Q * 10^3 ; // Total 3- magnetizing var in
                      Kvar
39 T_cost = Q_L1 * cost ; // Cost of each reactor in $
40
41 // For case (e)
42 g = %i * sqrt(x_L * (1-per)/x_C) ; // rad/mi
43 g_l = g * l ; // rad
44 V_R_oc = V_S/cosh(g_l) ; // Open circuit receiving-
                           end voltage in kV
45 X_L = x_L *l ;
46 X_C = (x_C * 2) / (1 * (1 - per)) ;
47 V_R_oc1 = V_S * ( -%i*X_C/(-%i*X_C + %i*X_L) ) ; //
                           Alternative method to find Open-circuit receiving-
                           end voltage in kV
48
49 // DISPLAY RESULTS
50 disp("EXAMPLE : 4.14 : SOLUTION :-") ;
51 printf("\n (a) Total 3-phase SIL of line , SIL = %.2
          f MVA/mi \n",SIL) ;
52 printf("\n      Total 3- SIL of line for total line
          length , SIL = %.2 f MVA \n",SIL1) ;
53 printf("\n (b) Maximum 3-phase theoretical power flow
          , P_max = %.2 f MW \n",P_max) ;
54 printf("\n (c) 3-phase MVA rating of each reactor ,
          (1/2)Q_L = %.2 f MVA \n",Q) ;
55 printf("\n (d) Cost of each reactor at $10/kVA = $ %
          .2 f \n",T_cost) ;
56 printf("\n (e) Open circuit receiving voltage ,
          V_Roc= %.2 f kV \n",V_R_oc) ;
57 printf("\n      From alternative method ,") ;
58 printf("\n      Open-circuit receiving-end voltage if
          line is open at receiving end , V_R_oc = %.2 f kV
          \n",V_R_oc1) ;

```

Scilab code Exa 4.15 calculate La XL Cn Xc

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
7
8 // EXAMPLE : 4.15 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 D_12 = 26 ; // distances in feet
13 D_23 = 26 ; // distances in feet
14 D_31 = 52 ; // distances in feet
15 d = 12 ; // Distance b/w 2 subconductors in inches
16 f = 60 ; // frequency in Hz
17 kv = 345 ; // voltage base in kv
18 p = 100 ; // Power base in MVA
19 l = 200 ; // length of line in km
20
21 // CALCULATIONS
22 // For case (a)
23 D_S = 0.0435 ; // from A.3 Appendix A . Geometric
    mean radius in feet
24 D_bS = sqrt(D_S * 0.3048 * d * 0.0254) ; // GMR of
    bundled conductor in m .[1 ft = 0.3048 m ; 1 inch
    = 0.0254 m]
25 D_eq = (D_12 * D_23 * D_31 * 0.3048^3)^(1/3) ; //
    Equ GMR in meter
26 L_a = 2 * 10^-7 * log(D_eq/D_bS) ; // Inductance in H
```

```

    /meter
27
28 // For case (b)
29 X_L = 2 * %pi * f * L_a ; // inductive reactance/
   phase in ohms/m
30 X_L0 = X_L * 10^3 ; // inductive reactance/phase in
   ohms/km
31 X_L1 = X_L0 * 1.609 ; // inductive reactance/phase in
   ohms/mi [1 mi = 1.609 km]
32
33 // For case (c)
34 Z_B = kv^2 / p ; // Base impedance in
35 X_L2 = X_L0 * 1/Z_B ; // Series reactance of line in
   pu
36
37 // For case (d)
38 r = 1.293*0.3048/(2*12) ; // radius in m . outside
   diameter is 1.293 inch given in A.3
39 D_bsC = sqrt(r * d * 0.0254) ;
40 C_n = 55.63 * 10^-12/log(D_eq/D_bsC) ; //
   capacitance of line in F/m
41
42 // For case (e)
43 X_C = 1/( 2 * %pi * f * C_n ) ; // capacitive
   reactance in ohm-m
44 X_CO = X_C * 10^-3 ; // capacitive reactance in ohm-
   km
45 X_C1 = X_CO/1.609 ; // capacitive reactance in ohm-
   mi
46
47 // DISPLAY RESULTS
48 disp("EXAMPLE : 4.15 : SOLUTION :-") ;
49 printf("\n (a) Average inductance per phase , L_a =
   %.4e H/m \n",L_a) ;
50 printf("\n (b) Inductive reactance per phase , X_L =
   %.4f /km \n",X_L0) ;
51 printf("\n      Inductive reactance per phase , X_L =
   %.4f /mi \n",X_L1) ;

```

```
52 printf("\n (c) Series reactance of line , X_L = %.4f  
      pu \n",X_L2) ;  
53 printf("\n (d) Line-to-neutral capacitance of line ,  
      C_n = %.4e F/m \n",C_n);  
54 printf("\n (e) Capacitive reactance to neutral of  
      line , X_C = %.3e -km \n",X_C0) ;  
55 printf("\n      Capacitive reactance to neutral of  
      line , X_C = %.3e -mi \n",X_C1) ;
```

Chapter 5

UNDERGROUND POWER TRANSMISSION AND GAS INSULATED TRANSMISSION LINES

Scilab code Exa 5.1 calculate Emax Emin r

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
7 // GAS-INSULATED TRANSMISSION LINES
8 // EXAMPLE : 5.1 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11 // GIVEN DATA
12 d = 2 ; // Diameter of conductor in cm
```

```

13 D = 5 ; // Inside diameter of lead sheath in cm
14 V = 24.9 ; // Line-to-neutral voltage in kV
15
16 // CALCULATIONS
17 // For case (a)
18 r = d/2 ;
19 R = D/2 ;
20 E_max = V/( r * log(R/r) ) ; // Maximum electric
   stress in kV/cm
21 E_min = V/( R * log(R/r) ) ; // Minimum electric
   stress in kV/cm
22
23 // For case (b)
24 r_1 = R/2.718 ; // Optimum conductor radius in cm .
   From equ 5.15
25 E_max1 = V/( r_1 * log(R/r_1) ) ; // Min value of
   max stress in kV/cm
26
27 // DISPLAY RESULTS
28 disp("EXAMPLE : 5.1 : SOLUTION :-") ;
29 printf("\n (a) Maximum value of electric stress ,
   E_max = %.2f kV/cm \n",E_max) ;
30 printf("\n      Minimum value of electric stress ,
   E_min = %.2f kV/cm \n",E_min) ;
31 printf("\n (b) Optimum value of conductor radius ,
   r = %.2f cm \n",r_1) ;
32 printf("\n      Minimum value of maximum stress ,
   E_max = %.2f kV/cm \n",E_max1) ;

```

Scilab code Exa 5.2 calculate potential gradient E1

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS

```

```

4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
    GAS-INSULATED TRANSMISSION LINES
7
8 // EXAMPLE : 5.2 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 r = 1 ; // Radius of conductor in cm
13 t_1 = 2 ; // Thickness of insulation layer in cm
14 r_1 = r + t_1 ;
15 r_2 = 2 ; // Thickness of insulation layer in cm .
    r_2 = t_1 = t_2
16 R = r_1 + r_2 ;
17 K_1 = 4 ; // Inner layer Dielectric constant
18 K_2 = 3 ; // Outer layer Dielectric constant
19 kv = 19.94 ; // potential difference b/w inner &
    outer lead sheath in kV
20
21 // CALCULATIONS
22 //  $E_1 = 2q/(r * K_1)$  &  $E_2 = 2q/(r_1 * K_2)$  . Let  $E = E_1/E_2$ 
23 E = ( r_1 * K_2 ) / ( r * K_1 ) ; //  $E = E_1/E_2$ 
24 V_1 = poly(0, 'V_1') ; // defining unknown V_1
25 E_1 = V_1 / ( r * log(r_1/r) ) ;
26 V_2 = poly(0, 'V_2') ; // defining unknown V_2
27 V_2 = kv - (V_1) ;
28 E_2 = V_2 / ( r_1 * log(R/r_1) ) ;
29 E_3 = E_1/E_2 ;
30 // Equating  $E = E_3$  . we get the value of V_1
31 V_1 = 12.30891068 ; // Voltage in kV
32 E_1s = V_1 / ( r * log(r_1/r) ) ; // Potential
    gradient at surface of conductor in kV/cm .  $E_1 = E_1s$ 
33
34 // DISPLAY RESULTS

```

```
35 disp("EXAMPLE : 5.2 : SOLUTION :-") ;
36 printf("\n Potential gradient at the surface of
conductor , E_1 = %.2f kV/cm \n",E_1s) ;
```

Scilab code Exa 5.3 calculate Ri Power loss

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
GAS-INSULATED TRANSMISSION LINES
7
8 // EXAMPLE : 5.3 :
9 clear ; clc ; close ; // Clear the work space and
console
10
11 // GIVEN DATA
12 D = 1.235 ; // Inside diameter of sheath in inch
13 d = 0.575 ; // Conductor diameter in inch
14 kv = 115 ; // Voltage in kV
15 l = 6000 ; // Length of cable in feet
16 r_si = 2000 ; // specific insulation resistance is
2000 M /1000 ft . From Table 5.2
17
18 // CALCULATIONS
19 // For case (a)
20 r_si0 = r_si * 1/1000 ;
21 R_i = r_si0 * log10 (D/d) ; // Total Insulation
resistance in M
22
23 // For case (b)
24 P = kv^2/R_i ; // Power loss due to leakage current
```

```

    in W
25
26 // DISPLAY RESULTS
27 disp("EXAMPLE : 5.3 : SOLUTION :-") ;
28 printf("\n (a) Total insulation resistance at 60
degree F , R_i= %.2f M \n",R_i) ;
29 printf("\n (b) Power loss due to leakage current , V
^2/R_i = %.4f W \n",P) ;
30
31 printf("\n NOTE : ERROR : Mistake in textbook case (
a) \n") ;

```

Scilab code Exa 5.4 calculate charging current Ic

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
   GAS-INSULATED TRANSMISSION LINES
7
8 // EXAMPLE : 5.4 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 C_a = 2 * 10^-6 ; // Capacitance b/w two conductors
   in F/mi
13 l = 2 ; // length in mi
14 f = 60 ; // Frequency in Hz
15 V_L_L = 34.5 * 10^3 ; // Line-to-line voltage in V
16
17 // CALCULATIONS

```

```

18 C_a1 = C_a * l ; // Capacitance for total cable
    length in F
19 C_N = 2 * C_a1 ; // capacitance of each conductor to
    neutral in F . From equ 5.56
20 V_L_N = V_L_L/sqrt(3) ; // Line-to-neutral voltage
    in V
21 I_c = 2 * %pi * f * C_N * (V_L_N) ; // Charging
    current of cable in A
22
23 // DISPLAY RESULTS
24 disp("EXAMPLE : 5.4 : SOLUTION :-") ;
25 printf("\n Charging current of the cable , I_c = %.2
    f A \n",I_c) ;

```

Scilab code Exa 5.5 calculate Ic Is pf

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
    GAS-INSULATED TRANSMISSION LINES
7
8 // EXAMPLE : 5.5 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 C_a = 0.45 * 10^-6 ; // Capacitance b/w two
    conductors in F/mi
13 l = 4 ; // length of cable in mi
14 f = 60 ; // Freq in Hz
15 V_L_L = 13.8 * 10^3 ; // Line-to-line voltage in V

```

```

16 pf = 0.85 ; // lagging power factor
17 I = 30 ; // Current drawn by load at receiving end
    in A
18
19 // CALCULATIONS
20 // For case (a)
21 C_a1 = C_a * l ; // Capacitance for total cable
    length in F
22 C_N = 2 * C_a1 ; // capacitance of each conductor to
    neutral in F
23 V_L_N = V_L_L/sqrt(3) ; // Line-to-neutral voltage
    in V
24 I_c = 2 * %pi * f * C_N * (V_L_N) ; // Charging
    current in A
25 I_c1 = %i * I_c ; // polar form of Charging current
    in A
26
27 // For case (b)
28 phi_r = acosd(pf) ; // pf angle
29 I_r = I * ( cosd(phi_r) - sind(phi_r) * %i ) ; //
    Receiving end current in A
30 I_s = I_r + I_c1 ; // sending end current in A
31
32 // For case (c)
33 pf_s = cosd( atan( imag(I_s), real(I_s) ) ) ; //
    Lagging pf of sending-end
34
35 // DISPLAY RESULTS
36 disp("EXAMPLE : 5.5 : SOLUTION :-") ;
37 printf("\n (a) Charging current of feeder , I_c = %
    .2f A \n",I_c) ;
38 printf("\n     Charging current of feeder in complex
    form , I_c = i*%.2f A \n",imag(I_c1)) ;
39 printf("\n (b) Sending-end current , I_s = %.2f<% .2f
    A\n",abs(I_s),atan( imag(I_s), real(I_s) )) ;
40 printf("\n (c) Sending-end power factor ,cos -s =
    %.2f Lagging power factor \n",pf_s) ;

```

Scilab code Exa 5.6 calculate Geometric factor G1 Ic

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
7 // GAS-INSULATED TRANSMISSION LINES
8 // EXAMPLE : 5.6 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 f = 60 ; // Freq in Hz
13 V_L_L = 138 ; // Line-to-line voltage in kV
14 T = 11/64 ; // Thickness of conductor insulation in
    inches
15 t = 5/64 ; // Thickness of belt insulation in inches
16 d = 0.575 ; // Outside diameter of conductor in
    inches
17
18 // CALCULATIONS
19 // For case (a)
20 T_1 = (T + t)/d ; // To find the value of geometric
    factor G for a single-conductor cable
21 G_1 = 2.09 ; // From table 5.3 , by interpolation
22 sf = 0.7858 ; // sector factor obtained for T_1 from
    table 5.3
23 G = G_1 * sf ; // real geometric factor
24
25 // For case (b)
```

```

26 V_L_N = V_L_L/sqrt(3) ; // Line-to-neutral voltage
   in V
27 K = 3.3 ; // Dielectric constant of insulation for
   impregnated paper cable
28 I_c = 3 * 0.106 * f * K * V_L_N/(1000 * G) ; //
   Charging current in A/1000 ft
29
30 // DISPLAY RESULTS
31 disp("EXAMPLE : 5.6 : SOLUTION :-") ;
32 printf("\n (a) Geometric factor of cable using table
      5.3 , G_1 = %.3f \n",G) ;
33 printf("\n (b) Charging current , I_c = %.3f A/1000
      ft \n",I_c) ;

```

Scilab code Exa 5.7 calculate Emax C Ic Ri Plc Pd1 Pdh

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
   GAS-INSULATED TRANSMISSION LINES
7
8 // EXAMPLE : 5.7 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 V_L_N = 7.2 ; // Line-to-neutral voltage in kV
13 d = 0.814 ; // Conductor diameter in inches
14 D = 2.442 ; // inside diameter of sheath in inches
15 K = 3.5 ; // Dielectric constant
16 pf = 0.03 ; // power factor of dielectric

```

```

17 l = 3.5 ; // length in mi
18 f = 60 ; // Freq in Hz
19 u = 1.3 * 10^7 ; // dielectric resistivity of
    insulation in M -cm
20
21 // CALCULATIONS
22 // For case (a)
23 r = d * 2.54/2 ; // conductor radius in cm . [1 inch
    = 2.54 cm]
24 R = D * 2.54/2 ; // Inside radius of sheath in cm
25 E_max = V_L_N/( r * log(R/r) ) ; // max electric
    stress in kV/cm
26
27 // For case (b)
28 C = 0.0388 * K/( log10 (R/r) ) ; // capacitance of
    cable in F /mi . From equ 5.29
29 C_1 = C * l ; // capacitance of cable for total
    length in F
30
31 // For case (c)
32 V_L_N1 = 7.2 * 10^3 ; // Line-to-neutral voltage in
    V
33 C_2 = C_1 * 10^-6 ; // capacitance of cable for
    total length in F
34 I_c = 2 * %pi * f * C_2 * (V_L_N1) ; // Charging
    current in A
35
36 // For case (d)
37 l_1 = l * 5280 * 12 * 2.54 ; // length in cm . [1 mi
    = 5280 feet] ; [1 feet = 12 inch]
38 R_i = u * log(R/r)/( 2 * %pi * l_1) ; // Insulation
    resistance in M
39
40 // For case (e)
41 P_lc = V_L_N^2/R_i ; // power loss in W
42
43 // For case (f)
44 P_dl = 2 * %pi * f * C_1 * V_L_N^2 * pf ; // Total

```

```

        dielectric loss in W
45
46 // For case (g)
47 P_dh = P_dl - P_lc ; // dielectric hysteresis loss
    in W
48
49 // DISPLAY RESULTS
50 disp("EXAMPLE : 5.7 : SOLUTION :-") ;
51 printf("\n (a) Maximum electric stress occurring in
    cable dielectric , E_max = %.2f kV/cm \n",E_max)
    ;
52 printf("\n (b) Capacitance of cable , C = %.4f F \
    n",C_1) ;
53 printf("\n (c) Charging current of cable , I_c = %.3
    f A \n",I_c) ;
54 printf("\n (d) Insulation resistance , R_i = %.2f
    M \n",R_i) ;
55 printf("\n (e) Power loss due to leakage current ,
    P_lc = %.2f W \n",P_lc) ;
56 printf("\n (f) Total dielectric loss , P_dl = %.2f W
    \n",P_dl) ;
57 printf("\n (g) Dielectric hysteresis loss , P_dh = %
    .2f W \n",P_dh) ;

```

Scilab code Exa 5.8 calculate Rdc Reff percent reduction

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
    GAS-INSULATED TRANSMISSION LINES
7

```

```

8 // EXAMPLE : 5.8 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 l = 3 ; // underground cable length in mi
13 f = 60 ; // frequency in hertz
14
15 // CALCULATIONS
16 // For case (a)
17 R_dc = 0.00539 ; // dc resistance of cable in
    /1000ft , From table 5.5
18 R_dc1 = (R_dc/1000) * 5280 * 3 ; // Total dc
    resistance in . [1 mi = 5280 feet]
19
20 // For case (b)
21 s_e = 1.233 ; // skin effect coefficient
22 R_eff = s_e * R_dc1 ; // Effective resistance in
23 percentage = ((R_eff - R_dc1)/(R_dc1)) * 100 ; //
    skin effect on effective resistance in %
24
25 // DISPLAY RESULTS
26 disp("EXAMPLE : 5.8 : SOLUTION :-") ;
27 printf("\n (a) Total dc resistance of the conductor
    , R_dc = %.4f \n", R_dc1) ;
28 printf("\n (b) Effective resistance at 60 hz , R_eff
    = %.4f \n", R_eff) ;
29 printf("\n     Skin effect on the Effective
    resistance in percent at 60 hz , R_eff = %.1f
    percent greater than for direct current\n",
    percentage) ;
30 printf("\n (c) Percentage of reduction in cable
    ampacity in part (b) = %.1f percent \n",
    percentage) ;

```

Scilab code Exa 5.9 calculate Xm Rs deltaR Ra ratio Ps

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
7 // GAS-INSULATED TRANSMISSION LINES
8 // EXAMPLE : 5.9 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11 // GIVEN DATA
12 kV = 35 ; // voltage in kV
13 f = 60 ; // operating frequency of cable in hertz
14 d = 0.681 ; // diameter of conductor in inches
15 t_i = 345 ; // Insulation thickness in cmil
16 t_s = 105 ; // Metal sheet thickness in cmil
17 r_c = 0.190 ; // Conductor ac resistance in /mi
18 l = 10 ; // Length of cable in mi
19
20 // CALCULATIONS
21 // For case (a)
22 T_i = t_i/1000 ; // insulation thickness in inch
23 T_s = t_s/1000 ; // Metal sheet thickness in inch
24 r_i = (d/2) + T_i ; // Inner radius of metal sheath
25 // in inches
26 r_0 = r_i + T_s ; // Outer radius of metal sheath in
27 // inches
28 S = r_i + r_0 + T_s ; // Spacing b/w conductor
29 // centers in inches
30 X_m = 0.2794 * (f/60) * log10 ( 2*S/(r_0 + r_i) ) ;
31 // Mutual reactance b/w conductor & sheath per
32 // phase in /mi . From Equ 5.78
33 X_m1 = X_m * l ; // Mutual reactance b/w conductor &
```

```

sheath in /phase
29
30 // For case (b)
31 r_s = 0.2/((r_0+r_i)*(r_0-r_i)) ; // sheet
      resistance per phase in /mi/phase . From equ
      5.79
32 r_s1 = r_s * l ; // sheet resistance per phase in
      /phase
33
34 // For case (c)
35 d_r = r_s * (X_m^2)/((r_s)^2 + (X_m)^2) ; //
      increase in conductor resistance due to sheath
      current in /mi/phase . From equ 5.77
36 d_r1 = d_r * l ; // // increase in conductor
      resistance due to sheath current in /phase
37
38 // For case (d)
39 r_a = r_c + (r_s * X_m^2)/((r_s)^2 + (X_m)^2) ;
      // Total positive or negative sequence resistance
      including sheath current effects in /mi/phase
      . From equ 5.84
40 r_a1 = r_a * l ; // Total positive or negative
      sequence resistance including sheath current
      effects in /phase
41
42 // For case (e)
43 ratio = d_r/r_c ; // ratio = sheath loss/conductor
      loss
44
45 // For case (f)
46 I = 400 ; // conductor current in A ( given for case
      (f) )
47 P_s = 3 * (I^2) * (r_s * X_m^2)/(r_s^2 + X_m^2) ;
      // For three phase loss in W/mi
48 P_s1 = P_s * l ; // Total sheath loss of feeder in
      Watts
49
50 // DISPLAY RESULTS

```

