Scilab Textbook Companion for Electric Power Transmission System Engineering Analysis And Design by T. Gonen[1]

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

June 12, 2014

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: 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

Chapter 2

TRANSMISSION LINE STRUCTURES AND EQUIPMENT

Scilab code Exa 2.1 calculate tolerable touch step potential

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 2 : TRANSMISSION LINE STRUCTURES AND EQUIPMENT

// EXAMPLE : 2.1 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
t_s = 0.49 ; // Human body is in contact with 60 Hz power for 0.49 sec
r = 100 ; // Resistivity of soil based on IEEE std
```

```
80-2000

// CALCULATIONS
// For case (a)
v_touch50 = 0.116*(1000+1.5*r)/sqrt(t_s) ; // Maximum allowable touch voltage for 50 kg body weight in volts

// For case (b)
v_step50  = 0.116*(1000+6*r)/sqrt(t_s) ; // Maximum allowable step voltage for 50 kg body weight in volts
// Above Equations of case (a) & (b) applicable if no protective surface layer is used

// For metal to metal contact below equation holds good . Hence resistivity is zero
r_1 = 0 ; // Resistivity is zero

// For case (c)
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

// For case (d)
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

// DISPLAY RESULTS
disp("EXAMPLE : 2.1 : SOLUTION :-") ;
printf("\n (a) Tolerable Touch potential , V_touch50 = %.f V , for 50 kg body weight \n",v_touch50) ;
printf("\n (b) Tolerable Step potential , V_step50 = %.f V , for 50 kg body weight \n",v_step50) ;
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) ;
```

```
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 3 : FUNDAMENTAL CONCEPTS

// EXAMPLE : 3.1 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
kV = 345 ; // Three phase transmission line voltage in kV
Z_s = 366 ; // Surge impedance of line in
a = 24.6 ; // Spacing between adjacent conductors in feet
d = 1.76 ; // Diameter of conductor in inches

// CALCULATIONS
```

```
SIL = (kV)^2/Z_s ; // Surge Impedance loading of
    line in MW

// DISPLAY RESULTS
disp("EXAMPLE : 3.1 : SOLUTION :-") ;
printf("\n Surge Impedance Loading of line , SIL = %
    .f MW \n",SIL) ;

printf("\n NOTE: Unit of SIL is MW and surge
    impedance is    ") ;
printf("\n ERROR: Mistake in unit of SIL in textbook
     \n") ;
```

Scilab code Exa 3.2 determine effective SIL

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 3 : FUNDAMENTAL CONCEPTS

// EXAMPLE : 3.2 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
SIL = 325 ; // Surge impedance Loading in MW . From
    exa 3.1
kV = 345 ; // Transmission line voltage in kV . From
     exa 3.1

// For case (a)
t_shunt1 = 0.5 ; // shunt capacitive compensation is
```

```
50%
t_series1 = 0 ; // no series compensation

// For case (b)
t_shunt2 = 0.5 ; // shunt compensation using shunt reactors is 50%
t_series2 = 0 ; // no series capacitive compensation

// For case (c)
t_shunt3 = 0 ; // no shunt compensation
t_series3 = 0.5 ; // series capacitive compensation is 50%

// For case (d)
t_shunt4 = 0.2 ; // shunt capacitive compensation is 20%
t_series4 = 0.5; // series capacitive compensation is 50%

// CALCULATIONS
// For case (a)
SIL1 = SIL*(sqrt( (1-t_shunt1)/(1-t_series1) )) ; // Effective SIL in MW

// For case (b)
SIL2 = SIL*(sqrt( (1+t_shunt2)/(1-t_series2) )) ; // Effective SIL in MW

// For case (c)
SIL3 = SIL*(sqrt( (1-t_shunt3)/(1-t_series3) )) ; // Effective SIL in MW

// For case (d)
SIL4 = SIL*(sqrt( (1-t_shunt4)/(1-t_series4) )) ; // Effective SIL in MW

// DISPLAY RESULTS
disp("EXAMPLE : 3.2 : SOLUTION :-") ;
```

```
printf("\n (a) Effective SIL , SIL_comp = %.f MW \n"
    ,SIL1) ;
printf("\n (b) Effective SIL , SIL_comp = %.f MW \n"
    ,SIL2) ;
printf("\n (c) Effective SIL , SIL_comp = %.f MW \n"
    ,SIL3) ;
printf("\n (d) Effective SIL , SIL_comp = %.f MW \n"
    ,SIL4) ;

printf("\n NOTE: Unit of SIL is MW and surge
    impedance is    ") ;
printf("\n ERROR: Mistake in unit of SIL in textbook
     \n") ;
```

Scilab code Exa 3.3 calculate RatedCurrent MVARrating CurrentValue

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 3 : FUNDAMENTAL CONCEPTS

// EXAMPLE : 3.3 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
// For case (c)
I_normal = 1000 ; // Normal full load current in
    Ampere

// CALCULATIONS
// For case (a) equation is (1.5pu)*I_rated = (2 pu)
```

```
*I_normal
// THEREFORE
// I_rated = (1.333pu)*I_normal ; // Rated current in terms of per unit value of the normal load current

// For case (b)
Mvar = (1.333)^2 ; // Increase in Mvar rating in per units

// For case (c)
I_rated = (1.333)*I_normal ; // Rated current value

// DISPLAY RESULTS
disp("EXAMPLE : 3.3 : SOLUTION :-") ;
printf("\n (a) Rated current , I_rated = (1.333 pu)*I_normal \n") ;
printf("\n (b) Mvar rating increase = %.2f pu \n", Mvar) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.1 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
V_RL_L = 23*10^3 ; // line to line voltage in volts
z_t = 2.48+%i*6.57 ; // Total impedance in ohm/phase
p = 9*10^6 ; // load in watts
pf = 0.85 ; // lagging power factor

```

```
// CALCULATIONS
// METHOD I : USING COMPLEX ALGEBRA

V_RL_N = (V_RL_L)/sqrt(3) ; // line-to-neutral reference voltage in V
I = (p/(sqrt(3)*V_RL_L*pf))*( pf - %i*sind(acosd(pf))) ; // Line current in amperes
IZ = I*z_t ;
V_SL_N = V_RL_N + IZ // Line to neutral voltage at sending end in volts
V_SL_L = sqrt(3)*V_SL_N ; // Line to line voltage at sending end in volts

// DISPLAY RESULTS
disp("EXAMPLE : 4.1 : SOLUTION :-") ;
disp("METHOD I : USING COMPLEX ALGEBRA") ;
printf("\n (a) Line-to-neutral voltage at sending end , V_SL_N = %.f<%.1f V \n",abs(V_SL_N),atand(imag(V_SL_N),real(V_SL_N) )) ;
printf("\n i.e Line-to-neutral voltage at sending end , V_SL_N = %.f V \n",abs(V_SL_N)) ;
printf("\n     Line-to-line voltage at sending end , V_SL_L = %.f<%.1f V \n",abs(V_SL_L),atand( imag(V_SL_L),real(V_SL_L) )) ;
printf("\n i.e Line-to-line voltage at sending end , V_SL_L = %.f V \n",abs(V_SL_L)) ;
printf("\n (b) load angle ,   = %.1f degree \n",atand( imag(V_SL_L),real(V_SL_L) )) ;
printf("\n") ;


// CALCULATIONS
// METHOD II : USING THE CURRENT AS REFERENCE PHASOR
theta_R = acosd(pf) ;
V1 = V_RL_N*cosd(theta_R) + abs(I)*real(z_t) ; // unit is volts
V2 = V_RL_N*sind(theta_R) + abs(I)*imag(z_t) ; // unit is volts
```

```
V_SL_N2 = sqrt( (V1^2) + (V2^2) ) ; // Line to neutral voltage at sending end in volts/phase
V_SL_L2 = sqrt(3) * V_SL_N2 ; // Line to line voltage at sending end in volts
theta_s = atand(V2/V1) ;
delta = theta_s - theta_R ;

// DISPLAY RESULTS
disp("METHOD II : USING THE CURRENT AS REFERENCE PHASOR");
printf("\n (a) Line-to-neutral voltage at sending end , V_SL_N = %.f V \n",V_SL_N2) ;
printf("\n     Line-to-line voltage at sending end , V_SL_L = %.f V \n",V_SL_L2) ;
printf("\n (b) load angle ,    = %.1f degree \n",delta) ;
printf("\n") ;

// CALCULATIONS
// METHOD III : USING THE RECEIVING-END VOLTAGE  AS REFERENCE PHASOR
// for case (a)
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) ;
V_SL_L3 = sqrt(3)*V_SL_N3 ;

// for case (b)
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)) ) ;

// DISPLAY RESULTS
disp("METHOD III : USING THE RECEIVING END VOLTAGE AS REFERENCE PHASOR") ;
printf("\n (a) Line-to-neutral voltage at sending
```

```
end , V_SL_N = %.f V \n",V_SL_N3) ;
printf("\n      Line-to-line voltage at sending end , V_SL_L = %.f V \n",V_SL_L3) ;
printf("\n (b) load angle ,    = %.1f degree \n",delta_3) ;
printf("\n") ;

// CALCULATIONS
// METHOD IV : USING POWER RELATIONSHIPS
P_4 = 9 ; // load in MW (Given)
P_loss = 3 * (abs(I))^2 * real(z_t) * 10^-6 ; // Power loss in line in MW
P_T = P_4 + P_loss ; // Total input power to line in MW
Q_loss = 3 * (abs(I))^2 * imag(z_t) * 10^-6 ; // Var loss of line in Mvar lagging
Q_T = ( (P_4*sind(theta_R))/cosd(theta_R) ) + Q_loss ; // Total megavar input to line in Mvar lagging
S_T = sqrt( (P_T^2)+(Q_T^2) ) ; // Total megavoltampere input to line
// for case (a)
V_SL_L4 = S_T*10^6/(sqrt(3) * abs(I)) ; // line to line voltage in volts
V_SL_N4 = V_SL_L4/sqrt(3) ; // Line to line neutral in volts

// for case (b)
theta_S4 = acosd(P_T/S_T) ; // Lagging
delta_4 = theta_s - theta_R ;

// DISPLAY RESULTS
disp("METHOD IV : USING POWER RELATIONSHIPS");
printf("\n (a) Line-to-neutral voltage at sending end , V_SL_N = %.f V \n",V_SL_N4) ;
printf("\n (a) Line-to-line voltage at sending end , V_SL_L = %.f V \n",V_SL_L4) ;
printf("\n (b) load angle ,    = %.1f degree \n",delta_4) ;
```