```

51 disp("EXAMPLE : 5.9 : SOLUTION :-") ;
52 printf("\n (a) Mutual reactance b/w conductors &
      sheath , X_m = %.5f /mi/phase \n",X_m) ;
53 printf("\n      or Mutual reactance b/w conductors &
      sheath , X_m = %.4f /phase \n",X_m1) ;
54 printf("\n (b) Sheath resistance of cable , r_s = %
      .4f /mi/phase \n",r_s) ;
55 printf("\n      or Sheath resistance of cable , r_s = %
      %.3f /phase \n",r_s1) ;
56 printf("\n (c) Increase in conductor resistance due
      to sheath currents , r = %.5f /mi/phase \n",
      d_r) ;
57 printf("\n      or Increase in conductor resistance
      due to sheath currents , r = %.4f /phase \n",
      d_r1) ;
58 printf("\n (d) Total resistance of conductor
      including sheath loss , r_a = %.5f /mi/phase \n
      ",r_a) ;
59 printf("\n      or Total resistance of conductor
      including sheath loss , r_a = %.4f /phase \n",
      r_a1) ;
60 printf("\n (e) Ratio of sheath loss to conductor
      loss , Ratio = %.4f \n",ratio) ;
61 printf("\n (f) Total sheath loss of feeder if
      current in conductor is 400A , P_s = %.2f W \n",
      P_s1) ;
62
63 printf("\n NOTE : ERROR : There are mistakes in some
      units in the Textbook \n") ;

```

Scilab code Exa 5.10 calculate zero sequence impedance Z00 Z0 Z0a

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN

```

```

3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
    GAS-INSULATED TRANSMISSION LINES
7
8 // EXAMPLE : 5.10 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 f= 60 ; // frequency in hertz
13 t = 245 ; // insulation thickness in mils
14 t_s = 95 ; // Lead/metal sheath thickness in mils
15 d = 0.575 ; // diameter of conductor in inches
16 r_s = 1.72 ; // sheath resistance in /mi
17 r_a = 0.263 ; // Conductor resistance in /mi
18 r = 100 ; // earth resistivity in -mi
19 D_s = 0.221 ; // GMR of one conductor in inches
20 D_ab = 24 ; // distance b/w conductor a & b in inch
    . refer fig 5.30
21 D_bc = 24 ; // distance b/w conductor b & c in inch
    . refer fig 5.30
22 D_ca = 48 ; // distance b/w conductor c & a in inch
    . refer fig 5.30
23
24 // CALCULATIONS
25 T = t/1000 ; // insulation thickness in inch . [1
    mils = 0.001 inch]
26 T_s = t_s/1000 ; // Lead/metal sheath thickness in
    mils
27 r_i = (d/2) + T ; // Inner radius of metal sheath in
    inches
28 r_o = r_i + T_s ; // Outer radius of metal sheath in
    inches
29 r_e = 0.00476 * f ; // AC resistance of earth
    return in /mi
30 D_e = 25920 * sqrt(r/f) ; // Equivalent depth of

```

```

        earth return path in inches
31 D_eq = (D_ab*D_bc*D_ca)^(1/3) ; // Mean distance
      among conductor centers in inches
32 Z_0a = (r_a + r_e) + (%i) * (0.36396) * log(D_e/(
      D_s*D_eq^2)^(1/3))) ;
33 D_s_3s = (D_eq^2 * (r_0+r_i)/2)^(1/3) ; // GMR of
      conducting path composed of 3 sheaths in parallel
      in inches
34 Z_0s = (r_s + r_e) + (%i) * 0.36396 * log (D_e/
      D_s_3s) ; // Zero sequence impedance of sheath in
      inches
35 D_m_3c_3s = D_s_3s ; // Zero sequence mutual
      impedance b/w conductors & sheaths in inches
36 Z_0m = r_e + (%i)*(0.36396)*log(D_e/D_m_3c_3s) ;
37
38 // For case (a)
39 Z_00 = Z_0a - (Z_0m^2/Z_0s) ; // Total zero sequence
      impedance when ground and return paths are
      present in /mi/phase
40
41 // For case (b)
42 Z_0 = Z_0a + Z_0s - 2*Z_0m ; // Total zero sequence
      impedance when there is only sheath return path
      in /mi/phase
43
44 // For case (c)
45 Z_01 = Z_0a ; // Total zero sequence impedance when
      there is only ground return path in /mi/phase
46
47 // DISPLAY RESULTS
48 disp("EXAMPLE : 5.10 : SOLUTION :-") ;
49 printf("\n (a) Total zero sequence impedance when
      both ground & return paths are present , Z_00 = %
      .3f<%1f /mi/phase \n",abs(Z_00),atand(imag(
      Z_00),real(Z_00))) ;
50 printf("\n (b) Total zero sequence impedance when
      there is only sheath return path , Z_0 = %.3f<%1
      f /mi/phase \n",abs(Z_0),atand(imag(Z_0),real(

```

```

        Z_0))) ;
51 printf("\n (c) Total zero sequence impedance when
      there is only ground return path , Z_0a = %.4f<%
      .1f /mi/phase \n",abs(Z_01),atand(imag(Z_01),
      real(Z_01))) ;
52
53 printf("\n NOTE : ERROR : There are mistakes in
      units in the Textbook \n") ;

```

Scilab code Exa 5.11 calculate C0 C1 C2 X0 X1 X2 I0 I1 I2

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
   GAS-INSULATED TRANSMISSION LINES
7
8 // EXAMPLE : 5.11 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 f= 60 ; // frequency in hertz
13 T = 0.175 ; // insulation thickness in inches
14 d = 0.539 ; // diameter of conductor in inches
15 G = 0.5 ; // Geometric factor from fig 5.3
16 K = 3.7 ; // Dielectric constant
17 V_LL = 13.8 ; // Line-to-line voltage in kV
18
19 // CALCULATIONS
20 D = d + 2 * T ; // Inside diameter of sheath in
   inches

```

```

21 G = 2.303 * log10 (D/d) ; // Geometric factor for a
    single conductor
22 sf = 0.710 ; // sector factor From Table 5.3 . For (
    T+t/d) obtained
23 V_LN = V_LL/sqrt(3) ; // Line-to-neutral voltage in
    kV
24
25 // For case (a)
26 C_0 = 0.0892 * K/(G * sf) ; // shunt capacitances in
    F /mi/phase . C_0 = C_1 = C_2 . From equ 5.161
27
28 // For case (b)
29 X_0 = 1.79 * G * sf/( f * K ) ; // shunt capacitive
    reactance in M /mi/phase .X_0 = X_1 = X_2. From
    equ 5.162
30
31 // For case (c)
32 I_0 = 0.323 * f * K * V_LN/( 1000 * G * sf ) ; //
    Charging current in A/mi/phase .I_0 = I_1 = I_2 .
    From equ 5.163
33
34 // DISPLAY RESULTS
35 disp("EXAMPLE : 5.11 : SOLUTION :-") ;
36 printf("\n (a) Shunt capacitances for zero ,
    positive & negative sequences , C_0 = C_1 = C_2 =
    %.2f F /mi/phase \n",C_0) ;
37 printf("\n (b) Shunt capacitive reactance for zero ,
    positive & negative sequences , X_0 = X_1 = X_2 =
    %.2e M /mi/phase \n",X_0) ;
38 printf("\n (c) Charging current for zero , positive
    & negative sequences , I_0 = I_1 = I_2 = %.3f A/
    mi/phase \n",I_0) ;
39
40 printf("\n NOTE : 2.87e-03 M /mi/phase can also be
    written as 2.87 k /mi/phase as in textbook case
    (b) \n") ;

```

Scilab code Exa 5.12 calculate Zabc Z012

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
7 // GAS-INSULATED TRANSMISSION LINES
8 // EXAMPLE : 5.12 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 f= 60 ; // frequency in hertz
13 r_a = 0.19 ; // Conductor resistance in /mi
14 l = 10 ; // length in mi
15 D_s = 0.262 ; // GMR of one conductor in inches
16 d = 18 ; // conductors spacing in inches
17
18 // CALCULATIONS
19 // For case (a)
20 X_aa = %i * 0.1213 * log (12/D_s) ; // reactance of
    individual phase conductor at 12 inch spacing in
    /mi
21 Z_aa = l * ( r_a + X_aa ) ; // Z_aa = Z_bb = .... =
    Z_zz
22 Z_bb = Z_aa ;
23 Z_zz = Z_aa ;
24 Z_cc = Z_aa ;
25 D_eq1 = d * 2 ;
26 Z_ab = (l) * ( %i * 0.1213 * log(12/D_eq1) ) ;
```

```

27 Z_bc = Z_ab ;
28 Z_xy = Z_ab ; // Z_xy = Z_yx
29 Z_yz = Z_ab ;
30 Z_ba = Z_ab ;
31 Z_cb = Z_ab ;
32 D_eq2 = d * 3 ;
33 Z_bz = (1) * (%i * 0.1213 * log(12/D_eq2) ) ;
34 Z_ay = Z_bz ; // Z_ya = Z_ay
35 Z_cx = Z_bz ; // Z_cx = Z_xc
36 Z_yz = Z_bz ; // Z_zy = Z_yz
37 D_eq3 = d * 4 ;
38 Z_ac = (1) * (%i * 0.1213 * log(12/D_eq3) ) ;
39 Z_ca = Z_ac ; // Z_ac = Z_xz = Z_zx
40 D_eq4 = d * 1 ;
41 Z_ax = (1) * (%i * 0.1213 * log(12/D_eq4) ) ;
42 Z_bx = Z_ax ; // Z_ax = Z_xa ; Z_bx = Z_xb
43 Z_by = Z_ax ; // Z_by = Z_yb
44 Z_cy = Z_ax ; // Z_cy = Z_yc
45 Z_cz = Z_ax ;
46 D_eq5 = d * 5 ;
47 Z_az = (1) * (%i*0.1213*log(12/D_eq5)) ; // Z_za=
Z_az
48
49 Z_s = [Z_aa Z_ab Z_ac ; Z_ba Z_bb Z_bc ; Z_ca Z_cb
Z_cc] ;
50 Z_tm = [Z_ax Z_bx Z_cx ; Z_ay Z_by Z_cy ; Z_az Z_bz
Z_cz] ;
51 Z_M = [Z_ax Z_ay Z_az ; Z_bx Z_by Z_bz ; Z_cx Z_cy
Z_cz] ;
52 Z_N = [Z_aa Z_xy Z_ac ; Z_xy Z_aa Z_ab ; Z_ac Z_ab
Z_aa] ;
53 Z_new = (Z_s)-(Z_M)*(Z_N)^(-1)*(Z_tm) ;
54
55 // For case (b)
56 a = 1*exp(%i*120*pi/180) ; // By symmetrical
components theory to 3- system
57 A = [1 1 1; 1 a^2 a ;1 a a^2] ;
58 Z_012 = inv(A) * Z_new * A ; // Sequence-impedance

```

```

        matrix
59
60 // DISPLAY RESULTS
61 disp("EXAMPLE : 5.12 : SOLUTION :-") ;
62 printf("\n (a) Phase Impedance Matrix , [ Z_abc ] = \n
") ; disp(Z_new) ;
63 printf("\n (b) Sequence-Impedance Matrix , [ Z_012 ] = \n") ; disp(Z_012) ;

```

Scilab code Exa 5.15 calculate P1OH P1GIL E1OH E1GIL C1OH ElavgGIL ClavgOH ClavgGI

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
   GAS-INSULATED TRANSMISSION LINES
7
8 // EXAMPLE : 5.15 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 L = 50 ; // length of transmission line in km
13 P_l_oh = 820 ; // Power loss at peak load for
   overhead transmission line in kW/km
14 P_l_g = 254 ; // Power loss at peak load for gas
   insulated transmission line in kW/km
15 cost_kwh = 0.10 // cost of electric energy in $ per
   kWh
16 lf_ann = 0.7 ; // Annual load factor
17 plf_ann = 0.7 ; // Annual Power loss factor
18 h_yr = 365*24 ; // Time in Hours for a year

```

```

19 total_invest = 200000000 ; // Investment cost of GIL
   in $ ( for case (j) )
20
21 // CALCULATIONS
22 // For case (a)
23 Power_loss_OHline = P_l_oh * L ; // Power loss of
   overhead line at peak load in kW
24
25 // For case (b)
26 Power_loss_GILline = P_l_g * L ; // Power loss of
   gas-insulated transmission line at peak load in
   kW
27
28 // For case (c)
29 energy_loss_OH = Power_loss_OHline * h_yr ; // Total
   annual energy loss of OH line at peak load in
   kWh/yr
30
31 // For case (d)
32 energy_loss_GIL = Power_loss_GILline * h_yr ; //
   Total annual energy loss of GIL at peak load in
   kWh/yr
33
34 // For case (e)
35 energy_ann_OH = lf_ann * energy_loss_OH ; // Average
   energy loss of OH line at peak load in kWh/yr
36
37 // For case (f)
38 energy_ann_GIL = lf_ann * energy_loss_GIL ; //
   Average energy loss of GIL line at peak load in
   kWh/yr
39
40 // For case (g)
41 cost_ann_OH = cost_kwh * energy_ann_OH ; // Average
   annual cost of losses of OH line in $ per year
42
43 // For case (h)
44 cost_ann_GIL = cost_kwh * energy_ann_GIL ; //

```

```

        Average annual cost of losses of GIL line in $
        per year

45
46 // For case (i)
47 P_loss_ann = cost_ann_OH - cost_ann_GIL ; // Annual
        resultant savings of losses per yr
48
49 // For case (j)
50 break_period = total_invest/P_loss_ann ; // Payback
        period if GIL alternative period is selected
51
52 // DISPLAY RESULTS
53 disp("EXAMPLE : 5.15 : SOLUTION :-") ;
54 printf("\n (a) Power loss of Overhead line at peak
        load , (Power loss)_OH_line = %d kW \n",
        Power_loss_OHline) ;
55 printf("\n (b) Power loss of Gas-insulated
        transmission line , (Power loss)_GIL_line = %d kW
        \n",Power_loss_GILLine) ;
56 printf("\n (c) Total annual energy loss of Overhead
        transmission line at peak load = %.4e kWh/yr \n",
        energy_loss_OH) ;
57 printf("\n (d) Total annual energy loss of Gas-
        insulated transmission line at peak load = %.5e
        kWh/yr \n",energy_loss_GIL);
58 printf("\n (e) Average energy loss of Overhead
        transmission line = %.5e kWh/yr \n",energy_ann_OH
        );
59 printf("\n (f) Average energy loss of Gas-insulated
        transmission line at peak load = %.5e kWh/yr \n",
        energy_ann_GIL);
60 printf("\n (g) Average annual cost of losses of
        Overhead transmission line = $ %.5e/yr \n",
        cost_ann_OH);
61 printf("\n (h) Average annual cost of losses of Gas-
        insulated transmission line = $ %.5e/yr \n",
        cost_ann_GIL);
62 printf("\n (i) Annual resultant savings in losses

```

```

        using Gas-insulated transmission line = $ %.6e/yr
        \n",P_loss_ann);
63 printf("\n (j) Breakeven period when GIL alternative
is selected = %.1f years \n",break_period);

```

Scilab code Exa 5.16 calculate A1 A2 A of OH GIL and submarine transmission line

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
   GAS-INSULATED TRANSMISSION LINES
7
8 // EXAMPLE : 5.16 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 n = 40 ; // useful life in years
13 i = 10/100 ; // carrying charge rate
14 A_P = (i*(1+i)^n)/((1 + i)^n - 1) ; // Refer page
   642
15 A_F = 0.00226 ; // A_F = A/F
16 pr_tax = 3/100 ; // Annual ad property taxes is 3%
   of 1st costs of each alternative
17
18 // FOR OVERHEAD TRANSMISSION
19 L_OH = 50 ; // length of route A in mi
20 cost_b_A = 1 * 10^6 ; // cost per mile to bulid in $
21 salvage_A = 2000 ; // salvage value per mile at end
   of 40 years
22 cost_mait_OH = 500 ; // cost in $ per mile to

```

```

        maintain
23
24 // SUBMARINE TRANSMISSION LINE
25 L_S = 30 ; // length of route B in mi
26 cost_b_B = 4*10^6 ; // cost per mile to build in $
27 salvage_B = 6000 ; // salvage value per mile at end
    of 40 years
28 cost_mait_S = 1500 ; // cost in $ per mile to
    maintain
29
30 // GIL TRANSMISSION
31 L_GIL = 20 ; // length of route C in mi
32 cost_b_C = 7.6*10^6 ; // cost per mile to build in $
33 salvage_C = 1000 ; // salvage value per mile at end
    of 40 years
34 cost_mait_GIL = 200 ; // cost in $ per mile to
    maintain
35 savings = 17.5*10^6 ; // relative savings in power
    loss per year in $
36
37
38 // CALCULATIONS
39 n = 25 ; // useful life in years
40 i = 20/100 ; // carrying charge rate
41 p = ((1 + i)^n - 1)/(i*(1+i)^n) ; // p = P/A
42 // FOR OVERHEAD TRANSMISSION
43 P_OH = cost_b_A * L_OH ; // first cost of 500 kV OH
    line in $
44 F_OH = salvage_A * L_OH ; // Estimated salvage value
    in $
45 A_1 = P_OH * A_P - F_OH * A_F ; // Annual equivalent
    cost of capital in $
46 A_2 = P_OH * pr_tax + cost_mait_OH * L_OH ; //
    annual equivalent cost of tax and maintenance in
    $
47 A = A_1 + A_2 ; // total annual equi cost of OH line
    in $
48

```

```

49 // SUBMARINE TRANSMISSION LINE
50 P_S = cost_b_B * L_S ; // first cost of 500 kV OH
   line in $
51 F_S = salvage_B * L_S ; // Estimated salvage value
   in $
52 B_1 = P_S * A_P - F_S * A_F ; // Annual equivalent
   cost of capital in $
53 B_2 = P_S * pr_tax + cost_mait_S * L_S ; // annual
   equivalent cost of tax and maintainance in $
54 B = B_1 + B_2 ; // total annual equi cost of OH line
   in $
55
56 // GIL TRANSMISSION
57 P_GIL = cost_b_C * L_GIL ; // first cost of 500 kV
   OH line in $
58 F_GIL = salvage_C * L_GIL ; // Estimated salvage
   value in $
59 C_1 = P_GIL * A_P - F_GIL * A_F ; // Annual
   equivalent cost of capital in $
60 C_2 = P_GIL * pr_tax + cost_mait_GIL * L_GIL ; // 
   annual equivalent cost of tax and maintainance in
   $
61 C = C_1 + C_2 ; // total annual equi cost of OH line
   in $
62 A_net = C - savings ; // Total net annual equi cost
   of GIL
63
64 // DISPLAY RESULTS
65 disp("EXAMPLE : 5.16 : SOLUTION :-") ;
66 printf("\n OVERHEAD TRANSMISSION LINE : \n") ;
67 printf("\n Annual equivalent cost of capital
   invested in line , A_1 = $ %d \n",A_1) ;
68 printf("\n Annual equivalent cost of Tax and
   maintainance , A_2 = $ %d \n",A_2) ;
69 printf("\n Total annual equivalent cost of OH
   transmission , A = $ %d \n",A) ;
70 printf("\n \n SUBMARINE TRANSMISSION LINE : \n") ;
71 printf("\n Annual equivalent cost of capital

```

```
    invested in line , A_1 = $ %d \n" ,B_1) ;  
72 printf("\n Annual equivalent cost of Tax and  
        maintainance , A_2 = $ %d \n" ,B_2) ;  
73 printf("\n Total annual equivalent cost of  
        Submarine power transmission , A = $ %d \n" ,B) ;  
74 printf("\n \n GIL TRANSMISSION LINE : \n") ;  
75 printf("\n Annual equivalent cost of capital  
        invested in line , A_1 = $ %d \n" ,C_1) ;  
76 printf("\n Annual equivalent cost of Tax and  
        maintainance , A_2 = $ %d \n" ,C_2) ;  
77 printf("\n Total annual equivalent cost of  
        Submarine power transmission , A = $ %d \n" ,C) ;  
78 printf("\n Total net equivalent cost of GIL  
        transmission = $ %d \n" ,A_net) ;  
79 printf("\n \n The result shows use of GIL is the  
        best choice \n") ;  
80 printf("\n The next best alternative is Overhead  
        transmission line \n") ;
```

Chapter 6

DIRECT CURRENT POWER TRANSMISSION

Scilab code Exa 6.1 determine Vd Id ratio of dc to ac insulation level

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING  
2 // ANALYSIS AND DESIGN  
3 // TURAN GONEN  
4 // CRC PRESS  
5 // SECOND EDITION  
6 // CHAPTER : 6 : DIRECT CURRENT POWER TRANSMISSION  
7  
8 // EXAMPLE : 6.1 :  
9 clear ; clc ; close ; // Clear the work space and  
// console  
10  
11 // GIVEN DATA  
12 K_1 = 2.5 ; // Factor  
13 K_2 = 1.7 ; // Factor  
14  
15 // CALCULATIONS  
16 // For case (b)  
17 I_d = poly(0, 'I_d') ; // since P_loss(dc) = P_loss (
```