```
printf("\n");

// CALCULATIONS
// 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
// for case (a)
I_line = (p/2)/(V_RL_L * pf) ; // Power delivered is 4.5 MW
R_loop = 2*real(z_t) ;
X_loop = 2*imag(z_t) ;
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
V_SL_N5 = V_SL_L5/sqrt(3) ; // line to neutral voltage in V

// for case (b)
theta_S5 = atand((V_RL_L * sind(theta_R) + I_line * X_loop)/(V_RL_L * cosd(theta_R) + I_line*R_loop)) ;
delta_5 = theta_S5 - theta_R ;

// DISPLAY RESULTS
disp("METHOD V : TREATING 3-   LINE AS 1-   LINE") ;
printf("\n (a) Line to neutral voltage at sending end , V_SL_N = %.f V \n",V_SL_N5) ;
printf("\n (a) Line to line voltage at sending end , V_SL_L = %.f V \n",V_SL_L5) ;
printf("\n (b) load angle ,   = %.1f degree \n",delta_5) ;
printf("\n") ;

printf("\n NOTE : ERROR : Change in answer because root(3) = 1.73 is considered in Textbook ") ;
printf("\n But here sqrt(3) = 1.7320508 is considered \n") ;
```

Scilab code Exa 4.2 calculate percentage voltage regulation using equation

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION
// EXAMPLE : 4.2 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
// for case (a)
V_S = 14803 ; // sending end phase voltage at no load in volts . From exa 4.1
V_R = 13279.056 ; // receiving end phase voltage at full load in volts . From exa 4.1

// for case (b)
I_R = 265.78785 ; // Line current in amperes . From exa 4.1
z_t = 2.48+%i*6.57 ; // Total impedance in ohm/phase
pf = 0.85 ; // power factor
theta_R = acosd(pf) ;

// CALCULATIONS
// for case (a)
V_reg1 = ( (V_S - V_R)/V_R )*100 ; // percentage voltage regulation using equ 4.29

// for case (b)
```

```
V_reg2 = (I_R * ( real(z_t) * cosd(theta_R) + imag(z_t) * sind(theta_R) )/ V_R)*100 ; // percentage voltage regulation using equ 4.31

// DISPLAY RESULTS
disp("EXAMPLE : 4.2 : SOLUTION :-") ;
printf("\n (a) Percentage of voltage regulation using equ 4.29 = %.1f \n",V_reg1) ;
printf("\n (b) Percentage of voltage regulation using equ 4.31 = %.1f \n",V_reg2) ;

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.3 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
Z_xy = 0.09 + %i*0.3 ; // Mutual impedance between two parallel feeders in /mi per phase
Z_xx = 0.604*exp(%i*50.4*%pi/180) ; // Self impedance of feeders in /mi per phase
Z_yy = 0.567*exp(%i*52.9*%pi/180) ; // Self impedance of feeders in /mi per phase
```

```

// SOLUTION
Z_2 = Z_xx - Z_xy ; // mutual impedance between feeders
Z_4 = Z_yy - Z_xy ; // mutual impedance between feeders

// DISPLAY RESULTS
disp("EXAMPLE : 4.3 : SOLUTION :-") ;
printf("\n Mutual impedance at node 2 , Z_2 = %.3f + j%.3f \n",real(Z_2),imag(Z_2)) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.4 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
V = 138*10^3 ; // transmission line voltage in V
P = 49*10^6 ; // load power in Watts
pf = 0.85 ; // lagging power factor
Z = 95 * exp(%i*78*%pi/180) ; // line constants in

Y = 0.001 * exp(%i*90*%pi/180) ; // line constants
```

```
in siemens

// CALCULATIONS
V_RL_N = V/sqrt(3) ;
theta_R = acosd(pf) ;
I_R = P/(sqrt(3)*V*pf)*( cosd(theta_R) - %i*sind(theta_R) ) ; // receiving end current in ampere

// for case (a)
// A,B,C,D constants for nominal-T circuit representation
A = 1 + (1/2)*Y*Z ;
B = Z + (1/4)*Y*Z^2 ;
C = Y ;
D = A ;

// for case (b)
P = [A B ; C D] * [V_RL_N ; I_R] ;
V_SL_N = P(1,1) ; // Line-to-neutral Sending end voltage in V
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
// 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


// for case (c)
I_S = P(2,1); // Sending end current in A

// for case (d)
theta_s = atand( imag(V_SL_N),real(V_SL_N) ) - atand( imag(I_S),real(I_S) ) ;

// for case (e)
n = (sqrt(3) * V * abs(I_R) * cosd(theta_R)/(sqrt(3) * abs(I_S) * abs(V_SL_L) * cosd(theta_s) ))*100
```

```
; // Efficiency

// DISPLAY RESULTS
disp("EXAMPLE : 4.4 : SOLUTION :-") ;
printf("\n (a) A constant of line , A = %.4f<%.1f \n
  ",abs(A),atand( imag(A),real(A) )) ;
printf("\n      B constant of line , B = %.2f<%.1f
  \n",abs(B),atand( imag(B),real(B) )) ;