```

    ac)
18 I_L = poly(0, 'I_L') ; // i.e.  $2*I_d^2*R_{dc} = 3*I_L^2*R_{ac}$ 
19 I_d = sqrt(3/2)*I_L ; // Ignoring skin effects  $R_{dc} = R_{ac}$ 
20 I_d1 = 1.225*I_L ; // Refer Equ 6.23
21
22 // For case (a)
23 V_d = poly(0, 'V_d') ; // Defining a polynomial  $V_d$ 
24 E_p = poly(0, 'E_p') ; // since  $P_{dc} = P_{ac}$  (or)  $V_d * I_d = 3 * E_p * I_L$ 
25 V_d = 2.45*E_p ; // Refer Equ 6.25
26
27 // For case (c)
28 ins_lvl = (K_2*(V_d/2))/(K_1*E_p) ; // Ratio of dc
      insulation level to ac insulation level
29 ins_lvl_1 = (K_2*2.45/2)/K_1 ; // simplifying above
      equ
30 dc_i = poly(0, 'dc_i') ; // dc_i = dc insulation
      level
31 ac_i = poly(0, 'ac_i') ; // ac_i = ac insulation
      level
32 dc_i = ins_lvl_1 * ac_i ;
33
34 // DISPLAY RESULTS
35 disp("EXAMPLE : 6.1 : SOLUTION :-") ;
36 printf("\n (a) Line-to-line dc voltage of  $V_d$  in
      terms of line-to-neutral voltage  $E_p$  ,  $V_d = \n$ ")
      ; disp(V_d) ;
37 printf("\n (b) The dc line current  $I_d$  in terms of
      ac line current  $I_L$  ,  $I_d = \n$ "); disp(I_d1) ;
38 printf("\n (c) Ratio of dc insulation level to ac
      insulation level = \n") ; disp(dc_i/ac_i) ;
39 printf("\n (or) dc insulation level = \n") ; disp(
      dc_i) ;

```

Scilab code Exa 6.2 determine Vd ratio of Pdc to Pac and Ploss dc to Ploss ac

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING  
2 // ANALYSIS AND DESIGN  
3 // TURAN GONEN  
4 // CRC PRESS  
5 // SECOND EDITION  
6 // CHAPTER : 6 : DIRECT CURRENT POWER TRANSMISSION  
7  
8 // EXAMPLE : 6.2 :  
9 clear ; clc ; close ; // Clear the work space and  
// console  
10  
11 // GIVEN DATA  
12 K = 3 ; // factor  
13  
14 // CALCULATIONS  
15 // For case (a)  
16 V_d = poly(0, 'V_d') ; // defining a polynomial  
17 E_p = poly(0, 'E_p') ;  
18 V_d = K*2*E_p ; // From equ 6.18  
19  
20 // For case (b)  
21 P_dc = poly(0, 'P_dc') ;  
22 P_ac = poly(0, 'P_ac') ;  
23 P_dc = 2*P_ac ;  
24  
25 // For case (c)  
26 P_ld = poly(0, 'P_ld') ; // P_loss(dc)  
27 P_la = poly(0, 'P_la') ; // P_loss(ac)  
28 P_ld = (2/3)*P_la ;  
29  
30 // DISPLAY RESULTS
```

```

31 disp("EXAMPLE : 6.2 : SOLUTION :-") ;
32 printf("\n (a) Maximum operating V_d in terms of
           voltage E_p , V_d = \n") ; disp(V_d) ;
33 printf("\n (b) Maximum power transmission capability
           ratio , i.e , ratio of P_dc to P_ac , P_dc/P_ac = \n
           ") ; disp(P_dc/P_ac) ;
34 printf("\n (or) P_dc = \n") ; disp(P_dc) ;
35 printf("\n (c) Ratio of total I^2*R losses , i.e ,
           Ratio of P_loss(dc) to P_loss(ac) , which accompany
           maximum power flow = \n") ; disp(P_ld/P_la) ;
36 printf("\n (or) P_loss(dc) = \n") ; disp(P_ld) ;

```

Scilab code Exa 6.3 calculate KVA rating Wye side KV rating

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION
7
8 // EXAMPLE : 6.3 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 V_d0 = 125 ; // voltage rating of bridge rectifier
   in kV
13 V_dro = V_d0 ; // Max continuos no-load direct
   voltage in kV
14 I = 1600 ; // current rating of bridge rectifier in
   A
15 I_d = I ; // Max continuous current in A
16

```

```

17 // CALCULATIONS
18 // For case (a)
19 S_B = 1.047 * V_d0 * I_d ; // 3-phase kVA rating of
    rectifier transformer
20
21 // For case (b)
22 // SINCE V_d0 = 2.34*E_LN
23 E_LN = V_d0/2.34 ; // Wye side kV rating
24
25 // DISPLAY RESULTS
26 disp("EXAMPLE : 6.3 : SOLUTION :-") ;
27 printf("\n (a) Three-phase kilovolt-ampere rating ,
    S_B = %d kVA \n",S_B) ;
28 printf("\n (b) Wye-side kilovolt rating , E_L-N = %
    .4f kV \n",E_LN) ;

```

Scilab code Exa 6.4 determine X_c for all 3 possible values of ac system reactance

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION
7
8 // EXAMPLE : 6.4 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 E_LN = 53.418803 ; // Wye-side kV rating . From exa
    6.3
13 I = 1600 ; // current rating of bridge rectifier in
    A

```

```

14 I_d = I ; // Max continuous current in A
15 X_tr = 0.10 ; // impedance of rectifier transformer
16   in pu
16
17 // For case (a)
18 sc_MVA1 = 4000 ; // short-ckt MVA
19
20 // For case (b)
21 sc_MVA2 = 2500 ; // short-ckt MVA
22
23 // For case (c)
24 sc_MVA3 = 1000 ; // short-ckt MVA
25
26 // CALCULATIONS
27 nom_kV = sqrt(3) * E_LN ; // Nominal kV_L-L
28 I_1ph = sqrt(2/3) * I_d ; // rms value of wye-side
29   phase current
30 E_LN1 = E_LN * 10^3 ; // Wye-side rating in kV
31 X_B = (E_LN1/I_1ph) ; // Associated reactance base
32   in
33
34 // For case (a)
35 X_sys1 = nom_kV^2/sc_MVA1 ; // system reactance in
36
37 // Reactance of rectifier
38 transformer
39 X_C = X_sys1 + X_tr ; // Commutating reactance in
40
41 // For case (b)
42 X_sys2 = nom_kV^2/sc_MVA2 ; // system reactance in
43
44 X_C2 = X_sys2 + X_tr ; // Commutating reactance in
45
46 // For case (b) When breaker 1 & 2 are open
47 X_sys3 = nom_kV^2/sc_MVA3 ; // system reactance in

```

```

43 X_C3 = X_sys3 + X_tra ; // Commutating reactance in
44
45 // DISPLAY RESULTS
46 disp("EXAMPLE : 6.4 : SOLUTION :-") ;
47 printf("\n (a) Commutating reactance When all three
        breakers are closed , X_C = %.4f \n",X_C) ;
48 printf("\n (b) Commutating reactance When breaker 1
        is open , X_C = %.4f \n",X_C2) ;
49 printf("\n (c) Commutating reactance When breakers 1
        and 2 are open , X_C = %.4f \n",X_C3) ;

```

Scilab code Exa 6.5 calculate u Vdr pf Qr

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION
7
8 // EXAMPLE : 6.5 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 X_C = 6.2292017 ; // commutating reactance when all
    3 breakers are closed
13 E_LN = 53.418803 * 10^3 ; // Wye-side volt rating
14 V_d0 = 125 * 10^3 ; // voltage rating of bridge
    rectifier in V
15 V_dro = V_d0 ; // Max continuos no-load direct
    voltage in V
16 I = 1600 ; // current rating of bridge rectifier in

```

A

```
17 I_d = I ; // Max continuous current
18 nom_kV = sqrt(3) * E_LN ; // Nominal kV_L-L
19 X_tr = 0.10 ; // impedance of rectifier transformer
    in pu
20 alpha = 0 ; // delay angle      = 0 degree
21
22 // CALCULATIONS
23 // For case (a)
24 E_m = sqrt(2) * E_LN ;
25 u = acosd(1 - (2*X_C*I_d)/(sqrt(3)*E_m)) ; // overlap
    angle when delay angle      = 0 degree
26
27 // For case (b)
28 R_C = (3/%pi) * X_C ; // Equ commutation resistance
    per phase
29 V_d = V_d0 * cosd(alpha) - R_C * I_d ; // dc voltage
    of rectifier in V
30
31 // For case (c)
32 cos_theta = V_d/V_d0 ; // Displacement or power
    factor of rectifier
33
34 // For case (d)
35 Q_r = V_d * I_d * tand(acosd(cos_theta)) ; // magnetizing var I/P
36
37 // DISPLAY RESULTS
38 disp("EXAMPLE : 6.5 : SOLUTION :-") ;
39 printf("\n (a) Overlap angle u of rectifier , u = %.2
    f degree\n",u) ;
40 printf("\n (b) The dc voltage V_dr of rectifier ,
    V_dr = %.2 f V \n",V_d) ;
41 printf("\n (c) Displacement factor of rectifier ,
    cos      = %.3 f \n",cos_theta) ;
42 printf("\n      and      = %.1 f degree \n ",acosd(
    cos_theta)) ;
43 printf("\n (d) Magnetizing var input to rectifier ,
```

```

        Q_r = %.4e var \n" ,Q_r) ;
44
45 printf("\n NOTE : In case(d) 7.6546e+07 var is same
           as 7.6546*10^7 var = 76.546 Mvar \n") ;

```

Scilab code Exa 6.6 determine alpha u pf Qr

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION
7
8 // EXAMPLE : 6.6 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 I_d = 1600 ; // Max continuous dc current in A
13 V_d0 = 125 * 10^3 ; // voltage rating of bridge
   rectifier in V
14 V_d = 100 * 10^3 ; // dc voltage of rectifier in V
15 X_C = 6.2292017 ; // commutating reactance when all
   3 breakers are closed
16
17 // CALCULATIONS
18 // For case (a)
19 R_C = (3/%pi) * X_C ;
20 cos_alpha = (V_d + R_C*I_d)/V_d0 ; // Firing angle
21 alpha = acosd(cos_alpha) ;
22
23 // For case (b)

```

```

24 // V_d = (1/2)*V_d0*(cos_alpha + cos_delta)
25 cos_delta = (2 * V_d/V_d0) - cos_alpha ;
26 delta = acosd(cos_delta) ;
27 u = delta - alpha ; // Overlap angle u in degree
28
29 // For case (c)
30 cos_theta = V_d/V_d0 ; // power factor
31 theta = acosd(cos_theta) ;
32
33 // For case (d)
34 Q_r = V_d * I_d * tand(theta) ; // magnetizing var I
    /P
35
36 // DISPLAY RESULTS
37 disp("EXAMPLE : 6.6 : SOLUTION :-") ;
38 printf("\n (a) Firing angle of rectifier , = %
    .2f degree\n",alpha) ;
39 printf("\n (b) Overlap angle u of rectifier , u = %.2
    f degree\n",u) ;
40 printf("\n (c) Power factor , cos = %.2f \n",
    cos_theta) ;
41 printf("\n and = %.2f degree \n ",theta) ;
42 printf("\n (d) Magnetizing var input , Q_r = %.2e
    var \n",Q_r) ;

```

Scilab code Exa 6.7 determine u mode Id or Vdr

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION
7

```

```

8 // EXAMPLE : 6.7 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 X_C = 12.649731 ; // commutating reactance when 2
    breakers are open
13 alpha = 0 ;
14 I_d = 1600 ; // DC current in A
15 E_LN = 53.4188 * 10^3 ; // Wye-side rating in V
16 V_d0 = 125 * 10^3 ; // voltage rating of bridge
    rectifier in V
17
18 // CALCULATIONS
19 // For case (a)
20 E_m = sqrt(2) * E_LN ;
21 u = acosd(1 - (2 * X_C * I_d)/(sqrt(3) * E_m)) ; // 
    overlap angle u =
22
23 // For case (b)
24 // since rectifier operates in first mode i.e doesn't
    operate in second mode
25 R_C = (3/%pi) * X_C ;
26 V_dr = ( V_d0 * cosd(alpha) ) - (R_C*I_d) ; // dc
    voltage of rectifier in V
27
28 // DISPLAY RESULTS
29 disp("EXAMPLE : 6.7 : SOLUTION :-") ;
30 printf("\n (a) u = %.1f degree \n",u) ;
31 printf("\n since u < 60 degree . The rectifier
    operates at FIRST mode , the normal operating
    mode \n") ;
32 printf("\n (b) When dc current is 1600 A , V_dr = %
    .2f V \n",V_dr) ;

```

Scilab code Exa 6.10 determine Vd0 E u pf Qr No of bucks

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6
7 // CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION
8
9 clear ; clc ; close ; // Clear the work space and
  console
10
11 // GIVEN DATA
12 X_C = 6.2292 ; // commutating reactance when all 3
  breakers are closed
13 I_db = 1600 ; // dc current base in A
14 V_db = 125 * 10^3 ; // dc voltage base in V
15 I_d = I_db ; // Max continuous current in A
16 V_d = 100 * 10^3 ; // dc voltage in V
17 alpha = 0 ; // Firing angle      = 0 degree
18
19 // CALCULATIONS
20 // For case (a)
21 R_c = (3/%pi) * X_C ;
22 R_cb = V_db/I_db ; // Resistance base in
23 V_d_pu = V_d/V_db ; // per unit voltage
24 I_d_pu = I_d/I_db ; // per unit current
25 R_c_pu = R_c/R_cb ; // per unit
26 E_pu = (V_d_pu + R_c_pu * I_d_pu)/cosd(alpha) ; //
  Open ckt dc voltage in pu
27 V_d0 = E_pu * V_db ; // Open ckt dc voltage in V
28
29 // For case (b)
30 E = V_d0/2.34; // Open ckt ac voltage on wye side of
  transformer in V
```

31

```

32 // For case (c)
33 E_1LN = 92.95 * 10^3 ; // voltage in V
34 E_1B = E_1LN ;
35 E_LN = 53.44 * 10^3 ; // voltage in V
36 a = E_1LN/E_LN ;
37 n = a ; // when LTC on neutral
38 X_c_pu = 2 * R_c_pu ;
39 E_1_pu = E_1LN / E_1B ; // per unit voltage
40 cos_delta = cosd(alpha) - ( (X_c_pu * I_d_pu)/( (a/n
    ) *E_1_pu) ) ;
41 delta = acosd(cos_delta) ;
42 u = delta - alpha ;
43
44 // For case (d)
45 cos_theta = V_d/V_d0 ; // pf of rectifier
46 theta = acosd(cos_theta) ;
47
48 // For case (e)
49 Q_r = V_d*I_d*tand(theta) ; // magnetizing var I/P
50
51 // For case (f)
52 d_V = E_LN - E ; // necessary change in voltage in V
53 p_E_LN = 0.00625 * E_LN ; // one buck step can
    change in V/step
54 no_buck = d_V / p_E_LN ; // No. of steps of buck
55
56 // DISPLAY RESULTS
57 disp("EXAMPLE : 6.10 : SOLUTION :-") ;
58 printf("\n (a) Open circuit dc Voltage , V_d0 = %.2f
    V \n",V_d0);
59 printf("\n (b) Open circuit ac voltage on wye side
        of transformer , E = %.2f V \n",E);
60 printf("\n (c) Overlap angle , u = %.2f degree \n",u
    )
61 printf("\n (d) Power factor , cos = %.3f \n",
    cos_theta);
62 printf("\n and = %.2f degree \n ",theta);
63 printf("\n (e) Magnetizing var input to rectifier ,

```

```
Q_r = %.4e var \n", Q_r);  
64 printf("\n (f) Number of 0.625 percent steps of buck  
required , No. of buck = %.f steps \n", no_buck);
```

Chapter 7

TRANSIENT OVERVOLTAGES AND INSULATION COORDINATION

Scilab code Exa 7.1 determine surge Power surge current

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND
7 // INSULATION COORDINATION
8 // EXAMPLE : 7.1 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11 // GIVEN DATA
12 V = 1000 ; // surge voltage in kV
```

```

13 Z_c = 500 ; // surge impedance in
14
15 // CALCULATIONS
16 // For case (a)
17 P = V^2/Z_c ; // Total surge power in MW
18
19 // For case (b)
20 V1 = V*10^3 ; // surge voltage in V
21 i = V1/Z_c ; // surge current in A
22
23 // DISPLAY RESULTS
24 disp("EXAMPLE : 7.1 : SOLUTION :-") ;
25 printf("\n (a) Total surge power in line , P = %d MW
           \n",P) ;
26 printf("\n (b) Surge current in line , i = %d A \n",
           i) ;

```

Scilab code Exa 7.2 determine surge Power surge current

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND
   INSULATION COORDINATION
7
8 // EXAMPLE : 7.2 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 V = 1000 ; // surge voltage in kV
13 Z_c = 50 ; // surge impedance in

```

```

14
15 // CALCULATIONS
16 // For case (a)
17 P = V^2/Z_c ; // Total surge power in MW
18
19 // For case (b)
20 V1 = V*10^3 ; // surge voltage in V
21 i = V1/Z_c ; // surge current in A
22
23 // DISPLAY RESULTS
24 disp("EXAMPLE : 7.1 : SOLUTION :-") ;
25 printf("\n (a) Total surge power in line , P = %d MW
          \n",P) ;
26 printf("\n (b) Surge current in line , i = %d A \n",
          i) ;

```

Scilab code Exa 7.4 determine Crv Cri vb v Crfv ib i Crfi

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6
6 // CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND
7 // INSULATION COORDINATION
8
8 // EXAMPLE : 7.4 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11
11 // GIVEN DATA
12 R = 500 ; // Resistance in
13 Z_c = 400 ; // characteristic impedance in
14 v_f = 5000 ; // Forward travelling voltage wave in V

```

```

15 i_f = 12.5 ; // Forward travelling current wave in A
16
17 // CALCULATIONS
18 // For case (a)
19 r_v = (R - Z_c)/(R + Z_c) ; // Reflection
    coefficient of voltage wave
20
21 // For case (b)
22 r_i = -(R - Z_c)/(R + Z_c) ; // Reflection
    coefficient of current wave
23
24 // For case (c)
25 v_b = r_v * v_f ; // Backward-travelling voltage
    wave in V
26
27 // For case (d)
28 v = v_f + v_b ; // Voltage at end of line in V
29 v1 = (2 * R/(R + Z_c)) * v_f ; // (or) Voltage at
    end of line in V
30
31 // For case (e)
32 t1 = (2 * R/(R + Z_c)) ; // Refraction coefficient
    of voltage wave
33
34 // For case (f)
35 i_b = -(v_b/Z_c) ; // backward-travelling current
    wave in A
36 i_b1 = -r_v * i_f ; // (or) backward-travelling
    current wave in A
37
38
39 // For case (g)
40 i = v/R ; // Current flowing through resistor in A
41
42 // For case (h)
43 t2 = (2 * Z_c/(R + Z_c)) ; // Refraction coefficient
    of current wave
44

```

```

45 // DISPLAY RESULTS
46 disp("EXAMPLE : 7.4 : SOLUTION :-") ;
47 printf("\n (a) Reflection coefficient of voltage
        wave ,      = %.4f \n",r_v) ;
48 printf("\n (b) Reflection coefficient of current
        wave ,      = %.4f \n",r_i) ;
49 printf("\n (c) Backward-travelling voltage wave ,
        v_b = %.3f V \n",v_b) ;
50 printf("\n (d) Voltage at end of line , v = %.3f V \
        \n",v) ;
51 printf("\n      From alternative method ")
52 printf("\n      Voltage at end of line , v = %.3f V \
        \n",v) ;
53 printf("\n (e) Refraction coefficient of voltage
        wave ,      = %.4f \n",t1) ;
54 printf("\n (f) Backward-travelling current wave ,
        i_b = %.4f A \n",i_b) ;
55 printf("\n (g) Current flowing through resistor , i =
        %.4f A \n",i) ;
56 printf("\n (h) Refraction coefficient of current
        wave ,      = %.4f \n",t2) ;

```

Scilab code Exa 7.5 determine if Cr Crf v i vb ib plot of voltage and current surge

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND
   INSULATION COORDINATION
7
8 // EXAMPLE : 7.5 :
9 clear ; clc ; close ; // Clear the work space and

```

```

        console
10
11 // GIVEN DATA
12 Z_c1 = 400 ; // Surge impedance of line in
13 Z_c2 = 40 ; // Surge impedance of cable in
14 v_f = 200 ; // Forward travelling surge voltage in
               kV
15
16 // CALCULATIONS
17 // For case (a)
18 v_f1 = v_f * 10^3 ; // surge voltage in V
19 i_f = v_f1/Z_c1 ; // Magnitude of forward current
                     wave in A
20
21 // For case (b)
22 r = (Z_c2 - Z_c1)/(Z_c2 + Z_c1) ; // Reflection
               coefficient
23
24 // For case (c)
25 t = 2 * Z_c2/(Z_c2 + Z_c1) ; // Refraction
               coefficient
26
27 // For case (d)
28 v = t * v_f ; // Surge voltage transmitted forward
                  into cable in kV
29
30 // For case (e)
31 v1 = v * 10^3 ; // Surge voltage transmitted forward
                  into cable in V
32 I = v1/Z_c2 ; // Surge current transmitted forward
                  into cable in A
33
34 // For case (f)
35 v_b = r * v_f ; // surge voltage reflected back
                  along overhead line in kV
36
37 // For case (g)
38 i_b = -r * i_f ; // surge current reflected back

```

```

        along overhead line in A
39
40 // For case (h)
41 // Arbitrary values are taken in graph. Only for
   reference not for scale
42 T = 0:0.1:300 ;
43
44 for i = 1:int(length(T)/3) ; // plotting Voltage
   values
45     vo(i) = 3;
46 end
47 for i = int(length(T)/3):length(T)
48     vo(i) = 1 ;
49 end
50 for i = int(length(T))
51     vo(i) = 0 ;
52 end
53
54
55 a=gca() ;
56 ylabel("CURRENT           SENDING END
          VOLTAGE           ") ;
57 b = newaxes() ; // creates new axis
58 b.y_location = "right" ; // Position of axis
59 ylabel ("RECEIVING END") ; // Labelling y-axis
60 b.axes_visible = ["off","off","off"] ;
61 e = newaxes() ;
62 e.y_location = "middle" ;
63 e.y_label.text = "JUNCTION" ;
64 subplot(2,1,1) ;
65 plot2d(T,vo,2,'012',',[0,0,310,6]) ;
66
67 for i = 1:int(length(T)/3) ; // Plotting current
   surges value
68     io(i) = 1 ;
69 end
70 for i = int(length(T)/3):length(T)
71     io(i) = 3 ;

```

```

72 end
73 for i = int(length(T))
74     io(i) = 0 ;
75 end
76
77
78 c=gca() ;
79 d = newaxes() ;
80 d.y_location = "right" ;
81 d.filled = "off" ;
82 f.y_location = "middle" ;
83 f.y_label.text = "JUNCTION" ;
84 subplot(2,1,2) ;
85 plot2d(T,io,5,'012','','',[0,0,310,6]) ;
86
87 // DISPLAY RESULTS
88 disp("EXAMPLE : 7.5 : SOLUTION :-") ;
89 printf("\n (a) Magnitude of forward current wave ,
    i_f = %d A \n",i_f) ;
90 printf("\n (b) Reflection coefficient ,      = %.4f \n
    ",r) ;
91 printf("\n (c) Refraction coefficient ,      = %.4f \n
    ",t) ;
92 printf("\n (d) Surge voltage transmitted forward
        into cable , v = %.2f kV \n",v) ;
93 printf("\n (e) Surge current transmitted forward
        into cable , i = %.f A \n",I) ;
94 printf("\n (f) Surge voltage reflected back along
        the OH line , v_b = %.2f kV \n",v_b) ;
95 printf("\n (g) Surge current reflected back along
        the OH line , i_b = %.f A \n",i_b) ;
96 printf("\n (h) Graph shows plot of voltage & current
        surges after arrival at the junction \n") ;

```

Scilab code Exa 7.6 determine Crs Crr lattice diagram volatge plot of receiving en

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND
7 // INSULATION COORDINATION
8 // EXAMPLE : 7.6 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11 // GIVEN DATA
12 v = 1000 ; // ideal dc voltage source in V
13 Z_s = 0 ; // internal impedance in
14 Z_c = 40 ; // characteristic impedance in
15 Z_r = 60 ; // Cable is terminated in 60 resistor
16
17 // CALCULATIONS
18 // For case (a)
19 r_s = (Z_s - Z_c)/(Z_s + Z_c) ; // Reflection
20 // coefficient at sending end
21 // For case (b)
22 r_r = (Z_r - Z_c)/(Z_r + Z_c) ; // Reflection
23 // coefficient at receiving end
24 // For case (c)
25 T = 0:0.001:10.6 ; // // plotting values
26 for i = 1:length(T) ;
27 if(T(i)<=1)
28     x(i) = (1.2)*T(i) - 1 ;
29 elseif(T(i)>=1 & T(i)<=2)
30     x(i) = (-1.2)*T(i) + 1.4 ;
31 elseif(T(i)>=2 & T(i)<=3)
32     x(i) = (1.2)*T(i)- 3.4 ;
33 elseif(T(i)>=3 & T(i)<=4)

```

```

34         x(i) = (-1.2)*T(i) + 3.8 ;
35     elseif(T(i)>=4 & T(i)<=5)
36         x(i) = (1.2)*T(i)- 5.8 ;
37     elseif(T(i)>=5 & T(i)<=6)
38         x(i) = (-1.2)*T(i) + 6.2 ;
39     elseif(T(i)>=6 & T(i)<=7)
40         x(i) = (1.2)*T(i)- 8.2 ;
41     elseif(T(i)>=7 & T(i)<=8)
42         x(i) = (-1.2)*T(i) + 8.6 ;
43     elseif(T(i)>=8 & T(i)<=9)
44         x(i) = (1.2)*T(i)- 10.6 ;
45     elseif(T(i)>=9 & T(i)<=10)
46         x(i) = (-1.2)*T(i) + 11 ;
47     elseif(T(i)>=10 & T(i)<=10.6)
48         x(i) = (1.2)*T(i) - 13 ;
49     end
50 end
51
52 subplot(2,1,1) ; // Plotting two graph in same
53 window
54 plot2d(T,x,5,'012','','',[0,-1,11,0.2]) ;
55 a = gca() ;
56 xlabel("TIME") ;
57 ylabel(" s = -1 DISTANCE
58 x = 0.2") ;
59 xtitle("Fig 7.6 (c) Lattice diagram") ;
60 a.thickness = 2 ; // sets thickness of plot
61 xset('thickness',2) ; // sets thickness of axes
62 xstring(1,-1,'T') ;
63 xstring(2,-1,'2T') ;
64 xstring(3,-1,'3T') ;
65 xstring(4,-1,'4T') ;
66 xstring(5,-1,'5T') ;
67 xstring(6,-1,'6T') ;
68 xstring(7,-1,'7T') ;
69 xstring(8,-1,'8T') ;
70 xstring(9,-1,'9T') ;

```

```

70 xstring(10,-1,'10T') ;
71 xstring(0.1,0.1,'0V') ;
72 xstring(2,0.1,'1200V') ;
73 xstring(4,0.1,'960V') ;
74 xstring(6,0.1,'1008V') ;
75 xstring(8,0.1,'998.4V') ;
76 xstring(1,-0.88,'1000V') ;
77 xstring(3,-0.88,'1000V') ;
78 xstring(5,-0.88,'1000V') ;
79 xstring(7,-0.88,'1000V') ;
80 xstring(9,-0.88,'1000V') ;
81
82 // For case (d)
83 q1 = v ; // Refer Fig 7.11 in textbook
84 q2 = r_r * v ;
85 q3 = r_s * r_r * v ;
86 q4 = r_s * r_r^2 * v ;
87 q5 = r_s^2 * r_r^2 * v ;
88 q6 = r_s^2 * r_r^3 * v ;
89 q7 = r_s^3 * r_r^3 * v ;
90 q8 = r_s^3 * r_r^4 * v ;
91 q9 = r_s^4 * r_r^4 * v ;
92 q10 = r_s^4 * r_r^5 * v ;
93 q11 = r_s^5 * r_r^5 * v ;
94 V_1 = v - q1 ;
95 V_2 = v - q3 ;
96 V_3 = v - q5 ;
97 V_4 = v - q7 ; // voltage at t = 6.5T & x = 0.25l in
Volts
98 V_5 = v - q9 ;
99
100 // For case (e)
101 t = 0:0.001:9 ;
102
103 for i= 1:length(t)
104 if(t(i)>=0 & t(i)<=1)
105 y(i) = V_1 ;
106 elseif(t(i)>=1 & t(i)<=3)

```

```

107      y(i) = V_2 ;
108      elseif(t(i)>=3 & t(i)<=5)
109          y(i)= V_3 ;
110      elseif(t(i)>=5 & t(i)<=7)
111          y(i)= V_4 ;
112      elseif(t(i)>=7 & t(i)<=9)
113          y(i)= V_5 ;
114      end
115 end
116 subplot(2,1,2) ;
117 a = gca() ;
118 a.thickness = 2 ; // sets thickness of plot
119 plot2d(t,y,2,'012','','',[0,0,10,1300]) ;
120 a.x_label.text = 'TIME (T)' ; // labels x-axis
121 a.y_label.text = 'RECEIVING-END VOLTAGE (V)' ; // labels y-axis
122 xtitle("Fig 7.6 (e) . Plot of Receiving end Voltage v/s Time") ;
123 xset('thickness',2); // sets thickness of axes
124 xstring(1,0,'1T') ; // naming points
125 xstring(3,0,'3T') ;
126 xstring(5,0,'5T') ;
127 xstring(7,0,'7T') ;
128 xstring(1,1200,'1200 V') ;
129 xstring(4,960,'960 V') ;
130 xstring(6,1008,'1008 V') ;
131 xstring(8,998.4,'998.4 V') ;
132
133
134 // DISPLAY RESULTS
135 disp("EXAMPLE : 7.6 : SOLUTION :-") ;
136 printf("\n (a) Reflection coefficient at sending end , r_s = %.f \n",r_s) ;
137 printf("\n (b) Reflection coefficient at sending end , r_r = %.1f \n",r_r)
138 printf("\n (c) The lattice diagram is shown in Fig 7.6 (c) \n") ;
139 printf("\n (d) From Fig 7.6 (c) , the voltage value

```

is at $t = 6.5T$ & $x = 0.25$ l is = %.d Volts \n” ,
V_4) ;
140 printf("\n (e) The plot of the receiving-end voltage
v/s time is shown in Fig 7.6 (e) \n") ;

Chapter 8

LIMITING FACTORS FOR EXTRA HIGH AND ULTRAHIGH VOLTAGE TRANSMISSION

Scilab code Exa 8.1 determine disruptive critical rms V₀ and visual critical rms V

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 8 : LIMITING FACTORS FOR EXTRA-HIGH AND
7 // ULTRAHIGH VOLTAGE TRANSMISSION
8 // EXAMPLE : 8.1 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11 // GIVEN DATA
12 m_0 = 0.90 ; // Irregularity factor
```

```

13 p = 74 ; // Atmospheric pressure in Hg
14 t = 10 ; // temperature in degree celsius
15 D = 550 ; // Equilateral spacing b/w conductors in
    cm
16 d = 3 ; // overall diameter in cm
17
18 // CALCULATIONS
19 // For case (a)
20 r = d/2 ;
21 delta = 3.9211 * p/( 273 + t ) ; // air density
    factor
22 V_0_ph = 21.1 * delta * m_0 * r * log(D/r) ; //
    disruptive critical rms line voltage in kV/phase
23 V_0 = sqrt(3) * V_0_ph ; // disruptive critical rms
    line voltage in kV
24
25 // For case (b)
26 m_v = m_0 ;
27 V_v_ph = 21.1*delta*m_v*r*(1 + (0.3/sqrt(delta*r) ))
    * log(D/r) ; // visual critical rms line voltage
    in kV/phase
28 V_v = sqrt(3)*V_v_ph ; // visual critical rms line
    voltage in kV
29
30 // DISPLAY RESULTS
31 disp("EXAMPLE : 8.1 : SOLUTION :-") ;
32 printf("\n (a) Disruptive critical rms line voltage
    , V_0 = %.1f kV \n",V_0) ;
33 printf("\n (b) Visual critical rms line voltage ,
    V_v = %.1f kV \n",V_v) ;

```

Scilab code Exa 8.2 determine total fair weather corona loss Pc

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN

```

```

2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 8 : LIMITING FACTORS FOR EXTRA-HIGH AND
   ULTRAHIGH VOLTAGE TRANSMISSION
7
8 // EXAMPLE : 8.2 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 f = 60 ; // freq in Hz
13 d = 3 ; // overall diameter in cm
14 D = 550 ; // Equilateral spacing b/w conductors in
   cm
15 V1 = 345 ; // operating line voltage in kV
16 V_0 = 172.4 ; // disruptive critical voltage in kV
17 L = 50 ; // line length in mi
18 p = 74 ; // Atmospheric pressure in Hg
19 t = 10 ; // temperature in degree celsius
20 m_0 = 0.90 ; // Irregularity factor
21
22 // CALCULATIONS
23 r = d/2 ;
24 delta = 3.9211 * p/( 273 + t ) ; // air density
   factor
25 V_0 = 21.1 * delta * m_0 * r * log(D/r) ; //
   disruptive critical rms line voltage in kV/phase
26 V = V1/sqrt(3) ; // Line to neutral operating voltage
   in kV
27 P_c = (390/delta)*(f+25)*sqrt(r/D)*(V - V_0)^2 *
   10^-5 ; // Fair weather corona loss per phase in
   kW/mi/phase
28 P_cT = P_c * L ; // For total line length corona
   loss in kW/phase
29 T_P_c = 3 * P_cT ; // Total corona loss of line in
   kW

```

```
30
31 // DISPLAY RESULTS
32 disp("EXAMPLE : 8.2 : SOLUTION :-") ;
33 printf("\n (a) Total fair weather corona loss of the
line , P_c = %.1f kW \n",T_P_c) ;
```

Chapter 9

SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

Scilab code Exa 9.1 determine symmetrical components for phase voltages

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
7 // ANALYSIS
8 // EXAMPLE : 9.1 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11 // GIVEN DATA
12 V_a = 7.3 * exp(%i*12.5*pi/180) ; // Phase voltage
13 in V
14 V_b = 0.4 * exp(%i*(-100)*pi/180) ; // Phase
```

```

    voltage in V
14 V_c = 4.4 * exp(%i*154*pi/180) ; // Phase voltage
    in V
15 a = 1 * exp(%i*120*pi/180) ; // operator 'a' by
    application of symmetrical components theory to
    3— system . Refer section 9.3 for details
16
17 // CALCULATIONS
18 V_a0 = (1/3) * (V_a + V_b + V_c) ; // Analysis equ
    in V
19 V_a1 = (1/3) * (V_a + a*V_b + a^2*V_c) ;
20 V_a2 = (1/3) * (V_a + a^2*V_b + a*V_c) ;
21 V_b0 = V_a0 ;
22 V_b1 = a^2 * V_a1 ;
23 V_b2 = a * V_a2 ;
24 V_c0 = V_a0 ;
25 V_c1 = a * V_a1 ;
26 V_c2 = a^2 * V_a2 ;
27
28 // DISPLAY RESULTS
29 disp("EXAMPLE : 9.1 : SOLUTION :-") ;
30 printf("\n The symmetrical components for the phase
    voltages V_a , V_b & V_c are\n") ;
31 printf("\n V_a0 = %.2f<%1f V \n",abs(V_a0),atand(
    imag(V_a0),real(V_a0)) );
32 printf("\n V_a1 = %.2f<%1f V \n",abs(V_a1),atand(
    imag(V_a1),real(V_a1)) );
33 printf("\n V_a2 = %.2f<%1f V \n",abs(V_a2),atand(
    imag(V_a2),real(V_a2)) );
34 printf("\n V_b0 = %.2f<%1f V \n",abs(V_b0),atand(
    imag(V_b0),real(V_b0)) );
35 printf("\n V_b1 = %.2f<%1f V \n",abs(V_b1),atand(
    imag(V_b1),real(V_b1)) );
36 printf("\n V_b2 = %.2f<%1f V \n",abs(V_b2),atand(
    imag(V_b2),real(V_b2)) );
37 printf("\n V_c0 = %.2f<%1f V \n",abs(V_c0),atand(
    imag(V_c0),real(V_c0)) );
38 printf("\n V_c1 = %.2f<%1f V \n",abs(V_c1),atand(

```

```

        imag(V_c1),real(V_c1) )) ;
39 printf("\n V_c2 = %.2f<%1f V \n",abs(V_c2),atan(
        imag(V_c2),real(V_c2) )) ;
40
41 printf("\n NOTE : V_b1 = 3.97<-99.5 V & V_c2 =
        2.52<-139.7 V result obtained is same as textbook
        answer V_b1 = 3.97<260.5 V & V_c2 = 2.52<220.3 V
        \n") ;
42 printf("\n Changes is due to a^2 = 1<240 = 1<-120
        where 1 is the magnitude & <240 is the angle in
        degree \n") ;

```

Scilab code Exa 9.2 determine complex power V012 I012

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
   ANALYSIS
7
8 // EXAMPLE : 9.2 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 V_abc = [0 ; 50 ; -50] ; // Phase voltages of a 3-
   system in V
13 I_abc = [-5 ; 5*i ; -5] ; // Phase current of a 3-
   system in A
14
15 // CALCULATIONS
16 // For case (a)

```

```

17 S_3ph = (V_abc)' * conj(I_abc) ; // 3- complex
      power in VA
18
19 // For case (b)
20 a = 1*exp(%i*120*pi/180) ; // By symmetrical
      components theory to 3-
      system
21 A = [1 1 1; 1 a^2 a ;1 a a^2] ;
22 V_012 = inv(A) * (V_abc) ; // Sequence voltage
      matrices in V
23 I_012 = inv(A) * (I_abc) ; // Sequence current
      matrices in A
24
25 // For case (c)
26 S_3ph1 = 3 * ([V_012(1,1) V_012(2,1) V_012(3,1)]) *
      (conj(I_012)) ; // Three-phase complex power in
      VA . Refer equ 9.34(a)
27
28 // DISPLAY RESULTS
29 disp("EXAMPLE : 9.2 : SOLUTION :-") ;
30 printf("\n (a) Three-phase complex power using equ
      9.30 , S_3- = %.4f<%f VA \n",abs(S_3ph) ,
      atand(imag(S_3ph),real(S_3ph)) ) ;
31 printf("\n (b) Sequence Voltage matrices , [V_012] =
      V \n") ;
32 printf("\n      %.f<%f ",abs(V_012(1,1)),atand( imag
      (V_012(1,1)),real(V_012(1,1)) ) );
33 printf("\n      %.4f<%f ",abs(V_012(2,1)),atand(
      imag(V_012(2,1)),real(V_012(2,1)) ) );
34 printf("\n      %.4f<%f ",abs(V_012(3,1)),atand(
      imag(V_012(3,1)),real(V_012(3,1)) ) );
35 printf("\n \n      Sequence current matrices , [I_012]
      = A \n") ;
36 printf("\n      %.4f<%1f ",abs(I_012(1,1)),atand(
      imag(I_012(1,1)),real(I_012(1,1)) ) );
37 printf("\n      %.4f<%f ",abs(I_012(2,1)),atand(
      imag(I_012(2,1)),real(I_012(2,1)) ) );
38 printf("\n      %.4f<%f ",abs(I_012(3,1)),atand(
      imag(I_012(3,1)),real(I_012(3,1)) ) );

```

```

39 printf("\n \n (c) Three-phase complex power using
equ 9.34 , S_3- = %.4f<%.f VA \n",abs(S_3ph1) ,
atand(imag(S_3ph1),real(S_3ph1)) );

```

Scilab code Exa 9.3 determine line impedance and sequence impedance matrix

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
ANALYSIS
7
8 // EXAMPLE : 9.3 :
9 clear ; clc ; close ; // Clear the work space and
console
10
11 // GIVEN DATA
12 l = 40 ; // line length in miles
13 // Conductor parameter from Table A.3
14 r_a = 0.206 ; // Ohms per conductor per mile in /mi
15 r_b = r_a ; // r_a = r_b = r_c in /mi
16 D_s = 0.0311 ; // GMR in ft where D_s = D_sa = D_sb
= D_sc
17 D_ab = sqrt(2^2 + 8^2) ; // GMR in ft
18 D_bc = sqrt(3^2 + 13^2) ; // GMR in ft
19 D_ac = sqrt(5^2 + 11^2) ; // GMR in ft
20 D_e = 2788.5 ; // GMR in ft since earth resistivity
is zero
21 r_e = 0.09528 ; // At 60 Hz in /mi
22
23 // CALCULATIONS

```

```

24 // For case (a)
25 Z_aa =[ (r_a + r_e) + %i * 0.1213*log(D_e/D_s)]*l ;
        // Self impedance of line conductor in
26 Z_bb = Z_aa ;
27 Z_cc = Z_bb ;
28 Z_ab = [r_e + %i * 0.1213*log(D_e/D_ab)]*l ; // 
        Mutual impedance in
29 Z_ba = Z_ab ;
30 Z_bc = [r_e + %i * 0.1213*log(D_e/D_bc)]*l ;
31 Z_cb = Z_bc ;
32 Z_ac = [r_e + %i * 0.1213*log(D_e/D_ac)]*l ;
33 Z_ca = Z_ac ;
34 Z_abc = [Z_aa Z_ab Z_ac ; Z_ba Z_bb Z_bc ; Z_ca Z_cb
            Z_cc] ; // Line impedance matrix
35
36 // For case (b)
37 a = 1*exp(%i*120*pi/180) ; // By symmetrical
        components theory to 3-
        system
38 A = [1 1 1; 1 a^2 a ;1 a a^2] ;
39 Z_012 = inv(A) * Z_abc*A ; // Sequence impedance
        matrix
40
41 // DISPLAY RESULTS
42 disp("EXAMPLE : 9.3 : SOLUTION :-") ;
43 printf("\n (a) Line impedance matrix , [ Z_abc ] = \n"
        ) ; disp(Z_abc) ;
44 printf("\n (b) Sequence impedance matrix of line , [
        Z_012 ] = \n") ; disp(Z_012) ;

```

Scilab code Exa 9.4 determine line impedance and sequence impedance matrix of trans

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
        ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS

```