printf("\n      C constant of line , C = %.3f<%.1f S
  \n",abs(C),atand( imag(C),real(C) )) ;
printf("\n      D constant of line , D = %.4f<%.1f \n
  ",abs(D),atand( imag(D),real(D) )) ;
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) )) ;
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) )) ;
printf("\n (c) sending end current , I_S = %.2f<%.1f
  A \n",abs(I_S),atand( imag(I_S),real(I_S) )) ;
printf("\n (d) sending end power factor , cos _s =
  %.3f \n",cosd(theta_s)) ;
printf("\n (e) Efficiency of transmission ,    = %.2
  f Percentage \n",n) ;

printf("\n NOTE : From A = 0.9536<0.6 , magnitude is
  0.9536 & angle is 0.6 degree") ;
printf("\n ERROR : Change in answer because root(3)
  = 1.73 is considered in Textbook ") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.5 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
V = 138*10^3 ; // Transmission line voltage in V
P = 49*10^6 ; // load power in Watts
pf = 0.85 ; // lagging power factor
Z = 95 * exp(%i*78*%pi/180) ; // line constants in

Y = 0.001 * exp(%i*90*%pi/180) ; // line constants in siemens

// CALCULATIONS
V_RL_N = V/sqrt(3) ;
theta_R = acosd(pf) ;
I_R = P/(sqrt(3)*V*pf) * ( cosd(theta_R) - %i*sind(theta_R) ) ; // Receiving end current in A

// for case (a)
// A,B,C,D constants for nominal−   circuit representation
A = 1 + (1/2)*Y*Z ;
B = Z ;
C = Y + (1/4)*(Y^2)*Z ;
D = 1 + (1/2)*Y*Z ;

// for case (b)
P = [A B ; C D] * [V_RL_N ; I_R] ;
V_SL_N = P(1,1) ; // Line−to−neutral Sending end
```

```
voltage in V
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
// 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


// for case (c)
I_S = P(2,1); // Sending end current in A

// for case (d)
theta_s = atand( imag(V_SL_N),real(V_SL_N) ) - atand( imag(I_S),real(I_S) ) ;

// for case (e)
n = (sqrt(3) * V * abs(I_R) * cosd(theta_R)/(sqrt(3) * abs(I_S) * abs(V_SL_L) * cosd(theta_s) ))*100 ; // Efficiency

// DISPLAY RESULTS
disp("EXAMPLE : 4.5 : SOLUTION :-") ;
printf("\n (a) A constant of line , A = %.4f<%.1f \n",abs(A),atand( imag(A),real(A) )) ;
printf("\n      B constant of line , B = %.2f<%.1f \n",abs(B),atand( imag(B),real(B) )) ;
printf("\n      C constant of line , C = %.3f<%.1f S \n",abs(C),atand( imag(C),real(C) )) ;
printf("\n      D constant of line , D = %.4f<%.1f \n",abs(D),atand( imag(D),real(D) )) ;
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) )) ;
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) )) ;
printf("\n (c) sending end current , I_S = %.2f<%.1f
```

```
A \n",abs(I_S),atand( imag(I_S),real(I_S) )) ;
printf("\n (d) sending end power factor , cos _s = %.3f \n",cosd(theta_s)) ;
printf("\n (e) Efficiency of transmission ,     = %.2f Percentage \n",n) ;

printf("\n NOTE : ERROR : Change in answer because root(3) = 1.73 is considered in Textbook ") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.6 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
V_RL_L = 138*10^3 ; // transmission line voltage in V
R = 0.1858 // Line constant in   /mi
f = 60 // frequency in Hertz
L = 2.60*10^-3 // Line constant in H/mi
C = 0.012*10^-6 // Line constant in F/mi
pf = 0.85 // Lagging power factor
P = 50*10^6 // load in VA
l = 150 // length of 3−   transmission line in mi
```

```
// CALCULATIONS
z = R + %i*2*%pi*f*L ; // Impedance per unit length in /mi
y = %i*2*%pi*C*f ; // Admittance per unit length in S/mi
g = sqrt(y*z) ; // Propagation constant of line per unit length
g_l = real(g) * l + %i * imag(g) * l ; // Propagation constant of line
Z_c = sqrt(z/y) ; // Characteristic impedance of line
V_RL_N = V_RL_L/sqrt(3) ;
theta_R = acosd(pf) ;
I_R = P/(sqrt(3)*V_RL_L)*( cosd(theta_R) - %i*sind(theta_R) ) ; // Receiving end current in A

// for case (a)
// A,B,C,D constants of line
A = cosh(g_l) ;
B = Z_c * sinh(g_l) ;
C = (1/Z_c) * sinh(g_l) ;
D = A ;

// for case (b)
P = [A B ; C D] * [V_RL_N ; I_R] ;
V_SL_N = P(1,1) ; // Line-to-neutral Sending end voltage in V
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
// 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

// for case (c)
I_S = P(2,1); // Sending end current in A