```

4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
    ANALYSIS
7
8 // EXAMPLE : 9.4 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 l = 40 ; // line length in miles
13 // Conductor parameter from Table A.3
14 r_a = 0.206 ; // Ohms per conductor per mile in /mi
15 r_b = r_a ; // r_a = r_b = r_c in /mi
16 D_s = 0.0311 ; // GMR in ft where D_s = D_sa = D_sb
    = D_sc
17 D_ab = sqrt(2^2 + 8^2) ; // GMR in ft
18 D_bc = sqrt(3^2 + 13^2) ; // GMR in ft
19 D_ac = sqrt(5^2 + 11^2) ; // GMR in ft
20 D_e = 2788.5 ; // GMR in ft since earth resistivity
    is zero
21 r_e = 0.09528 ; // At 60 Hz in /mi
22
23 // CALCULATIONS
24 // For case (a)
25 Z_s = [(r_a + r_e) + %i*0.1213*log(D_e/D_s)]*l ; //
    Self impedance of line conductor in . From equ
    9.49
26 D_eq = (D_ab * D_bc * D_ac)^(1/3) ; // Equ GMR
27 Z_m = [r_e + %i*0.1213*log(D_e/D_eq)]*l ; // From
    equ 9.50
28 Z_abc = [Z_s Z_m Z_m ; Z_m Z_s Z_m ; Z_m Z_m Z_s] ;
    // Line impedance matrix
29
30 // For case (b)
31 Z_012 = [(Z_s+2*Z_m) 0 0 ; 0 (Z_s-Z_m) 0 ; 0 0 (Z_s-
    Z_m)] ; // Sequence impedance matrix . From equ

```

9.54

```
32
33 // DISPLAY RESULTS
34 disp("EXAMPLE : 9.4 : SOLUTION :-") ;
35 printf("\n (a) Line impedance matrix when line is
completely transposed , [Z_abc] = \n") ; disp(
Z_abc) ;
36 printf("\n (b) Sequence impedance matrix when line
is completely transposed , [Z_012] = \n") ; disp(
Z_012) ;
```

Scilab code Exa 9.5 determine mo m2 for zero negative sequence unbalance

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
ANALYSIS
7
8 // EXAMPLE : 9.5 :
9 clear ; clc ; close ; // Clear the work space and
console
10
11 // GIVEN DATA
12 Z_012 = [(19.6736 + 109.05044*i) (0.5351182 +
0.4692097*i) (- 0.5351182 + 0.4692097*i) ; (-
0.5351182 + 0.4692097*i) (8.24 + 28.471684*i)
(- 1.0702365 - 0.9384195*i) ; (0.5351182 +
0.4692097*i) (1.0702365 - 0.9384195*i) (8.24 +
28.471684*i)] ; // Line impedance matrix .
result of exa 9.3
13 Y_012 = inv(Z_012) ; // Sequence admittance of line
```

```

14
15 // CALCULATIONS
16 // For case (a)
17 Y_01 = Y_012(1,2) ;
18 Y_11 = Y_012(2,2) ;
19 m_0 = Y_01/Y_11 ; // Per-unit unbalance for zero-
    sequence in pu from equ 9.67b
20 m_0_per = m_0 * 100 ; // Per-unit unbalance for zero
    -sequence in percentage

21
22 // For case (b)
23 Z_01 = Z_012(1,2) ;
24 Z_00 = Z_012(1,1) ;
25 m_01 = -(Z_01/Z_00) ; // Per-unit unbalance for zero
    -sequence in pu from equ 9.67b
26 m_01_per = m_01 * 100 ; // Per-unit unbalance for
    zero-sequence in percentage

27
28 // For case (c)
29 Y_21 = Y_012(3,2) ;
30 Y_11 = Y_012(2,2) ;
31 m_2 = (Y_21/Y_11) ; // Per-unit unbalance for zero-
    sequence in pu from equ 9.67b
32 m_2_per = m_2 * 100 ; // Per-unit unbalance for zero
    -sequence in percentage

33
34 // For case (d)
35 Z_21 = Z_012(3,2) ;
36 Z_22 = Z_012(3,3) ;
37 m_21 = -(Z_21/Z_22) ; // Per-unit unbalance for zero
    -sequence in pu from equ 9.67b
38 m_21_per = m_21 * 100 ; // Per-unit unbalance for
    zero-sequence in percentage

39
40 // DISPLAY RESULTS
41 disp("EXAMPLE : 9.5 : SOLUTION :-") ;
42 printf("\n (a) Per-unit electromagnetic unbalance
    for zero-sequence , m_0 = %.2f<%.1f percent pu \n"

```

```

    " ,abs(m_0_per),atand( imag(m_0_per),real(m_0_per)
)) ;
43 printf("\n (b) Approximate value of Per-unit
electromagnetic unbalance for negative-sequence ,
m_0 = %.2f<% .1f percent pu \n" ,abs(m_01_per),
atand( imag(m_01_per),real(m_01_per) ) ) ;
44 printf("\n (c) Per-unit electromagnetic unbalance
for negative-sequence , m_2 = %.2f<% .1f percent
pu \n" ,abs(m_2_per),atand( imag(m_2_per),real(
m_2_per) ) ) ;
45 printf("\n (d) Approximate value of Per-unit
electromagnetic unbalance for negative-sequence ,
m_2 = %.2f<% .1f percent pu \n" ,abs(m_21_per),
atand( imag(m_21_per),real(m_21_per) ) ) ;

```

Scilab code Exa 9.6 determine Pabc Cabc C012 d0 d2

```

1
2 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6
7 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
ANALYSIS
8
9 // EXAMPLE : 9.6 :
10 clear ; clc ; close ; // Clear the work space and
console
11
12 // GIVEN DATA
13 kv = 115 ; // Line voltage in kV
14
15 // For case (a)
```

```

16 h_11 = 90 ; // GMD b/w ground wires & their images
17 r_a = 0.037667 ; // Radius in metre
18 p_aa = 11.185 * log(h_11/r_a) ; // unit is F^(-1)m
19 p_bb = p_aa ;
20 p_cc = p_aa ;
21 l_12 = sqrt(22 + (45 + 37)^2) ;
22 D_12 = sqrt(2^2 + 8^2) ; // GMR in ft
23 p_ab = 11.185*log(l_12/D_12) ; // unit is F^(-1)m
24 p_ba = p_ab ;
25 D_13 = sqrt(3^2 + 13^2) ; // GMR in ft
26 l_13 = 94.08721051 ;
27 p_ac = 11.185 * log(l_13/D_13) ; // unit is F^(-1)m
28 p_ca = p_ac ;
29 l_23 = 70.72279912 ;
30 D_23 = sqrt(5^2 + 11^2) ; // GMR in ft
31 p_bc = 11.185 * log(l_23/D_23) ; // unit is F^(-1)m
32 p_cb = p_bc ;
33 P_abc = [p_aa p_ab p_ac ; p_ba p_bb p_bc ; p_ca p_cb
            p_cc] ; // Matrix of potential coefficients
34
35 // For case (b)
36 C_abc = inv(P_abc) ; // Matrix of maxwells
                           coefficients
37
38 // For case (c)
39 a = 1*exp(%i*120*pi/180) ; // By symmetrical
                               components theory to 3-
                               system
40 A = [1 1 1; 1 a^2 a ;1 a a^2] ;
41 C_012 = inv(A) * C_abc * A ; // Matrix of sequence
                               capacitances
42
43 // For case (d)
44 C_01 = C_012(1,2) ;
45 C_11 = C_012(2,2) ;
46 C_21 = C_012(3,2) ;
47 d_0 = C_01/C_11 ; // Zero-sequence electrostatic
                     unbalances . Refer equ 9.115
48 d_2 = -C_21/C_11 ; // Negative-sequence

```

```

electrostatic unbalances . Refer equ 9.116
49
50 // DISPLAY RESULTS
51 disp("EXAMPLE : 9.6 : SOLUTION :-") ;
52 printf("\n (a) Matrix of potential coefficients , [
      P_abc] = \n") ; disp(P_abc) ;
53 printf("\n (b) Matrix of maxwells coefficients , [
      C_abc] = \n") ; disp(C_abc) ;
54 printf("\n (c) Matrix of sequence capacitances , [
      C_012] = \n") ; disp(C_012) ;
55 printf("\n (d) Zero-sequence electrostatic
      unbalances , d_0 = %.4f<%1f \n" , abs(d_0) , atand(
      imag(d_0) , real(d_0) ) ) ;
56 printf("\n      Negative-sequence electrostatic
      unbalances , d_2 = %.4f<%1f \n" , abs(d_2) , atand(
      imag(d_2) , real(d_2) ) ) ;

```

Scilab code Exa 9.9 determine Iphase Isequence Vphase Vsequence LinetoLineVoltages

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
   ANALYSIS
7
8 // EXAMPLE : 9.9 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 kv = 230 ; // Line voltage in kV
13 Z_0 = 0.56 * %i ; // impedance in

```

```

14 Z_1 = 0.2618 * %i ; // Impedance in
15 Z_2 = 0.3619 * %i ; // Impedance in
16 z_f = 5 + 0*%i ; // fault impedance in
17 v = 1 * exp(%i*0*pi/180) ;
18
19 // CALCULATIONS
20 // For case (a)
21 Z_B = kv^2/200 ; // Imedance base on 230 kV line
22 Z_f = z_f/Z_B ; // fault impedance in pu
23 I_a0 = v/(Z_0 + Z_1 + Z_2 + 3*Z_f) ; // Sequence
    currents in pu A
24 I_a1 = I_a0 ;
25 I_a2 = I_a0 ;
26 a = 1 * exp(%i*120*pi/180) ; // By symmetrical
    components theory to 3-
        system
27 A = [1 1 1; 1 a^2 a ;1 a a^2] ;
28 I_f = A * [I_a0 ; I_a1 ; I_a2] ; // Phase currents
    in pu A
29
30 // For case (b)
31 V_a = [0 ; v ; 0] - [Z_0 0 0 ; 0 Z_1 0 ; 0 0 Z_2]*[
    I_a0 ; I_a1 ; I_a2] ; // Sequence voltage in pu V
32 V_f = A*V_a ; // Phase voltage in pu V
33
34 // For case (c)
35 V_abf = V_f(1,1) - V_f(2,1) ; // Line-to-line
    voltages at fault points in pu V
36 V_bcf = V_f(2,1) - V_f(3,1) ; // Line-to-line
    voltages at fault points in pu V
37 V_caf = V_f(3,1) - V_f(1,1) ; // Line-to-line
    voltages at fault points in pu V
38
39 // DISPLAY RESULTS
40 disp("EXAMPLE : 9.9 : SOLUTION :-") ;
41 printf("\n (b) Sequence currents , I_a0 = I_a1 =
    I_a2 = %.4f<%.1f pu A \n",abs(I_a0),atand(imag(
    I_a0),real(I_a0))) ;
42 printf("\n Phase currents in pu A , [ I_af ; I_bf ;

```

```

    I_cf] = pu A \n") ;
43 printf("\n %.4f<%.1f ",abs(I_f),atand(imag(I_f),
    real(I_f)) ) ;
44 printf("\n \n (c) Sequence voltages are , [V_a0 ;
    V_a1 ; V_a2 ] = pu V \n") ;
45 printf("\n %.4f<%.1f ",abs(V_a),atand(imag(V_a),
    real(V_a)) ) ;
46 printf("\n \n Phase voltages are , [V_af ; V_bf ;
    V_cf ] = pu V \n") ;
47 printf("\n %.4f<%.1f ",abs(V_f),atand(imag(V_f),
    real(V_f)) ) ;
48 printf("\n \n (d) Line-to-line voltages at fault
    points are , V_abf = %.4f<%.1f pu V \n",abs(V_abf),
    atand(imag(V_abf),real(V_abf)) ) ;
49 printf("\n Line-to-line voltages at fault points
    are , V_abf = %.4f<%.1f pu V \n",abs(V_bcf),
    atand(imag(V_bcf),real(V_bcf)) ) ;
50 printf("\n Line-to-line voltages at fault points
    are , V_caf = %.4f<%.1f pu V \n",abs(V_caf),
    atand(imag(V_caf),real(V_caf)) ) ;
51
52 printf("\n NOTE : ERROR : Calculation mistake in
    textbook from case(c) onwards \n") ;

```

Scilab code Exa 9.10 determine Isequence Iphase Vsequence at fault G1 G2

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
   ANALYSIS
7

```

```

8 // EXAMPLE : 9.10 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 Z_0 = 0.2619 * %i ;
13 Z_1 = 0.25 * %i ;
14 Z_2 = 0.25 * %i ;
15 v = 1 * exp(%i*0*pi/180) ;
16 a = 1 * exp(%i*120*pi/180) ; // By symmetrical
    components theory to 3- system
17 A = [1 1 1; 1 a^2 a ;1 a a^2] ;
18
19 // CALCULATIONS
20 // For case (b)
21 I_a0 = v/(Z_0 + Z_1 + Z_2) ; // Sequence currents at
    fault point F in pu A
22 I_a1 = I_a0 ;
23 I_a2 = I_a0 ;
24
25 // For case (c)
26 I_a1g1 = (1/2) * I_a1 ; // Sequence current at
    terminals of generator G1 in pu A
27 I_a2g1 = (1/2) * I_a2 ;
28 I_a0g1 = 0.5/(0.55 + 0.5)*I_a0 ; // By current
    division in pu A
29
30 // For case (d)
31 I_f = [A] * [I_a0g1 ; I_a1g1 ; I_a2g1] ; // Phase
    current at terminal of generator G1 in pu A
32
33 // For case (e)
34 V_a = [0 ; v ; 0] - [Z_0 0 0 ; 0 Z_1 0 ; 0 0 Z_2]*[
    I_a0g1 ; I_a1g1 ; I_a2g1] ; // Sequence voltage
    in pu V
35
36 // For case (f)
37 V_f = [A]*[V_a] ; // Phase voltage at terminal of

```

```

        generator G1 in pu V
38
39 // For case (g)
40 I_a1g2 = (1/2) * I_a1 ; // By symmetry for Generator
   G2
41 I_a2g2 = (1/2) * I_a2 ;
42 I_a0g2 = 0 ; // By inspection
43 // V_a1(HV) leads V_a1(LV) by 30 degree & V_a2(HV)
   lags V_a2(LV) by 30 degree
44 I_a0G2 = I_a0g2 ;
45 I_a1G2 = abs(I_a1g2)*exp(%i * (atand( imag(I_a1g2),
   real(I_a1g2) ) - 30) * %pi/180) ; // (-90-30) =
   (-120)
46 I_a2G2 = abs(I_a2g2)*exp(%i *(atand( imag(I_a2g2),
   real(I_a2g2) ) + 30) * %pi/180) ; // (-90+30) =
   (-60)
47
48 I_f2 = [A] * [I_a0G2 ; I_a1G2 ; I_a2G2] ; // Phase
   current at terminal of generator G2 in pu A
49
50 // Sequence voltage at terminal of generator G2 in
   pu V
51 V_a0G2 = 0 ;
52 V_a1G2 = abs(V_a(2,1))*exp(%i * (atand( imag(V_a
   (2,1)),real(V_a(2,1)) ) - 30) * %pi/180) ; //
   (0-30) = (-30)
53 V_a2G2 = abs(V_a(3,1))*exp(%i * (atand( imag(V_a
   (3,1)),real(V_a(3,1)) ) + 30) * %pi/180) ; //
   (180+30)=(210)=(-150)
54
55 V_f2 = A * [V_a0G2 ; V_a1G2 ; V_a2G2] ; // Phase
   voltage at terminal of generator G2 in pu V
56
57 // DISPLAY RESULTS
58 disp("EXAMPLE : 9.10 : SOLUTION :-") ;
59 printf("\n (b) The sequence current at fault point F
   , I_a0 = I_a1 = I_a2 = %.4f<%f pu A \n",abs(
   I_a0),atand(imag(I_a0),real(I_a0))) ;

```

```

60 printf("\n (c) Sequence currents at the terminals of
           generator G1 , \n") ;
61 printf("\n      I_a0 ,G_1 = %.4f<%.f pu A ",abs(I_a0g1
           ),atand( imag(I_a0g1),real(I_a0g1) )) ;
62 printf("\n      I_a1 ,G_1 = %.4f<%.f pu A ",abs(I_a1g1
           ),atand( imag(I_a1g1),real(I_a1g1) )) ;
63 printf("\n      I_a2 ,G_1 = %.4f<%.f pu A ",abs(I_a2g1
           ),atand( imag(I_a2g1),real(I_a2g1) )) ;
64 printf("\n \n (d) Phase currents at terminal of
           generator G1 are , [ I_af ; I_bf ; I_cf ] = pu A \n
           ") ;
65 printf("\n      %.4f<%.f ",abs(I_f),atand(imag(I_f)
           ,real(I_f) )) ;
66 printf("\n \n (e) Sequence voltages at the terminals
           of generator G1 , [ V_a0 ; V_a1 ; V_a2 ] = pu V \
           n") ;
67 printf("\n      %.4f<%.1f ",abs(V_a),atand(imag(V_a
           ),real(V_a) )) ;
68 printf("\n \n (f) Phase voltages at terminal of
           generator G1 are , [ V_af ; V_bf ; V_cf ] = pu V \n
           ") ;
69 printf("\n      %.4f<%.1f ",abs(V_f),atand(imag(V_f
           ),real(V_f) )) ;
70 printf("\n \n (g) Sequence currents at the terminals
           of generator G2 , \n") ;
71 printf("\n      I_a0 ,G_2 = %.f<% .f pu A ",abs(I_a0G2)
           ,atand( imag(I_a0G2),real(I_a0G2) )) ;
72 printf("\n      I_a1 ,G_2 = %.4f<%.f pu A ",abs(I_a1G2)
           ,atand( imag(I_a1G2),real(I_a1G2) )) ;
73 printf("\n      I_a2 ,G_2 = %.4f<%.f pu A ",abs(I_a2G2)
           ,atand( imag(I_a2G2),real(I_a2G2) )) ;
74 printf("\n \n      Phase currents at terminal of
           generator G2 are , [ I_af ; I_bf ; I_cf ] = pu A \n
           ") ;
75 printf("\n      %.4f<%.f ",abs(I_f2),atand(imag(
           I_f2),real(I_f2) )) ;
76 printf("\n \n      Sequence voltages at the terminals
           of generator G2 , [ V_a0 ; V_a1 ; V_a2 ] = pu V\n

```

```

    ") ;
77 printf("\n      %.f<%.f ",abs(V_a0G2),atand( imag(
    V_a0G2),real(V_a0G2) )) ;
78 printf("\n      %.4f<%.f ",abs(V_a1G2),atand( imag(
    V_a1G2),real(V_a1G2) )) ;
79 printf("\n      %.4f<%.f ",abs(V_a2G2),atand( imag(
    V_a2G2),real(V_a2G2) )) ;
80 printf("\n \n      Phase voltages at terminal of
generator G2 are , [V_af ; V_bf ; V_cf] = pu V \n
") ;
81 printf("\n      %.4f<%.1f ",abs(V_f2),atand(imag(
    V_f2),real(V_f2) )) ;
82
83 printf("\n \n NOTE : ERROR : Calclation mistake in
textbook case(f) ") ;
84 printf("\n In case (g) V_a2 = 0.1641<-150 is same as
textbook answer V_a2 = 0.1641<210 , i.e
(360-150)=210 \n") ;

```

Scilab code Exa 9.11 determine Iphase Isequence Vphase Vsequence LinetoLineVoltage

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
   ANALYSIS
7
8 // EXAMPLE : 9.11 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA

```

```

12 kv = 230 ; // Line voltage in kV from Exa 9.9
13 Z_0 = 0.56*i ; // Zero-sequence impedance in pu
14 Z_1 = 0.2618*i ; // Zero-sequence impedance in pu
15 Z_2 = 0.3619*i ; // Zero-sequence impedance in pu
16 z_f = 5 ; // Fault impedance in
17 v = 1*exp(%i*0*pi/180) ; //
18 a = 1*exp(%i*120*pi/180) ; // By symmetrical
      components theory to 3-
      system
19 A = [1 1 1; 1 a^2 a ;1 a a^2] ;
20
21 // CALCULATIONS
22 // For case (b)
23 I_a0 = 0 ; // Sequence current in A
24 Z_B = kv^2/200 ; // Base impedance of 230 kV line
25 Z_f = z_f/Z_B ; // fault impedance in pu
26 I_a1 = v/(Z_1 + Z_2 + Z_f) ; // Sequence current in
      pu A
27 I_a2 = - I_a1 ; // Sequence current in pu A
28 I_f = [A] * [I_a0 ; I_a1 ; I_a2] ; // Phase current
      in pu A
29
30 // For case (c)
31 V_a = [0 ; v ; 0]-[Z_0 0 0 ; 0 Z_1 0 ; 0 0 Z_2]*[
      I_a0 ; I_a1 ; I_a2] ; // Sequence voltages in pu
      V
32 V_f = A*V_a ; // Phase voltages in pu V
33
34 // For case (d)
35 V_abf = V_f(1,1) - V_f(2,1) ; // Line-to-line
      voltages at fault points in pu V
36 V_bcf = V_f(2,1) - V_f(3,1) ; // Line-to-line
      voltages at fault points in pu V
37 V_caf = V_f(3,1) - V_f(1,1) ; // Line-to-line
      voltages at fault points in pu V
38
39
40
41 // DISPLAY RESULTS

```