```

```
// for case (d)
theta_s = atand( imag(V_SL_N),real(V_SL_N) ) - atand( imag(I_S),real(I_S) ) ; // Sending-end pf

// For case (e)
P_S = sqrt(3) * abs(V_SL_L) * abs(I_S) * cosd(theta_s) ; // Sending end power

// For case (f)
P_R = sqrt(3)*abs(V_RL_L)*abs(I_R)*cosd(theta_R) ; // Receiving end power
P_L = P_S - P_R ; // Power loss in line

// For case (g)
n = (P_R/P_S)*100 ; // Transmission line efficiency

// For case (h)
reg = (( abs(V_SL_N) - V_RL_N )/V_RL_N )*100 ; // Percentage of voltage regulation

// For case (i)
Y = y * l ; // unit is S
I_C = (1/2) * Y * V_SL_N ; // Sending end charging current in A

// For case (j)
Z = z * l ;
V_RL_N0 = V_SL_N - I_C*Z ;
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

// DISPLAY RESULTS
disp("EXAMPLE : 4.6 :SOLUTION :-") ;
printf("\n (a) A constant of line , A = %.4f<%.2f \n",abs(A),atand( imag(A),real(A) )) ;
printf("\n     B constant of line , B = %.2f<%.2f \n",abs(B),atand( imag(B),real(B) )) ;
```

```
printf("\n      C constant of line , C = %.5f<%.2f S
   \n",abs(C),atand( imag(C),real(C) )) ;
printf("\n      D constant of line , D = %.4f<%.2f \n
   ",abs(D),atand( imag(D),real(D) )) ;
printf("\n (b) Sending end line-to-neutral voltage ,
    V_SL_N = %.2f<%.2f V \n",abs(V_SL_N),atand( imag
   (V_SL_N),real(V_SL_N) )) ;
printf("\n      Sending end line-to-line voltage ,
   V_SL_L = %.2f<%.2f V \n",abs(V_SL_L),atand( imag(
   V_SL_L),real(V_SL_L) )) ;
printf("\n (c) sending-end current , I_S = %.2f<%.2f
    A \n",abs(I_S),atand( imag(I_S),real(I_S) )) ;
printf("\n (d) sending-end power factor , cos _s =
   %.4f \n",cosd(theta_s)) ;
printf("\n (e) sending-end power , P_S = %.5e W \n",
   P_S) ;
printf("\n (f) Power loss in line , P_L = %.5e W \n"
   ,P_L) ;
printf("\n (g) Transmission line Efficiency ,    = %
   .1f Percentage\n",n) ;
printf("\n (h) Percentage of voltage regulation = %
   .1f Percentage \n",reg) ;
printf("\n (i) Sending-end charging current at no
   load , I_C = %.2f A \n",abs(I_C)) ;
printf("\n (j) Receiving-end voltage rise at no load
    ,V_RL_N = %.2f<%.2f V \n",abs(V_RL_N0),atand(
   imag(V_RL_N0),real(V_RL_N0)));
printf("\n      Line-to-line voltage at receiving end
    at no load ,V_RL_L = %.2f<%.2f V \n",abs(V_RL_L0
   ),atand(imag(V_RL_L0),real(V_RL_L0)));

printf("\n NOTE : ERROR : Change in answer because
   root(3) = 1.73 is considered in Textbook & change
    in    &    values ") ;
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

```

// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.7 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
R = 0.1858 // Line constant in   /mi
f = 60 // frequency in Hertz
L = 2.60*10^-3 // Line constant in H/mi
C = 0.012*10^-6 // Line constant in F/mi
l = 150 // length of 3-   transmission line in mi

// CALCULATIONS
z = R + %i*2*%pi*f*L ; // Impedance per unit length in   /mi
y = %i*2*%pi*C*f ; // Admittance per unit length in S/mi
g = sqrt(y*z) ; // Propagation constant of line per unit length
g_l = real(g) * l + %i * imag(g) * l ; // Propagation constant of line
Z_c = sqrt(z/y) ; // Characteristic impedance of line

```

```
A = cosh(g_l) ;
B = Z_c * sinh(g_l) ;
C = (1/Z_c) * sinh(g_l) ;
D = A ;
Z_pi = B ;
Y_pi_by2 = (A-1)/B ; // Unit in Siemens
Z = l * z ; // unit in ohms
Y = y * l ;
Y_T = C ;
Z_T_by2 = (A-1)/C ; // Unit in

// DISPLAY RESULTS
disp("EXAMPLE : 4.7 : SOLUTION :-") ;
printf("\n FOR EQUIVALENT-  CIRCUIT ") ;
printf("\n   Z_  = B = %.2f<%.2f    \n",abs(Z_pi),atand( imag(Z_pi),real(Z_pi) )) ;
printf("\n   Y_  /2 = %.6f<%.2f S \n",abs(Y_pi_by2),atand( imag(Y_pi_by2),real(Y_pi_by2) )) ;
printf("\n FOR NOMINAL-  CIRCUIT ") ;
printf("\n   Z = %.3f<%.2f    \n",abs(Z),atand( imag(Z),real(Z) )) ;
printf("\n   Y/2 = %.6f<%.1f S \n",abs(Y/2),atand( imag(Y/2),real(Y/2) )) ;
printf("\n FOR EQUIVALENT-T CIRCUIT ") ;
printf("\n   Z_T/2 = %.2f<%.2f    \n",abs(Z_T_by2),atand( imag(Z_T_by2),real(Z_T_by2) )) ;
printf("\n   Y_T = C = %.5f<%.2f S \n",abs(Y_T),atand( imag(Y_T),real(Y_T) )) ;
printf("\n FOR NOMINAL-T CIRCUIT ") ;
printf("\n   Z/2 = %.2f<%.2f    \n",abs(Z/2),atand( imag(Z/2),real(Z/2) )) ;
printf("\n   Y = %.6f<%.1f S \n",abs(Y),atand( imag(Y),real(Y) )) ;
```

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.8 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
V_RL_L = 138*10^3 ; // transmission line voltage in
   V
R = 0.1858 // Line constant in  /mi
f = 60 // frequency in Hertz
L = 2.60*10^-3 // Line constant in H/mi
C = 0.012*10^-6 // Line constant in F/mi
pf = 0.85 // Lagging power factor
P = 50*10^6 // load in VA
l = 150 // length of 3-  transmission line in mi

// CALCULATIONS
// For case (a)
z = R + %i*2*%pi*f*L ; // Impedance per unit length
   in  /mi
y = %i*2*%pi*C*f ; // Admittance per unit length in
   S/mi
g = sqrt(y*z) ; // Propagation constant of line per
   unit length

// For case (b)
lamda = (2 * %pi)/imag(g) ; // Wavelength of
   propagation in mi
```

```
V = lamda * f ; // Velocity of propagation in mi/sec

 // For case (c)
Z_C = sqrt(z/y) ;
V_R = V_RL_L/sqrt(3) ;
theta_R = acosd(pf) ;
I_R = P/(sqrt(3)*V_RL_L) * ( cosd(theta_R) - %i*sind(theta_R) ) ; // Receiving end current in A
V_R_incident = (1/2)*(V_R + I_R*Z_C) ; // Incident voltage at receiving end in V
V_R_reflected = (1/2)*(V_R - I_R*Z_C) ; // Reflected voltage at receiving end in V

// For case (d)
V_RL_N = V_R_incident + V_R_reflected ; // Line-to-neutral voltage at receiving end in V
V_RL_L = sqrt(3)*V_RL_N // Receiving end Line voltage in V

// For case (e)
g_l = real(g) * l + %i * imag(g) * l ; // Propagation constant of line
a = real(g) ; // a =    is the attenuation constant
b = imag(g) ; // b =    is the phase constant
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
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

// For case (f)
V_SL_N = V_S_incident + V_S_reflected ; // Line-to-neutral voltage at sending end in V
V_SL_L = sqrt(3)*V_SL_N ; // sending end Line voltage in V

// DISPLAY RESULTS
```

```
disp("EXAMPLE : 4.8 : SOLUTION :-") ;
printf("\n (a) Attenuation constant ,     = %.4f Np/mi \n",real(g)) ;
printf("\n     Phase change constant ,     = %.4f rad/mi \n",imag(g)) ;
printf("\n (b) Wavelength of propagation = %.2f mi \n",lamda) ;
printf("\n     velocity of propagation = %.2f mi/s \n",V) ;
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));
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)) ;
printf("\n (d) Line voltage at receiving end , V_RL_L = %d V \n",V_RL_L) ;
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)) ;
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)) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
```

```
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.9 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
L = 2.60 * 10^-3 ; // Inductance of line in H/mi
R = 0.1858 ; // Resistance of line in /mi
C = 0.012 * 10^-6 ; // Capacitance in F/mi
kV = 138 ; // Transmission line voltage in kV
Z_c1 = 469.60085 // Characteristic impedance of line in . Obtained from example 4.6

// CALCULATIONS
Z_c = sqrt(L/C) ; // Approximate value of surge Impedance of line in ohm
SIL = kV^2/Z_c ; // Approximate Surge impedance loading in MW
SIL1 = kV^2/Z_c1 ; // Exact value of SIL in MW

// DISPLAY RESULTS
disp("EXAMPLE : 4.9 : SOLUTION :-") ;
printf("\n Approximate value of SIL of transmission line , SIL_app = %.3f MW\n",SIL) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
```

```
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.10 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
Z_1 = 10 * exp(%i*(30)*%pi/180) ; // Impedance in
Z_2 = 40 * exp(%i*(-45)*%pi/180) ; // Impedance in


// CALCULATIONS
P = [1 ,Z_1 ; 0 , 1]; // For network 1
Y_2 = 1/Z_2 ; // unit is S
Q = [1 0 ; Y_2 1]; // For network 2
EQ = P * Q ;

// DISPLAY RESULTS
disp("EXAMPLE : 4.10 : SOLUTION :-") ;
printf("\n Equivalent A , B , C , D constants are \n
    ") ;
printf("\n A_eq = %.3f<%.1f \n",abs( EQ(1,1) ),atand
    ( imag(EQ(1,1)),real(EQ(1,1)) )) ;
printf("\n B_eq = %.3f<%.1f \n",abs( EQ(1,2) ),atand
    ( imag(EQ(1,2)),real(EQ(1,2)) )) ;
printf("\n C_eq = %.3f<%.1f \n",abs( EQ(2,1) ),atand
    ( imag(EQ(2,1)),real(EQ(2,1)) )) ;
printf("\n D_eq = %.3f<%.1f \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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.11 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
Z_1 = 10*exp(%i*(30)*%pi/180) ; // Impedance in
Z_2 = 40*exp(%i*(-45)*%pi/180) ; // Impedance in
Y_2 = 1/Z_2 ;
A_1 = 1 ;
B_1 = Z_1 ;
C_1 = 0 ;
D_1 = 1 ;
A_2 = 1 ;
B_2 = 0 ;
C_2 = Y_2 ;
D_2 = 1 ;

// CALCULATIONS
P = [A_1 B_1 ; C_1 D_1]; // For network 1
Q = [A_2 B_2 ; C_2 D_2]; // For network 2
A_eq = ( A_1*B_2 + A_2*B_1 )/( B_1 + B_2 ) ; //
   Constant A
B_eq = ( B_1*B_2 )/(B_1 + B_2) ; // Constant B
C_eq = C_1 + C_2 + ( (A_1 - A_2) * (D_2 -D_1)/(B_1 +
    B_2) ) ; // Constant C
D_eq = ( D_1*B_2 + D_2*B_1 )/(B_1+B_2) ; // Constant
    D

// DISPLAY RESULTS
disp("EXAMPLE : 4.11 : SOLUTION :-") ;
```

```
printf("\n Equivalent A , B , C , D constants are \n") ;
printf("\n A_eq = %.2f<%.f \n",abs(A_eq),atand( imag(A_eq),real(A_eq) )) ;
printf("\n B_eq = %.2f<%.f \n",abs(B_eq),atand( imag(B_eq),real(B_eq) )) ;
printf("\n C_eq = %.3f<%.f \n",abs(C_eq),atand( imag(C_eq),real(C_eq) )) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.12 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
Z = 2.07 + 0.661 * %i ; // Line impedance in
V_L = 2.4 * 10^3 ; // Line voltage in V
p = 200 * 10^3; // Load in VA
pf = 0.866 ; // Lagging power factor

// CALCULATIONS
// for case (a)
A = 1 ;
B = Z ;
```

```
C = 0 ;
D = A ;
theta = acosd(pf) ;
S_R = p * ( cosd(theta) + %i * sind(theta) ) ; // Receiving end power in VA
I_L1 = S_R/V_L ;
I_L = conj(I_L1) ;
I_S = I_L ; // sending end current in A
I_R = I_S ; // Receiving end current in A

// for case (b)
Z_L = V_L/I_L ; // Impedance in
V_R = Z_L * I_R ;
V_S = A * V_R + B * I_R ; // sending end voltage in V
P = [A B ;C D] * [V_R ; I_R] ;

// for case (c)
V_S = P(1,1) ;
I_S = P(2,1) ;
Z_in = V_S/I_S ; // Input impedance in

// for case (d)
S_S = V_S * conj(I_S) ;
S_L = S_S - S_R ; // Power loss of line in VA

// DISPLAY RESULTS
disp("EXAMPLE : 4.12 : SOLUTION :-") ;
printf("\n (a) Sending-end current , I_S = %.2f<%.2f A \n",abs(I_S),atand( imag(I_S),real(I_S) )) ;
printf("\n (b) Sending-end voltage , V_S = %.2f<%.2f V \n",abs(V_S),atand( imag(V_S),real(V_S) )) ;
printf("\n (c) Input impedance , Z_in = %.2f<%.2f \n",abs(Z_in),atand( imag(Z_in),real(Z_in) )) ;
printf("\n (d) Real power loss in line , S_L = %.2f W \n",real(S_L)) ;
printf("\n     Reactive power loss in line , S_L = %.2f var \n",imag(S_L)) ;
```