```

42 disp("EXAMPLE : 9.11 :SOLUTION :-") ;
43 printf("\n (b) Sequence currents are , \n") ;
44 printf("\n I_a0 = %.f pu A ",I_a0) ;
45 printf("\n I_a1 = %.4f<%.2f pu A ",abs(I_a1),atand(
        imag(I_a1),real(I_a1))) ;
46 printf("\n I_a2 = %.4f<%.2f pu A ",abs(I_a2),atand(
        imag(I_a2),real(I_a2))) ;
47 printf("\n \n Phase currents are , [ I_af ; I_bf ;
        I_cf ] = pu A \n") ;
48 printf("\n %.4f<%1f ",abs(I_f),atand(imag(I_f),
        real(I_f))) ;
49 printf("\n \n (c) Sequence voltages are , [ V_a0 ;
        V_a1 ; V_a2 ] = pu V \n") ;
50 printf("\n %.4f<%1f ",abs(V_a),atand(imag(V_a),
        real(V_a))) ;
51 printf("\n \n Phase voltages are , [ V_af ; V_bf ;
        V_cf ] = pu V \n") ;
52 printf("\n %.4f<%1f ",abs(V_f),atand(imag(V_f),
        real(V_f))) ;
53 printf("\n \n (d) Line-to-line voltages at the fault
        points are \n") ;
54 printf("\n V_abf = %.4f<%1f pu V \n",abs(V_abf),
        ,atand( imag(V_abf),real(V_abf))) ;
55 printf("\n V_bcf = %.4f<%1f pu V \n",abs(V_bcf),
        ,atand( imag(V_bcf),real(V_bcf))) ;
56 printf("\n V_caf = %.4f<%1f pu V \n",abs(V_caf),
        ,atand( imag(V_caf),real(V_caf))) ;
57
58 printf("\n \n NOTE : ERROR : Minor calclation
        mistake in textbook ") ;

```

Scilab code Exa 9.12 determine Iphase Isequence Vphase Vsequence LinetoLineVoltage

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
ANALYSIS AND DESIGN

```

2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
    ANALYSIS
7
8 // EXAMPLE : 9.12 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 z_f = 5 ; // Fault-impedance in
13 z_g = 10 ; // Ground-impedance in
14 kv = 230 ; // Line voltage in kV from Exa 9.9
15 Z_0 = 0.56*%i ; // Zero impedance in pu
16 Z_1 = 0.2618*%i ; // Positive sequence Impedance in
    pu
17 Z_2 = 0.3619*%i ; // Negative sequence Impedance in
    pu
18 v = 1*exp(%i*0*180/%pi) ;
19 a = 1*exp(%i*120*%pi/180) ; // By symmetrical
    components theory to 3-
        system
20 A = [1 1 1; 1 a^2 a ;1 a a^2] ;
21
22 // CALCULATIONS
23 // For case (b)
24 Z_B = kv^2/200 ; // Base impedance of 230 kV line
25 Z_f = z_f/Z_B ; // fault impedance in pu
26 Z_g = z_g/Z_B ;
27 I_a1 = v/( (Z_1 + Z_f) + ( (Z_2 + Z_f)*(Z_0 + Z_f +
    3*Z_g)/((Z_2 + Z_f)+(Z_0 + Z_f + 3*Z_g)) ) ) ; //
    Sequence current in pu A
28 I_a2 = -[(Z_0 + Z_f + 3*Z_g)/( (Z_2 + Z_f )+(Z_0 +
    Z_f + 3*Z_g) )]*I_a1 ; // Sequence current in pu
    A
29 I_a0 = -[(Z_2 + Z_f)/( (Z_2 + Z_f)+(Z_0 + Z_f + 3*
    Z_g) )]*I_a1 ; // Sequence current in pu A

```

```

30 I_f = A*[I_a0 ; I_a1 ; I_a2] ; // Phase currents in
   pu A
31
32 // For case (c)
33 V = [0 ; v ; 0] - [Z_0 0 0 ; 0 Z_1 0 ; 0 0 Z_2]*[
   I_a0 ; I_a1 ; I_a2] ; // Sequence Voltages in pu
   V
34 V_f = A*[V] ; // Phase voltages in pu V
35
36 // For case (d)
37 V_abf = V_f(1,1) - V_f(2,1) ; // Line-to-line
   voltages at fault points a & b
38 V_bcf = V_f(2,1) - V_f(3,1) ; // Line-to-line
   voltages at fault points b & c
39 V_caf = V_f(3,1) - V_f(1,1) ; // Line-to-line
   voltages at fault points c & a
40
41 // DISPLAY RESULTS
42 disp("EXAMPLE : 9.12 : SOLUTION :-") ;
43 printf("\n (b) Sequence currents are , \n") ;
44 printf("\n   I_a0 = %.4f<%.2f pu A ",abs(I_a0),atand
   ( imag(I_a0),real(I_a0) )) ;
45 printf("\n   I_a1 = %.4f<%.2f pu A ",abs(I_a1),atand
   ( imag(I_a1),real(I_a1) )) ;
46 printf("\n   I_a2 = %.4f<%.2f pu A ",abs(I_a2),atand
   ( imag(I_a2),real(I_a2) )) ;
47 printf("\n \n   Phase currents are , [ I_af ; I_bf ;
   I_cf ] = pu A \n") ;
48 printf("\n           %.4f<%.1f ",abs(I_f),atand(imag(I_f)
   ),real(I_f) )) ;
49 printf("\n \n (c) Sequence voltages , [ V_a0 ; V_a1 ;
   V_a2 ] = pu V \n") ;
50 printf("\n           %.4f<%.1f ",abs(V),atand(imag(V),
   real(V) )) ;
51 printf("\n \n   Phase voltages , [ V_af ; V_bf ; V_cf ]
   = pu V \n") ;
52 printf("\n           %.4f<%.1f ",abs(V_f),atand(imag(V_f)
   ),real(V_f) )) ;

```

```

53 printf("\n \n (d) Line-to-line voltages at the fault
      points are , \n") ;
54 printf("\n     V_abf = %.4f<%.1f pu V \n" ,abs(V_abf) ,
      atan( imag(V_abf),real(V_abf) ) ) ;
55 printf("\n     V_bcf = %.4f<%.1f pu V \n" ,abs(V_bcf) ,
      atan( imag(V_bcf),real(V_bcf) ) ) ;
56 printf("\n     V_caf = %.4f<%.1f pu V \n" ,abs(V_caf) ,
      atan( imag(V_caf),real(V_caf) ) ) ;

```

Scilab code Exa 9.13 determine Iphase Isequence Vphase Vsequence LinetoLineVoltage

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
   ANALYSIS
7
8 // EXAMPLE : 9.13 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 z_f = 5 ; // Fault-impedance in
13 Z_0 = 0.56*i ; // Zero impedance in pu
14 Z_1 = 0.2618*i ; // Positive sequence Impedance in
   pu
15 Z_2 = 0.3619*i ; // Negative sequence Impedance in
   pu
16 kv = 230 ; // Line voltage in kV from Exa 9.9
17 a = 1 * exp(%i*120*pi/180) ; // By symmetrical
   components theory to 3-
18 A = [1 1 1; 1 a^2 a ;1 a a^2] ;

```

```

19
20 // CALCULATIONS
21 // For case (b)
22 Z_B = kv^2/200 ; // Base impedance of 230 kV line
23 Z_f = z_f/Z_B ; // fault impedance in pu
24 v = 1*exp(%i*0*%pi/180) ;
25 I_a0 = 0 ; // Sequence current in pu A
26 I_a1 = v/(Z_1 + Z_f) ; // Sequence current in pu A
27 I_a2 = 0 ; // Sequence current in pu A
28 I_f = A*[I_a0 ; I_a1 ; I_a2] ; // Phase-current in
pu A
29
30 // For case (c)
31 V = [0 ; v ; 0] - [Z_0 0 0 ; 0 Z_1 0 ; 0 0 Z_2]*[
I_a0 ; I_a1 ; I_a2] ; // Sequence Voltages in pu
V
32 V_f = A*[V] ; // Phase voltages in pu V
33
34 // For case (d)
35 V_abf = V_f(1,1) - V_f(2,1) ; // Line-to-line
voltages at fault points a & b
36 V_bcf = V_f(2,1) - V_f(3,1) ; // Line-to-line
voltages at fault points b & c
37 V_caf = V_f(3,1) - V_f(1,1) ; // Line-to-line
voltages at fault points c & a
38
39 // DISPLAY RESULTS
40 disp("EXAMPLE : 9.13 : SOLUTION :-") ;
41 printf("\n (b) Sequence currents are , \n") ;
42 printf("\n I_a0 = %.1f pu A ",I_a0) ;
43 printf("\n I_a1 = %.4f<%.1f pu A ",abs(I_a1),
atand(imag(I_a1),real(I_a1))) ;
44 printf("\n I_a2 = %.1f pu A ",I_a2) ;
45 printf("\n \n Phase currents are , [ I_af ; I_bf ;
I_cf ] = pu A \n ") ;
46 printf("\n %.4f<%1f ",abs(I_f),atand(imag(I_f),
real(I_f))) ;
47 printf("\n \n (c) Sequence voltages , [ V_a0 ; V_a1 ;

```

```

        V_a2] = pu V \n ") ;
48 printf("\n %.4f<%.1f ",abs(V),atand(imag(V),
      real(V))) ;
49 printf("\n \n Phase voltages , [V_af ; V_bf ;
      V_cf] = pu V \n ") ;
50 printf("\n %.4f<%.1f ",abs(V_f),atand(imag(V_f),
      real(V_f))) ;
51 printf("\n \n (d) Line-to-line voltages at the fault
      points are , \n") ;
52 printf("\n V_abf = %.4f<%.1f pu V \n",abs(V_abf),
      atand( imag(V_abf),real(V_abf))) ;
53 printf("\n V_bcf = %.4f<%.1f pu V \n",abs(V_bcf),
      atand( imag(V_bcf),real(V_bcf))) ;
54 printf("\n V_caf = %.4f<%.1f pu V \n",abs(V_caf),
      atand( imag(V_caf),real(V_caf))) ;
55
56 printf("\n \n NOTE : ERROR : Calclation mistake in
      textbook case(d) " ) ;

```

Scilab code Exa 9.14 determine admittance matrix

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
   ANALYSIS
7
8 // EXAMPLE : 9.14 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA

```

```

12 VG_1 = 1*exp(%i*0*%pi/180) ;
13 VG_2 = 1*exp(%i*0*%pi/180) ;
14
15 // CALCULATIONS
16 // For case (a)
17 I_1 = 1*exp(%i*0*%pi/180) ;
18 I_2 = 1*exp(%i*0*%pi/180) ;
19 V_1 = 0.4522*exp(%i*90*%pi/180) ;
20 V_2 = 0.4782*exp(%i*90*%pi/180) ;
21 Y_11 = I_1/V_1 ; // When V_2 = 0
22 Y_21 = (-0.1087)*Y_11 ; // When V_2 = 0
23 Y_22 = I_2/V_2 ; // When V_1 = 0
24 Y_12 = Y_21 ;
25 Y = [Y_11 Y_12 ; Y_21 Y_22] ; // Admittance matrix
        associated with positive-sequence n/w
26
27 // For case (b)
28 I_S1_12 = 2.0193*exp(%i*90*%pi/180) ; // Short-ckt F
        & F' to neutral & by superposition theorem
29 I_S1_10 = 0.2884*exp(%i*90*%pi/180) ; // Short-ckt F
        & F' to neutral & by superposition theorem
30 I_S2_12 = 0.4326*exp(%i*90*%pi/180) ;
31 I_S2_10 = 1.4904*exp(%i*90*%pi/180) ;
32 I_S1 = I_S1_12 + I_S1_10 ;
33 I_S2 = I_S2_12 + I_S2_10 ;
34
35 // DISPLAY RESULTS
36 disp("EXAMPLE : 9.14 :SOLUTION :-") ;
37 printf("\n (a) Admittance matrix associated with
        positive-sequence network , Y = \n") ; disp(Y) ;
38 printf("\n (b) Source currents Two-port Thevenin
        equivalent positive sequence network are , \n") ;
39 printf("\n     I_S1 = %.4f<%.f pu " ,abs(I_S1),atand(
        imag(I_S1),real(I_S1)) ) ;
40 printf("\n     I_S2 = %.4f<%.f pu \n" ,abs(I_S2),
        atand( imag(I_S2),real(I_S2)) ) ;

```

Scilab code Exa 9.15 determine uncoupled positive and negative sequence

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6
7 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
8 // ANALYSIS
9 // EXAMPLE : 9.15 :
10 clear ; clc ; close ; // Clear the work space and
   console
11
12 // GIVEN DATA
13 Y_11 = -2.2115*i ;
14 Y_12 = 0.2404*i ;
15 Y_21 = 0.2404*i ;
16 Y_22 = -2.0912*i ;
17 Y = [Y_11 Y_12 ; Y_21 Y_22] ;
18 I_S1 = 2.3077*i ;
19 I_S2 = 1.9230*i ;
20
21 I_a1 = poly(0, 'I_a1') ;
22 I_a2 = poly(0, 'I_a2') ;
23 a = Y_12*I_S2 - Y_22*I_S1 ;
24 b = (Y_12+Y_22)*I_a1 ;
25 c = Y_12*I_S1 - Y_11*I_S2 ;
26 d = (Y_12 + Y_11)*I_a1 ;
27 V1 = (1/det(Y))*[(a-b) ; (c+d)] ; // Gives the
   uncoupled positive sequence N/W
28 A = (Y_12+Y_22)*I_a2 ;
```

```

29 B = (Y_12 + Y_11)*I_a2 ;
30 V2 = (1/det(Y))*[A ; B] ; // Gives the uncoupled
   negative sequence N/W
31
32 // DISPLAY RESULTS
33 disp("EXAMPLE : 9.15 : SOLUTION :-") ;
34 printf("\n (a) [V_a1 ; V_a11] = ") ; disp(V1) ;
35 printf("\n      Values of Uncoupled positive-sequence
   network \n") ;
36 printf("\n (b) [V_a2 ; V_a22] = ") ; disp(V2) ;
37 printf("\n      Values of Uncoupled negative-sequence
   network \n") ;

```

Scilab code Exa 9.16 determine Xc0 C0 Ipc Xpc Lpc Spc Vpc

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
   ANALYSIS
7
8 // EXAMPLE : 9.16 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 H_aa = 81.5 ;
13 D_aa = 1.658 ;
14 f = 60 ; // Freq in Hz
15 I = 20 ;
16 kV = 69 ; // Line voltage in kV
17 MVA = 25 ; // Transformer T1 rating in MVA

```

```

18
19 // CALCULATIONS
20 // For case (a)
21 C_0 = 29.842*10^-9/(log(H_aa/D_aa)) ; // Capacitance
   in F/mi
22 b_0 = 2*%pi*f*C_0 ; // Susceptance in S/mi
23 B_0 = b_0*I ; // For total system
24 X_C0 = (1/B_0) ; // Total zero-sequence reactance in
25 TC_0 = B_0/(2*%pi*f) ; // Total zero-sequence
   capacitance in F
26
27 // For case (c)
28 X_1 = 0.05 ; // Leakage reactance of transformer T1
   in pu
29 X_0 = X_1 ;
30 X_2 = X_1 ;
31 Z_B = kV^2/MVA ;
32 X_01 = X_0*Z_B ; // Leakage reactance in
33 V_F = 69*10^3/sqrt(3) ;
34 I_a0PC = V_F/(17310.8915*%i) ; // Zero-sequence
   current flowing through PC in A
35 I_PC = 3*abs(I_a0PC) ; // Continuous-current rating
   of the PC in A
36
37 // For case (d)
38 X_PC = (17310.8915 - X_01)/3 ; // Required reactance
   value for PC in
39
40 // For case (e)
41 L_PC = X_PC/(2*%pi*f) ; // Inductance in H
42
43 // For case (f)
44 S_PC = (I_PC^2)*X_PC ; // Rating in VA
45 S_PC1 = S_PC*10^-3 ; // Continuous kVA rating in kVA
46
47 // For case (g)
48 V_PC = I_PC * X_PC ; // continuous-voltage rating

```

```

        for PC in V
49
50 // DISPLAY RESULTS
51 disp("EXAMPLE : 9.16 :SOLUTION :-") ;
52 printf("\n (a) Total zero-sequence susceptance per
      phase of system at 60 Hz , X_C0 = %.4f \n",
      X_C0) ;
53 printf("\n      Total zero-sequence capacitance per
      phase of system at 60 Hz , C_0 = %.4e F \n",
      TC_0) ;
54 printf("\n (c) Continuous-current rating of the PC ,
      I_PC = 3I_a0PC = %.4f A \n",abs(I_PC)) ;
55 printf("\n (d) Required reactance value for the PC ,
      X_PC = %.4f \n",X_PC) ;
56 printf("\n (e) Inductance value of the PC , L_PC = %
      .4f H \n",L_PC) ;
57 printf("\n (f) Continuous kVA rating for the PC ,
      S_PC = %.2f kVA \n",S_PC1) ;
58 printf("\n (g) Continuous-voltage rating for PC ,
      V_PC = %.2f V \n",V_PC) ;

```

Chapter 10

PROTECTIVE EQUIPMENT AND TRANSMISSION SYSTEM PROTECTION

Scilab code Exa 10.1 calculate subtransient fault current in pu and ampere

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
  ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 10 : PROTECTIVE EQUIPMENT AND
  TRANSMISSION SYSTEM PROTECTION
7
8 // EXAMPLE : 10.1 :
9 clear ; clc ; close ; // Clear the work space and
  console
10
11 // GIVEN DATA
12 X_d = 0.14*i ; // Reactance of generator in pu
13 E_g = 1*exp(%i*0*pi/180) ;
14 S_B = 25*10^3 ; // voltage in kVA
```

```

15 V_BL_V = 13.8 ; // low voltage in kV
16
17 // CALCULATIONS
18 I_f = E_g/X_d ; // Subtransient fault current in pu
19 I_BL_V = S_B/( sqrt(3)*V_BL_V) ; // Current base for
    low-voltage side
20 I_f1 = abs(I_f)*I_BL_V ; // magnitude of fault
    current in A
21
22 // DISPLAY RESULTS
23 disp("EXAMPLE : 10.1 : SOLUTION :-") ;
24 printf("\n Subtransient fault current for 3- fault
    in per units = pu \n") ; disp(I_f) ;
25 printf("\n Subtransient fault current for 3- fault
    in ampere = %.f A \n",I_f1) ;

```

Scilab code Exa 10.2 determine max Idc I_{max} I_{momentary} S_{interrupting} Sm_{omentary}

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 10 : PROTECTIVE EQUIPMENT AND
    TRANSMISSION SYSTEM PROTECTION
7
8 // EXAMPLE : 10.2 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 // For case (a)
13 I_f = 7.1428571 ; // Subtransient fault current in
    pu . Result of exa 10.1

```

```

14
15 // For case (d)
16 V_pf = 13800 ; // voltage in V
17 zeta = 1.4 ;
18 I_f1 = 7471 ; // magnitude of fault current in A
19
20 // CALCULATIONS
21 // For case (a)
22 I_fdc_max = sqrt(2)*I_f ; // Max dc current in pu
23
24 // For case (b)
25 I_f_max = 2*I_fdc_max ; // Total max instantaneous
   current in pu
26
27 // For case (c)
28 I_momt = 1.6*I_f ; // Total rms momentary current
29
30 // For case (d)
31 S_int = sqrt(3)*(V_pf)*I_f1*zeta*10^-6 ; // 
   Interrupting rating in MVA
32
33 // For case (e)
34 S_momt = sqrt(3)*(V_pf)*I_f1*1.6*10^-6 ; // 
   Momentary duty of CB in MVA
35
36 // DISPLAY RESULTS
37 disp("EXAMPLE : 10.2 : SOLUTION :-") ;
38 printf("\n (a) Maximum possible dc current component
      , I_fdc_max = %.1f pu \n",I_fdc_max) ;
39 printf("\n (b) Total maximum instantaneous current ,
      I_max = %.1f pu \n",I_f_max) ;
40 printf("\n (c) Momentary current , I_momentary = %.2
      f pu \n",I_momt) ;
41 printf("\n (d) Interrupting rating of a 2-cycle CB ,
      S_interrupting = %.f MVA \n",S_int) ;
42 printf("\n (e) Momentary duty of a 2-cycle CB ,
      S_momentary = %.2f MVA \n",S_momt) ;

```

Scilab code Exa 10.4 determine Rarc Z LineImpedanceAngle with Rarc and without

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 10 : PROTECTIVE EQUIPMENT AND
7 // TRANSMISSION SYSTEM PROTECTION
8 // EXAMPLE : 10.4 :
9 clear ; clc ; close ; // Clear the work space and
10 // console
11 // GIVEN DATA
12 z_l = 0.2 + %i * 0.7 ; // Line impedance in pu
13 f_l = 0.7 ; // Fault point at a distance from A in
14 pu
15 f_m = 1.2 ; // magnitude of fault current in pu
16 l = 10.3 ; // Line spacing in ft
17 p = 100 ; // Power in MVA
18 v = 138 ; // voltage in kV
19 i = 418.4 ; // current in A
20 z = 190.4 ; // Impedance in
21 // CALCULATIONS
22 // For case (a)
23 I = f_m * i ; // Current in arc in A
24 R_arc = 8750 * l/(I^1.4) ; // Arc resistance in
25 ohm
26 R_arc1 = R_arc/z ; // Arc resistance in pu
27 // For case (b)
28 Z_L = z_l * f_l ;
```

```

29 Z_r = Z_L + R_arc1 ; // Impedance seen by the relay
   in pu
30
31 // For case (c)
32 phi_1 = atand( imag(Z_L),real(Z_L) ) ; // Line
      impedance angle without arc resistance in degree
33 phi_2 = atand( imag(Z_r),real(Z_r) ) ; // Line
      impedance angle with arc resistance in degree
34
35 // DISPLAY RESULTS
36 disp("EXAMPLE : 10.4 : SOLUTION :-") ;
37 printf("\n (a) Value of arc resistance at fault
      point in , R_arc = %.2f \n",R_arc) ;
38 printf("\n      Value of arc resistance at fault
      point in pu , R_arc = %.2f pu \n",R_arc1) ;
39 printf("\n (b) Value of line impedance including the
      arc resistance , Z_L + R_arc = pu \n") ; disp(
      Z_r) ;
40 printf("\n (c) Line impedance angle without arc
      resistance , = %.2f degree \n",phi_1) ;
41 printf("\n      Line impedance angle with arc
      resistance , = %.2f degree \n",phi_2) ;