Scilab code Exa 4.13 calculate SIL Pmax Qc Vroc

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.13 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
KV = 345 ; // Transmission line voltage in kV
V_R = KV ;
V_S = KV ;
x_L = 0.588 ;// Inductive reactance in  /mi/phase
b_c = 7.20*10^-6 ;// susceptance S phase to neutral per phase
l = 200 ;// Total line length in mi

// CALCULATIONS
// for case (a)
x_C = 1/b_c ;//  /mi/phase
Z_C = sqrt(x_C * x_L) ;
SIL = KV^2/Z_C ; // Surge impedance loading in MVA/mi . [1MVA = 1MW]
SIL1 = (KV^2/Z_C) * l ; // Surge impedance loading of line in MVA . [1MVA = 1MW]

// for case (b)
delta = 90 ; // Max 3-  theoretical steady-state
```

```
    power flow limit occurs for  = 90 degree
X_L = x_L * l ; // Inductive reactance  /phase
P_max = V_S * V_R * sind(delta)/(X_L) ;

// for case (c)
Q_C = V_S^2 * (b_c * l/2) + V_R^2 *( b_c * l/2) ; // Total 3-  magnetizing var in Mvar

// for case (d)
g = %i * sqrt(x_L/x_C) ; // rad/mi
g_l = g * l ; // rad
V_R_oc = V_S / cosh(g_l) ; // Open-circuit receiving-end voltage in kV
X_C = x_C * 2 / l ;
V_R_oc1 = V_S * ( - %i * X_C/( - %i * X_C + %i * X_L ) ) ; // Alernative method to find Open-circuit receiving-end voltage in kV