```

Scilab code Exa 10.5 determine protection zones and plot of operating time vs impe

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 10 : PROTECTIVE EQUIPMENT AND
   TRANSMISSION SYSTEM PROTECTION
7
8 // EXAMPLE : 10.5 :

```

```

9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // CALCULATIONS
12 // For case (a)
13 // Coordinate Values taken here are only for
    reference . Refer exa 10.5
14
15 T = 0:0.01:300 ;
16
17 for i = 1:int(length(T)/1.1) ;
18     po(i) = 4 ;
19 end
20 for i = int(length(T)/1.1):length(T)
21     po(i) = 5 ;
22 end
23 for i = 1:int(length(T)/1.1)
24     io(i) = 4 ;
25 end
26 for i = int(length(T)/1.1):length(T)
27     io(i) = 3 ;
28 end
29
30 a= gca() ;
31 subplot(2,1,1) ; // To plot 2 graph in same graphic
    window
32 a.thickness = 2 ; // sets thickness of plot of
    points
33 plot2d(T,po,3,'012',',[0 0 310 7]) ;
34 plot2d(T,io,3,'012',',[0 0 310 7]) ;
35 xtitle("Fig 10.5 (a)    Zones of protection for relay
    R_12") ;
36 xset('thickness',2); // sets thickness of axes
37 xstring(25,3.8,['']) ;
38 xstring(45,4.2,['(1)']) ;
39 plot(45,4,'+') ;
40 xstring(60,3.8,['']) ;
41 xstring(60,4.2,['B_12']) ;

```

```

42 xstring(120,3.8,'[]') ;
43 xstring(120,4.2,'B_21') ;
44 xstring(140,4.2,'(2)') ;
45 plot(140,4,'+') ;
46 xstring(155,3.8,'[]') ;
47 xstring(155,4.2,'B_23') ;
48 xstring(220,3.8,'[]') ;
49 xstring(220,4.2,'B_32') ;
50 xstring(270,5.0,'(3)') ;
51 xstring(285,2.8,'[]') ;
52 xstring(285,3.2,'B_35') ;
53 xstring(285,4.8,'[]') ;
54 xstring(285,5.2,'B_34') ;
55 xstring(85,3.4,'TL_12') ;
56 xstring(180,3.4,'TL_23') ;
57 xstring(60,3,'ZONE 1') ;
58 xstring(100,2,'ZONE 2') ;
59 xstring(190,1,'ZONE 3') ;
60
61 // For case (b)
62
63 for i = 1:int(length(T)/4) ;
64     vo(i) = 0.5;
65 end
66 for i = int(length(T)/4):length(T/1.7)
67     vo(i) = 2;
68 end
69 for i = int(length(T)/1.7):length(T)
70     vo(i) = 4
71 end
72
73 for i = int(length(T)/2.14):length(T/1.35) ; // 
    plotting Voltage values
74     uo(i) = 0.5;
75 end
76 for i = int(length(T)/1.35):length(T)
77     uo(i) = 2;
78 end

```

```

79
80 a = gca() ;
81 a.thickness = 2 ;
82 subplot(2,1,2)
83 plot2d(T,vo,2,'012',',[0 0 310 7]) ;
84 plot2d(T,uo,2,'012',',[0 0 310 7]) ;
85 ylabel("OPERATING TIME") ;
86 xlabel("IMPEDANCE") ;
87 xtitle("Fig 10.5 (b) Coordination of distance relays
          , Operating time v/s Impedance") ;
88 xset('thickness',2); // sets thickness of axes
89 xstring(0.1,0.3,'T_1') ;
90 xstring(30,0.6,'R_12') ;
91 xstring(58,1.3,'T_2') ;
92 xstring(100,2.0,'R_12') ;
93 xstring(160,3.0,'T_3') ;
94 xstring(230,4.0,'R_12') ;
95 xstring(160,0.6,'R_23') ;
96 xstring(260,2.1,'R_23') ;
97
98 // DISPLAY RESULTS
99 disp("EXAMPLE : 10.5 : SOLUTION :-") ;
100 printf("\n (a) The zone of protection for relay R_12
           is shown in Fig 10.5 (a) \n") ;
101 printf("\n ZONE 1 lies b/w (1) & B_21 \n") ;
102 printf("\n ZONE 2 lies b/w (1) & TL_23 \n") ;
103 printf("\n ZONE 3 lies after (1) \n") ;
104 printf("\n (b) The coordination of the distance
           relays R_12 & R_21 in terms of Operating time v/s
           Impedance is shown in Fig 10.5 (b)" ) ;

```

Scilab code Exa 10.6 determine I_{max} CT VT ZLoad Zr

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN

```

```

2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 10 : PROTECTIVE EQUIPMENT AND
    TRANSMISSION SYSTEM PROTECTION
7
8 // EXAMPLE : 10.6 :
9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 kv = 230 * 10^3 ; // transmission system voltage in
    V
13 VA = 100 * 10^6 ; // Maximum peak load supplied by
    TL_12 in VA
14 ZTL_12 = 2 + %i * 20 ; // Positive-sequence
    impedances of line TL_12
15 ZTL_23 = 2.5 + %i * 25 ; // Positive-sequence
    impedances of line TL_23
16 pf = 0.9 ; // Lagging pf
17
18 // CALCULATIONS
19 // For case (a)
20 I_max = VA/(sqrt(3)*kv) ; // Maximum load current in
    A
21
22 // For case (b)
23 CT = 250/5 ; // CT ratio which gives about 5A in
    secondary winding under the maximum loading
24
25 // For case (c)
26 vr = 69 ; // selecting Secondary voltage of 69 V
    line to neutral
27 VT = (kv/sqrt(3))/vr ; // Voltage ratio
28
29 // For case (d)
30 Z_r = CT/VT ; // impedance measured by relay . Z_r =

```

```

(V/VT) / (I/CT)
31 Z_TL_12 = Z_r * ZTL_12 ; // Impedance of lines TL_12
   as seen by relay
32 Z_TL_23 = Z_r * ZTL_23 ; // Impedance of lines TL_23
   as seen by relay
33
34 // For case (e)
35 Z_load = vr * CT * (pf + %i*sind(acosd(pf)))/(I_max)
   ; // Load impedance based on secondary ohms
36
37 // For case (f)
38 Z_r1 = 0.80 * Z_TL_12 ; // Zone 1 setting of relay
   R_12
39
40 // For case (g)
41 Z_r2 = 1.20 * Z_TL_12 ; // Zone 2 setting of relay
   R_12
42
43 // For case (h)
44 Z_r3 = Z_TL_12 + 1.20*(Z_TL_23) ; // Zone 3 setting
   of relay R_12
45
46 // DISPLAY RESULTS
47 disp("EXAMPLE : 10.6 : SOLUTION :-") ;
48 printf("\n (a) Maximum load current , I_max = %.2f A
   \n",I_max) ;
49 printf("\n (b) CT ratio , CT = %.1f \n",CT) ;
50 printf("\n (c) VT ratio , VT = %.1f \n",VT) ;
51 printf("\n (d) Impedance measured by relay = %.3f
   Z_line \n",Z_r) ;
52 printf("\n (e) Load impedance based on secondary
   ohms , Z_load = (secondary) \n") ; disp(Z_load)
   ;
53 printf("\n (f) Zone 1 setting of relay R_12 , Z_r =
   (secondary) \n") ; disp(Z_r1) ;
54 printf("\n (g) Zone 2 setting of relay R_12 , Z_r =
   (secondary) \n") ; disp(Z_r2) ;
55 printf("\n (h) Zone 3 setting of relay R_12 , Z_r =

```

```
(secondary) \n") ; disp(Z_r3) ;
```

Scilab code Exa 10.7 determine setting of zone1 zone2 zone3 of mho relay R12

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING  
2 // ANALYSIS AND DESIGN  
3 // TURAN GONEN  
4 // CRC PRESS  
5 // SECOND EDITION  
6 // CHAPTER : 10 : PROTECTIVE EQUIPMENT AND  
7 // TRANSMISSION SYSTEM PROTECTION  
8 // EXAMPLE : 10.7 :  
9 clear ; clc ; close ; // Clear the work space and  
// console  
10  
11 // GIVEN DATA  
12 Z_r1 = 0.0415692 + %i*0.4156922 ; // Required zone 1  
// setting . From result of exa 10.6  
13 Z_r2 = 0.0623538 + %i*0.6235383 ; // Required zone 2  
// setting . From result of exa 10.6  
14 Z_r3 = 0.1299038 + %i*1.2990381 ; // Required zone 3  
// setting . From result of exa 10.6  
15  
16 // CALCULATIONS  
17 // For case (a)  
18 theta1 = atand(imag(Z_r1),real(Z_r1)) ;  
19 Z_1 = abs(Z_r1)/cosd(theta1 - 30) ; // Zone 1  
// setting of mho relay R_12  
20  
21 // For case (b)  
22 theta2 = atand(imag(Z_r2),real(Z_r2)) ;  
23 Z_2 = abs(Z_r2)/cosd(theta2 - 30) ; // Zone 2  
// setting of mho relay R_12
```

```
24
25 // For case (b)
26 theta3 = atan2(imag(Z_r3),real(Z_r3)) ;
27 Z_3 = abs(Z_r3)/cosd(theta3 - 30) ; // Zone 3
      setting of mho relay R_12
28
29 // DISPLAY RESULTS
30 disp("EXAMPLE : 10.7 : SOLUTION :-") ;
31 printf("\n (a) Zone 1 setting of mho relay R_12 = %
      .4f (secondary) \n",Z_1) ;
32 printf("\n (b) Zone 2 setting of mho relay R_12 = %
      .4f (secondary) \n",Z_2) ;
33 printf("\n (c) Zone 3 setting of mho relay R_12 = %
      .4f (secondary) \n",Z_3) ;
```

Chapter 12

CONSTRUCTION OF OVERHEAD LINES

Scilab code Exa 12.1 calculate cost of relocating affordability

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 12 : CONSTRUCTION OF OVERHEAD LINES
7
8 // EXAMPLE : 12.1 :
9 clear ; clc ; close ; // Clear the work space and
console
10
11 // GIVEN DATA
12 cost_avg = 1500 ; // Average cost on each repair in
$ 
13 r_0 = 0 ; // No. of times repair required for damage
to line
14 r_1 = 1 ; // No. of times repair required
15 r_2 = 2 ; // No. of times repair required
```

```

16 r_3 = 3 ; // No. of times repair required
17 P_r_0 = 0.4 ; // Probability of exactly no. of
     repairs for r_0
18 P_r_1 = 0.3 ; // Probability of exactly no. of
     repairs for r_1
19 P_r_2 = 0.2 ; // Probability of exactly no. of
     repairs for r_2
20 P_r_3 = 0.1 ; // Probability of exactly no. of
     repairs for r_3
21 R_0 = 0 ; // No. of times repair required for
     relocating & rebuilding
22 R_1 = 1 ; // No. of times repair required
23 P_R_0 = 0.9 ; // Probability of exactly no. of
     repairs for R_0
24 P_R_1 = 0.1 ; // Probability of exactly no. of
     repairs for R_1
25 n = 25 ; // useful life in years
26 i = 20/100 ; // carrying charge rate
27 p = ((1 + i)^n - 1)/(i*(1+i)^n) ; // p = P/A . Refer
     page 642
28
29 // CALCULATIONS
30 B = cost_avg*(r_0*P_r_0 + r_1*P_r_1 + r_2*P_r_2 +
     r_3*P_r_3 - R_0*P_R_0 - R_1*P_R_1)*p ; //
     Affordable cost of relocating line
31
32 // DISPLAY RESULTS
33 disp("EXAMPLE : 12.1 : SOLUTION :-") ;
34 printf("\n Affordable cost of relocating line , B =
     $ %.1f \n" ,B) ;
35 printf("\n Since actual relocating & rebuilding of
     line would cost much more than amount found \n")
     ;
36 printf("\n The distribution engineer decides to
     keep the status quo \n") ;

```

Scilab code Exa 12.2 calculate pressure of wind on pole and conductors

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING  
2 // ANALYSIS AND DESIGN  
3 // TURAN GONEN  
4 // CRC PRESS  
5 // SECOND EDITION  
6 // CHAPTER : 12 : CONSTRUCTION OF OVERHEAD LINES  
7  
8 // EXAMPLE : 12.2 :  
9 clear ; clc ; close ; // Clear the work space and  
// console  
10  
11 // GIVEN DATA  
12 V = 40 ; // Actual wind velocity in mi/hr  
13 c_pg = 40 ; // Circumference at ground level in  
// inches  
14 c_pt = 28 ; // Circumference at pole top in inches  
15 l = 35 ; // height of pole in feet  
16 l_g = 6 ; // Height of pole set in ground in feet  
17 d_c = 0.81 ; // dia. of copper conductor in inches  
18 span_avg = 120 ; // Average span in ft  
19 no_c = 8 ; // NO. of conductors  
20  
21 // CALCULATIONS  
22 // For case (a)  
23 p = 0.00256 * (V^2) ; // Buck's Formula to find wind  
// pressure on cylindrical surface in lb/ft^2  
24 d_pg = c_pg/(%pi) ; // dia. of pole at ground line  
// in inches  
25 d_pt = c_pt/(%pi) ; // dia. of pole at pole top in  
// inches  
26 h_ag = ( l - l_g ) * 12 ; // Height of pole above
```

```

        ground in inch
27 S_pni = (1/2) * (d_pg + d_pt) * h_ag ; // projected
      area of pole in square inch
28 S_pni_ft = S_pni * 0.0069444 ; // projected area of
      pole in square ft
29 P = S_pni_ft * p ; // Total pressure of wind on pole
      in lb
30
31 // For case (b)
32 S_ni = d_c * span_avg * 12 ; // Projected area of
      conductor in square inch . [1 feet = 12 inch]
33 S_ni_ft = S_ni * 0.0069444 ; // Projected area of
      conductor in square ft . [1 sq inch = (0.0833333)
      ^2 sq feet      0.069444 sq feet]
34 P_C = S_ni_ft * p * no_c ; // Total pressure of wind
      on conductor in lb
35
36 // DISPLAY RESULTS
37 disp("EXAMPLE : 12.2 : SOLUTION :-");
38 printf("\n (a) Total pressure of wind on pole , P =
      %.2f lb \n",P);
39 printf("\n (b) Total pressure of wind on conductors
      , P = %.2f lb \n",P_C);

```

Scilab code Exa 12.3 calculate min required pole circumference at ground line

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
      ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 12 : CONSTRUCTION OF OVERHEAD LINES
7
8 // EXAMPLE : 12.3 :

```

```

9 clear ; clc ; close ; // Clear the work space and
    console
10
11 // GIVEN DATA
12 a = 45 ; // OH line to be bulit on wood poles in ft
13 b = 6.5 ; // Ground depth in ft
14 c = 1 ; // Top cross-arm below pole top in ft
15 d = 3 ; // Lower cross-arm below pole top in ft
16 m_t = 0.6861 ; // Transverse wind load on top cross-
    arm in lb/ft
17 m_l = 0.4769 ; // Transverse wind load on lower
    cross-arm in lb/ft
18 u_s = 8000 ; // Ultimate strength of wood pole in lb
    /sq.in
19 sf = 2 ; // Safety factor
20 span_avg = 250 ; // Average span in ft
21 p = 9 ; // Transverse wind load on wood poles in clb
    /sq.ft
22
23 // CALCULATIONS
24 h_1j = a - b - c ; // Moment arms for top arm in ft
25 h_2j = a - b - d ; // Moment arms for top arm in ft
26 M_tc1 = 1 * 4* m_t * span_avg * h_1j ; // Total
    bending moment for top arm in lb-ft
27 M_tc2 = 1 * 4* m_l * span_avg * h_2j ; // Total
    bending moment for lower arm in lb-ft
28 M_tc = M_tc1 + M_tc2 ; // Total bending moment for
    both cross-arms together in lb-ft
29 S = u_s/sf ; // Allowable max fiber stress in pounds
    per sq.inch
30 c_pg = ( M_tc/( 2.6385*10^-4*S ) )^(1/3) ; //
    circumference of pole at ground line in inch
31
32 c_pt = 22 ; // From proper tables , for 8000 psi ,
33 h_ag = a - b ; // Height of pole above ground in ft
34 d_pg = c_pg/(%pi) ; // circumference of pole at
    ground line in inches
35 d_pt = c_pt/(%pi) ; // circumference of pole at pole

```

```

    top in inches
36 M_gp = (1/72)*p *(h_ag^2)*(d_pg + 2*d_pt) ; //  

    Bending moment due to wind on pole in pound ft .  

    using equ 12.9
37 M_T = M_tc + M_gp ; // Total bending moment due to  

    wind on conductor & pole
38 c_pg1 = (M_T/( 2.6385 * 10^-4 * S ) )^(1/3) ; //  

    using equ 12.11
39
40 // DISPLAY RESULTS
41 disp("EXAMPLE : 12.3 : SOLUTION :-") ;
42 printf("\n Minimum required pole circumference at  

    the ground line , c = %.1f in \n",c_pg1) ;
43 printf("\n Therefore , the nearest standard size  

    pole , which has a ground-line circumference larger  

    than c = %.1f in , has to be used \n",c_pg1) ;
44 printf("\n Therefore required pole circumference at  

    the ground line to be used is , c = %.f inch \n",
    c_pg1) ;

```

Scilab code Exa 12.4 calculate Th beta Tv Tg

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // CHAPTER : 12 : CONSTRUCTION OF OVERHEAD LINES
7
8 // EXAMPLE : 12.4 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA

```

```

12 T1 = 3000 ; // Bending moments in lb
13 T2 = 2500 ; // Bending moments in lb
14 h1 = 37.5 ; // Bending moments at heights in ft
15 h2 = 35.5 ; // Bending moments at heights in ft
16 h_g = 36.5 ; // Height at which Guy is attached to
                  pole in ft
17 L = 15 ; // Lead of guy in ft
18
19 // CALCULATIONS
20 // For case (a)
21 T_h = ( T1*h1 + T2*h2 )/h_g ; // Horizontal
          component of tension in guy wire in lb . From equ
          12.26
22
23 // For case (b)
24 bet = atand(h_g/L) ; // beta angle in degree . From
          equ 12.28
25
26 // For case (c)
27 T_v = T_h * tand(bet) ; // Vertical component of
          tension in guy wire in lb . From equ 12.34
28
29 // For case (d)
30 T_g = T_h/( cosd(bet) ) ; // Tension in guy wire in
          lb . From equ 12.29
31 T_g1 = sqrt( T_h^2 + T_v^2 ) ; // Tension in guy
          wire in lb
32
33 // DISPLAY RESULTS
34 disp("EXAMPLE : 12.4 : SOLUTION :-") ;
35 printf("\n (a) Horizontal component of tension in
          guy wire , T_h = %.1f lb \n",T_h) ;
36 printf("\n (b) Angle , = %.2f degree \n",bet)
          ;
37 printf("\n (c) Vertical component of tension in guy
          wire , T_v = %.2f lb \n",T_v) ;
38 printf("\n (d) Tension in guy wire , T_g = %.1f lb \
          \n",T_g) ;

```

```
39 printf("\n      (or) From another equation , \n") ;  
40 printf("\n      Tension in guy wire , T_g = %.1f lb \\\n", T_g1) ;
```

Chapter 13

SAG AND TENSION ANALYSIS

Scilab code Exa 13.1 calculate length sag Tmax Tmin Tappr

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // CHAPTER : 13 : SAG AND TENSION ANALYSIS
7
8 // EXAMPLE : 13.1 :
9 clear ; clc ; close ; // Clear the work space and
console
10
11 // GIVEN DATA
12 c = 1600 ; // Length of conductor in feet
13 L = 500 ; // span b/w conductors in ft
14 w1 = 4122 ; // Weight of conductor in lb/mi
15
16 // CALCULATIONS
17 // For case (a)
```

```

18 l = 2 * c * ( sinh(L/(2*c)) ) ; // Length of
    conductor in ft using eq 13.6
19 l_1 = L * (1 + (L^2)/(24*c^2) ) ; // Length of
    conductor in ft using eq 13.8
20
21 // For case (b)
22 d = c*( cosh( L/(2*c) ) - 1 ) ; // sag in ft
23
24 // For case (c)
25 w = w1/5280 ; // Weight of conductor in lb/ft . [1
    mile = 5280 feet]
26 T_max = w * (c + d) ; // Max conductor tension in lb
27 T_min = w * c ; // Min conductor tension in lb
28
29 // For case (d)
30 T = w * (L^2)/(8*d) ; // Appr value of tension in lb
    using parabolic method
31
32 // DISPLAY RESULTS
33 disp("EXAMPLE : 13.1 : SOLUTION :-") ;
34 printf("\n (a) Length of conductor using eq 13.6 , l
    = %.3f ft \n",l);
35 printf("\n & Length of conductor using eq 13.8 , l
    = %.4f ft \n",l_1);
36 printf("\n (b) Sag , d = %.1f ft \n",d);
37 printf("\n (c) Maximum value of conductor tension
    using catenary method , T_max = %.1f lb \n",T_max
    );
38 printf("\n     Minimum value of conductor tension
    using catenary method , T_min = %.1f lb \n",T_min
    );
39 printf("\n (d) Approximate value of tension using
    parabolic method , T = %.2f lb \n",T);

```

Scilab code Exa 13.2 calculate Wi Wt P We sag vertical sag

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6
7 // CHAPTER : 13 : SAG AND TENSION ANALYSIS
8
9 // EXAMPLE : 13.2 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 L = 500 ; // span b/w conductors in ft
13 p = 4 ; // Horizontal wind pressure in lb/sq ft
14 t_i = 0.50 ; // Radial thickness of ice in inches
15 d_c = 1.093 ; // outside diameter of ACSR conductor
   in inches
16 w1 = 5399 ; // weight of conductor in lb/mi
17 s = 28500 ; // ultimate strength in lb
18
19 // CALCULATIONS
20 // For case (a)
21 w_i = 1.25 * t_i * (d_c + t_i) ; // Weight of ice in
   pounds per feet
22
23 // For case (b)
24 w = w1/5280 ; // weight of conductor in lb/ft . [1
   mile = 5280 feet]
25 W_T = w + w_i ; // Total vertical load on conductor
   in pounds per feet
26
27 // For case (c)
28 P = ( (d_c + 2*t_i)/(12) )*p ; // Horizontal wind
   force in lb/ft
29
30 // For case (d)
31 w_e = sqrt( P^2 + (w + w_i)^2 ) ; // Effective load

```

```

on conductor in lb/ft
32
33 // For case (e)
34 T = s/2 ;
35 d = w_e * L^2/(8*T) ; // sag in feet
36
37 // For case (f)
38 d_v = d * W_T/w_e ; // vertical sag in feet
39
40 // DISPLAY RESULTS
41 disp("EXAMPLE :13.2 : SOLUTION :-") ;
42 printf("\n (a) Weight of ice in pounds per feet ,
        w_i = %.4f lb/ft \n",w_i) ;
43 printf("\n (b) Total vertical load on conductor in
        pounds per feet , W_T = %.4f lb/ft \n",W_T) ;
44 printf("\n (c) Horizontal wind force in pounds per
        feet , P = %.4f lb/ft \n",P) ;
45 printf("\n (d) Effective load acting in pounds per
        feet , w_e = %.4f lb/ft \n",w_e) ;
46 printf("\n (e) Sag in feet , d = %.2f ft \n",d) ;
47 printf("\n (f) Vertical Sag in feet = %.2f ft \n",
        d_v) ;

```

Chapter 14

APPENDIX C REVIEW OF BASICS

Scilab code Exa 1.C determine power S12 P12 Q12

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
  ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // APPENDIX C : REVIEW OF BASICS
7
8 // EXAMPLE : C.1 :
9 clear ; clc ; close ; // Clear the work space and
  console
10
11 // GIVEN DATA
12 z = 100 * exp(60*i*pi/180) ; // Impedance of
  transmission line in
13 v1 = 73034.8 * exp(30*i*pi/180) ; // Bus voltages
  in V
14 v2 = 66395.3 * exp(20*i*pi/180) ; // Bus voltages
  in V
```

```

15
16 // CALCULATIONS
17 // For case (a)
18 S_12 = v1 * ( conj(v1) - conj(v2) )/( conj(z) ) ; // 
    Complex power per phase in VA
19
20
21 // For case (b)
22 P_12 = real(S_12) ; // Active power per phase in W
23
24 // For case (c)
25 Q_12 = imag(S_12) ; // Reactive power per phase in
    vars
26
27 // DISPLAY RESULTS
28 disp("EXAMPLE : C.1 : SOLUTION :-") ;
29 printf("\n (a) Complex power per phase that is being
    transmitted from bus 1 to bus 2 , S12 = %.2f<%2
    f VA \n",abs(S_12), atan(imag(S_12),real(S_12))
    *(180/%pi)) ;
30 printf("\n (b) Active power per phase that is being
    transmitted , P12 = %.2f W \n",P_12) ;
31 printf("\n (b) Reactive power per phase that is
    being transmitted , Q12 = %.2f vars \n",Q_12) ;

```

Scilab code Exa 2.C determine reactance Zbhv Zblv Xhv Xlv

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS
4 // SECOND EDITION
5
6 // APPENDIX C : REVIEW OF BASICS
7

```

```

8 // EXAMPLE : C.2 :
9 clear ; clc ; close ; // Clear the work space and
   console
10
11 // GIVEN DATA
12 X_pu = 12/100 ; // Leakage reactance in pu
13 kV_B_HV = 345 ; // HV side ratings in Y kV
14 kV_B_LV = 34.5 ; // LV side ratings in Y kV
15 MVA_B = 20 ; // selected Base on HV side in MVA
16
17 // CALCULATIONS
18 // For case (a)
19 X_pu = 12/100 ; // Reactance of transformer in pu
20
21 // For case (b)
22 Z_B_HV = (kV_B_HV)^2/MVA_B ; // HV side base
   impedance in
23
24 // For case (c)
25 Z_B_LV = (kV_B_LV)^2/MVA_B ; // LV side base
   impedance in
26
27 // For case (d)
28 X_HV = X_pu * Z_B_HV ; // Reactance referred to HV
   side in
29
30 // For case (e)
31 X_LV = X_pu * Z_B_LV ; // Reactance referred to LV
   side in
32 n = (kV_B_HV/sqrt(3))/(kV_B_LV/sqrt(3)) ; // Turns
   ratio of winding
33 X_LV1 = X_HV/n^2 ; // From equ C.89
34
35 // DISPLAY RESULTS
36 disp("EXAMPLE : C.2 : SOLUTION :-") ;
37 printf("\n (a) Reactance of transformer in pu , X_pu
   = %.2f pu \n",X_pu) ;
38 printf("\n (b) High-voltage side base impedance ,

```

```

        Z_B_HV = %.2f      \n",Z_B_HV) ;
39 printf("\n (c) Low-voltage side base impedance ,
        Z_B_LV = %.4f      \n",Z_B_LV) ;
40 printf("\n (d) Transformer reactance referred to
        High-voltage side , X_HV = %.2f      \n",X_HV) ;
41 printf("\n (e) Transformer reactance referred to Low
        -voltage side , X_LV = %.4f      \n",X_LV) ;
42 printf("      (or) From another equation C.89 ,") ;
43 printf("\n      Transformer reactance referred to Low
        -voltage side , X_LV = %.4f      \n",X_LV1) ;

```

Scilab code Exa 3.C determine turns ratio Xlv Xpu

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6 // APPENDIX C : REVIEW OF BASICS
7
8 // EXAMPLE : C.3 :
9 clear ; clc ; close ; // Clear the work space and
// console
10
11 // GIVEN DATA
12 X_pu = 12/100 ; // Leakage reactance in pu
13 kV_B_HV = 345 ; // HV side ratings in Y kV
14 kV_B_LV = 34.5 ; // LV side ratings in     kV
15 MVA_B = 20 ; // Base on HV side in MVA
16
17 // CALCULATIONS
18 // For case (a)
19 n = ( kV_B_HV/sqrt(3) )/kV_B_LV ; // Turns ratio of
// windings

```

```

20
21 // For case (b)
22 Z_B_HV = (kV_B_HV)^2/MVA_B ; // HV side base
   impedance in
23 X_HV = X_pu * Z_B_HV ; // Reactance referred to HV
   side in
24 X_LV = X_HV/(n^2) ; // transformer reactance
   referred to delta LV side in
25
26 // For case (c)
27 Z_dt = X_LV ;
28 Z_Y = Z_dt/3 ; // Reactance of equi wye connection
29 Z_B_LV = kV_B_LV^2/MVA_B ; // LV side base impedance
   in
30 X_pu1 = Z_Y/Z_B_LV ; // reactance in pu referred to
   LV side
31
32 // Alternative method For case (c)
33 n1 = kV_B_HV/kV_B_LV ; // Turns ratio if line-to-
   line voltages are used
34 X_LV1 = X_HV/(n1^2) ; // Reactance referred to LV
   side in
35 X_pu2 = X_LV1/Z_B_LV ; // reactance in pu referred
   to LV side
36
37 // DISPLAY RESULTS
38 disp("EXAMPLE : C.3 : SOLUTION :-") ;
39 printf("\n (a) Turns ratio of windings , n = %.4f \n",
   ,n) ;
40 printf("\n (b) Transformer reactance referred to LV
   side in ohms ,X_LV = %.4f \n" ,X_LV) ;
41 printf("\n (c) Transformer reactance referred to LV
   side in per units ,X_pu = %.2f pu \n" ,X_pu1) ;
42 printf("\n (or) From another equation if line-to-
   line voltages are used ,") ;
43 printf("\n Transformer reactance referred to LV
   side in per units ,X_pu = %.2f pu \n" ,X_pu2) ;

```

Scilab code Exa 4.C determine KVA KV Zb Ib I new Zpu V1 V2 V4 S1 S2 S4 table

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING  
2 // ANALYSIS AND DESIGN  
3 // TURAN GONEN  
4 // CRC PRESS  
5 // SECOND EDITION  
6  
7 // APPENDIX C : REVIEW OF BASICS  
8  
9 // EXAMPLE : C.4 :  
9 clear ; clc ; close ; // Clear the work space and  
// console  
10  
11 // GIVEN DATA  
12 I_1 = 1000 ; // Physical current in A for 2.4 kV  
// circuit  
13 Z_pu = 0.04 ; // Leakage reactance in pu  
14 I_pu = 2.08*exp(%i*(-90)*%pi/180) ; // Generator  
// supply for pure inductive load  
15 kVA_Bg1 = 6000 ; // Rated kVA values for T1  
16 kVA_Bg2 = 4000 ; // Rated kVA values for T2  
17 N2 = 2.4 ; // N2 = V2 in Y kV , refer fig C.4  
18 N1 = 24 ; // N1 = V1 in Y kV , refer fig C.4  
19 N3 = 24 ; // N3 = V3 = N1 in Y kV , refer fig C.4  
20 N4 = 12 ; // N4 = V4 in Y kV , refer fig C.4  
21  
22 // CALCULATIONS  
23 // For case (a)  
24 kVA_B = 2080 ; // arbitrarily selected kVA values  
// for all 3 ckt  
25  
26 // For case (b)  
27 n1 = N2/N1 ; // Turns ratio of transformer T1 & T2 i
```

```

.e N2/N1
28 n2 = N3/N4 ; // Turns ratio N1'/N2'
29 kV_BL_L1 = 2.5 ; // arbitrarily selected Base
    voltage for 2.4 kV ckt in kV
30 kV_BL_L2 = kV_BL_L1/n1 ; // arbitrarily selected
    Base voltage for 24 kV ckt in kV
31 kV_BL_L3 = kV_BL_L2/n2 ; // arbitrarily selected
    Base voltage for 12 kV ckt in kV
32
33 // For case (c)
34 Z_B1 = (kV_BL_L1)^2 * 1000/(kVA_B) ; // Base
    impedance in for 2.4 kV ckt
35 Z_B2 = (kV_BL_L2)^2 * 1000/(kVA_B) ; // Base
    impedance in for 24 kV ckt
36 Z_B3 = (kV_BL_L3)^2 * 1000/(kVA_B) ; // Base
    impedance in for 12 kV ckt
37
38 // For case (d)
39 I_B1 = kVA_B/(sqrt(3)*kV_BL_L1) ; // Base current in
    A for 2.4 kV ckt
40 I_B2 = kVA_B/(sqrt(3)*kV_BL_L2) ; // Base current in
    A for 24 kV ckt
41 I_B3 = kVA_B/(sqrt(3)*kV_BL_L3) ; // Base current in
    A for 12 kV ckt
42
43 // For case (e)
44 I_2 = (n1) * I_1 ; // Physical current in A for 24
    kV circuit
45 I_4 = (n2) * I_2 ; // Physical current in A for 12
    kV circuit
46
47 // For case (f)
48 I_pu_3ckt = abs(I_pu) ; // per-unit current values
    for all 3-ckt
49
50 // For case (g)
51 kV_B1 = N2 ; // Given voltage in kV
52 kV_B2 = N4 ; // Given voltage in kV

```

```

53 Z_pu_T1 = (%i)*Z_pu*(kVA_B/kVA_Bg1)*(kV_B1/kV_BL_L1)
      ^2 ; // New reactance of T1
54 Z_pu_T2 = (%i)*Z_pu*(kVA_B/kVA_Bg2)*(kV_B2/kV_BL_L3)
      ^2 ; // New reactance of T2
55
56 // For case (h)
57 V1 = kV_B1/kV_BL_L1 ; // voltage in pu at bus 1
58 V2 = V1 - I_pu * (Z_pu_T1) ; // voltage in pu at bus
      2
59 V4 = V2 - I_pu * (Z_pu_T2) ; // voltage in pu at bus
      3
60
61 // For case (i)
62 S1 = V1 * abs(I_pu) ; // Apparent power value at bus
      1 in pu
63 S2 = V2 * abs(I_pu) ; // Apparent power value at bus
      2 in pu
64 S4 = V4 * abs(I_pu) ; // Apparent power value at bus
      4 in pu
65
66 // DISPLAY RESULTS
67 disp("EXAMPLE : C.3 : SOLUTION :-") ;
68 printf("\n (a) Base kilovoltampere value for all 3-
      circuits is , kVA_B = %.1f kVA \n",kVA_B) ;
69 printf("\n (b) Base line-to-line kilovolt value for
      2.4 kV circuit , kV_BL_L = %.1f kV \n",kV_BL_L1)
      ;
70 printf("\n      Base line-to-line kilovolt value for
      24 kV circuit , kV_BL_L = %.1f kV \n",kV_BL_L2) ;
71 printf("\n      Base line-to-line kilovolt value for
      24 kV circuit , kV_BL_L = %.1f kV \n",kV_BL_L3) ;
72 printf("\n (c) Base impedance value of 2.4 kV
      circuit , Z_B = %.3f \n",Z_B1) ;
73 printf("\n      Base impedance value of 24 kV circuit
      , Z_B = %.1f \n",Z_B2) ;
74 printf("\n      Base impedance value of 12.5 kV
      circuit , Z_B = %.1f \n",Z_B3) ;
75 printf("\n (d) Base current value of 2.4 kV circuit

```

```

        , I_B = %d A \n" ,I_B1) ;
76 printf("\n      Base current value of 24 kV circuit ,
        I_B = %d A \n" ,I_B2) ;
77 printf("\n      Base current value of 2.4 kV circuit
        , I_B = %d A \n" ,I_B3) ;
78 printf("\n (e) Physical current of 2.4 kV circuit ,
        I = %.f A \n" ,I_1) ;
79 printf("\n      Physical current of 24 kV circuit , I
        = %.f A \n" ,I_2) ;
80 printf("\n      Physical current of 12 kV circuit , I
        = %.f A \n" ,I_4) ;
81 printf("\n (f) Per unit current values for all 3
        circuits , I_pu = %.2f pu \n" ,I_pu_3ckt) ;
82 printf("\n (g) New transformer reactance of T1 ,
        Z_pu_T1 = j%.4f pu \n" ,abs(Z_pu_T1)) ;
83 printf("\n      New transformer reactance of T2 ,
        Z_pu_T2 = j%.4f pu \n" ,abs(Z_pu_T2)) ;
84 printf("\n (h) Per unit voltage value at bus 1 ,V1 =
        %.2f<%1f pu \n" ,abs(V1),atand(imag(V1),real(V1)
        )) ;
85 printf("\n      Per unit voltage value at bus 2 ,V2 =
        %.4f<%1f pu \n" ,abs(V2),atand(imag(V2),real(V2)
        )) ;
86 printf("\n      Per unit voltage value at bus 4 ,V4 =
        %.4f<%1f pu \n" ,abs(V4),atand(imag(V4),real(V4)
        )) ;
87 printf("\n (i) Per-unit apparent power value at bus
        1 , S1 = %.2f pu \n" ,S1) ;
88 printf("\n      Per-unit apparent power value at bus
        2 , S2 = %.4f pu \n" ,S2) ;
89 printf("\n      Per-unit apparent power value at bus
        4 , S4 = %.4f pu \n" ,S4) ;
90 printf("\n (j) TABLE C.2 \n" ) ;
91 printf("\n      Results Of Example C.4 \n" ) ;
92 printf("\n
-----\n" ) ;
93 printf("\n      QUANTITY          \t 2.4-kV circuit    \t

```

```

24-kV circuit \t 12-kV circuit ") ;
94 printf("\n
-----;
95 printf("\n      kVA_B(3- ) \t %d kVA
      \t %d kVA \n",kVA_B,kVA_B,
      kVA_B) ;
96 printf("\n      kV_B(L-L) \t %.1f kV
      \t %.1f kV \n",kV_BL_L1,
      kV_BL_L2,kV_BL_L3) ;
97 printf("\n      Z_B \t %.3f
      \t %.1f \n",Z_B1,Z_B2,
      Z_B3) ;
98 printf("\n      I_B \t %d A
      \t %d A \n",I_B1,I_B2,I_B3) ;
99 printf("\n      I_physical \t %d A
      \t %.f A \n",I_1,I_2,I_4) ;
100 printf("\n      I_pu \t %.2f pu
      \t %.2f pu \n",I_pu_3ckt,
      I_pu_3ckt) ;
101 printf("\n      V_pu \t %.2f pu
      \t %.4f pu \n",abs(V1),abs(V2)
      ,abs(V4)) ;
102 printf("\n      S_pu \t %.2f pu
      \t %.4f pu \n",S1,S2,S4) ;
103 printf(
-----;

```

Scilab code Exa 5.C determine inductive reactance using equ C135 and tables

```

1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
2 // TURAN GONEN
3 // CRC PRESS

```

```

4 // SECOND EDITION
5
6 // APPENDIX C : REVIEW OF BASICS
7
8 // EXAMPLE : C.5 :
9 clear ; clc ; close ; // Clear the work space and
  console
10
11 // GIVEN DATA
12 D_ab = 6.8 ; // distance b/w conductors center-to-
  center in ft
13 D_bc = 5.5 ; // distance b/w conductors center-to-
  center in ft
14 D_ca = 4 ; // distance b/w conductors center-to-
  center in ft
15
16 // CALCULATIONS
17 // For case (a)
18 D_eq = (D_ab * D_bc * D_ca)^(1/3) ; // Equi spacing
  for pole top in ft
19 D_s = 0.01579 ; // GMR in ft From Table A.1
20 X_L = 0.1213 * log(D_eq/D_s) ; // Inductive
  reactance in /mi . From equ C.135
21
22 // For case (b)
23 X_a = 0.503 ; // Inductive reactance in /mi From
  Table A.1
24 X_d = 0.2026 ; // From Table A.8 for D_eq , by linear
  interpolation in /mi
25 X_L1 = X_a + X_d ; // Inductive reactance in /mi
26
27 // DISPLAY RESULTS
28 disp("EXAMPLE : C.5 : SOLUTION :-") ;
29 printf("\n (a) Inductive reactance using equation C
  .135 , X_L = %.4f /mi \n",X_L );
30 printf("\n (b) Inductive reactance using tables ,
  X_L = %.4f /mi \n",X_L1) ;

```

Scilab code Exa 6.C determine shunt capacitive reactance using equ C156 and tables

```
1 // ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
2 // ANALYSIS AND DESIGN
3 // TURAN GONEN
4 // CRC PRESS
5 // SECOND EDITION
6
7 // APPENDIX C : REVIEW OF BASICS
8
9 // EXAMPLE : C.6 :
9 clear ; clc ; close ; // Clear the work space and
10 console
11
12 // GIVEN DATA
13 D_ab = 6.8 ; // distance b/w conductors center-to-
14 // center in ft
13 D_bc = 5.5 ; // distance b/w conductors center-to-
14 // center in ft
14 D_ca = 4 ; // distance b/w conductors center-to-
14 // center in ft
15 l = 100 ; // Line length in miles
16
17 // CALCULATIONS
18 // For case (a)
19 D_m = (D_ab * D_bc * D_ca)^(1/3) ; // Equi spacing
20 // for pole top in ft
20 r = 0.522/(2 * 12) ; // feet
21 X_C = 0.06836 * log10 (D_m/r) ; // Shunt capacitive
22 // reactance in M *mi
23 // For case (b)
24 X_a = 0.1136 ; // Shunt capacitive reactance in M *
24 mi , From table A.1
```

```

25 X_d = 0.049543 ; // Shunt capacitive reactance
                     spacing factor in M *mi , From table A.9
26 X_C1 = X_a + X_d ; // Shunt capacitive reactance in
                     M *mi
27 X_C2 = X_C1/l ; // Capacitive reactance of 100 mi
                     line in M
28
29 // DISPLAY RESULTS
30 disp("EXAMPLE : C.6 : SOLUTION :-") ;
31 printf("\n (a) Shunt capacitive reactance using
          equation C.156 , X_C = %.6f M *mi \n",X_C) ;
32 printf("\n (b) Shunt capacitive reactance using
          tables , X_C = %.6f M *mi \n",X_C1) ;
33 printf("\n (c) Capacitive reactance of total line ,
          X_C = %.5e M \n",X_C2) ;

```