// DISPLAY RESULTS
disp("EXAMPLE : 4.13 : SOLUTION :-") ;
printf("\n (a) Total 3-  SIL of line , SIL = %.2f MVA/mi \n",SIL) ;
printf("\n     Total 3-  SIL of line for total line length , SIL = %.2f MVA \n",SIL1) ;
printf("\n (b) Maximum 3-  theortical steady-state power flow limit , P_max = %.2f MW \n",P_max) ;
printf("\n (c) Total 3-  magnetizing var generation by line capacitance , Q_C = %.2f Mvar \n",Q_C) ;
printf("\n (d) Open-circuit receiving-end voltage if line is open at receiving end , V_R_oc = %.2f kV \n",V_R_oc) ;
printf("\n     From alternative method ,") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.14 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
KV = 345 ; // Transmission line voltage in kV
V_R = KV ; // Sending end voltage in kV
x_L = 0.588 ;// Inductive reactance in  /mi/phase
b_c = 7.20*10^-6 ;// susceptance S phase to neutral per phase
l = 200 ;// Total line length in mi
per = 60/100 ; // 2 shunt reactors absorb 60% of total 3-  magnetizing var
cost = 10 ; // cost of each reactor is $10/kVA

// CALCULATIONS
// For case (a)
x_C = 1/b_c ;//  /mi/phase
Z_C = sqrt(x_C * x_L) ;
SIL = KV^2/Z_C ; // Surge impedance loading in MVA/mi
SIL1 = (KV^2/Z_C) * l ; // Surge impedance loading of line in MVA . [1MVA = 1MW]

```

```
// For case (b)
delta = 90 ; // Max 3−   theoretical steady−state power flow limit occurs for    = 90 degree
V_S = V_R ; // sending end voltage in kV
X_L = x_L * l ; // Inductive reactance   /phase
P_max = V_S * V_R * sind(delta)/(X_L) ;

// For case (c)
Q_C = V_S^2 * (b_c * l/2) + V_R^2 *( b_c * l/2) ; // Total 3−   magnetizing var in Mvar
Q = (1/2) * per * Q_C ; // 3−   megavoltampere rating of each reactor . Q = (1/2)*Q_L

// For case (d)
Q_L1 = Q * 10^3 ; // Total 3−   magnetizing var in Kvar
T_cost = Q_L1 * cost ; // Cost of each reactor in $

// For case (e)
g = %i * sqrt(x_L * (1-per)/x_C) ; // rad/mi
g_l = g * l ; // rad
V_R_oc = V_S/cosh(g_l) ; // Open circuit receiving−end voltage in kV
X_L = x_L *l ;
X_C = (x_C * 2) / (l * (1 - per)) ;
V_R_oc1 = V_S * ( -%i*X_C/(-%i*X_C + %i*X_L) ) ; // Alernative method to find Open−circuit receiving−end voltage in kV

// DISPLAY RESULTS
disp("EXAMPLE : 4.14 : SOLUTION :−") ;
printf("\n (a) Total 3−phase SIL of line , SIL = %.2f MVA/mi \n",SIL) ;
printf("\n     Total 3−   SIL of line for total line length , SIL = %.2f MVA \n",SIL1) ;
printf("\n (b) Maximum 3−phase theortical power flow , P_max = %.2f MW \n",P_max) ;
printf("\n (c) 3−phase MVA rating of each reactor ,
```

```
    (1/2)Q_L = %.2f MVA \n",Q);
printf("\n (d) Cost of each reactor at $10/kVA = $ %.2f \n",T_cost);
printf("\n (e) Open circuit receiving voltage , V_Roc= %.2f kV \n",V_R_oc);
printf("\n    From alternative method ,");
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 4.15 calculate La XL Cn Xc

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 4 : OVERHEAD POWER TRANSMISSION

// EXAMPLE : 4.15 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
D_12 = 26 ; // distances in feet
D_23 = 26 ; // distances in feet
D_31 = 52 ; // distances in feet
d = 12 ; // Distance b/w 2 subconductors in inches
f = 60 ; // frequency in Hz
kv = 345 ; // voltage base in kv
p = 100 ; // Power base in MVA
l = 200 ; // length of line in km

// CALCULATIONS
```

```
// For case (a)
D_S = 0.0435 ; // from A.3 Appendix A . Geometric mean radius in feet
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]
D_eq = (D_12 * D_23 * D_31 * 0.3048^3)^(1/3) ; // Equ GMR in meter
L_a = 2 * 10^-7 * log(D_eq/D_bS); // Inductance in H /meter

// For case (b)
X_L = 2 * %pi * f * L_a ; // inductive reactance/ phase in ohms/m
X_L0 = X_L * 10^3 ; // inductive reactance/phase in ohms/km
X_L1 = X_L0 * 1.609 ;// inductive reactance/phase in ohms/mi [1 mi = 1.609 km]

// For case (c)
Z_B = kv^2 / p ; // Base impedance in
X_L2 = X_L0 * 1/Z_B ; // Series reactance of line in pu

// For case (d)
r = 1.293*0.3048/(2*12) ; // radius in m . outside diameter is 1.293 inch given in A.3
D_bsC = sqrt(r * d * 0.0254) ;
C_n = 55.63 * 10^-12/log(D_eq/D_bsC) ; // capacitance of line in F/m

// For case (e)
X_C = 1/( 2 * %pi * f * C_n ) ; // capacitive reactance in ohm-m
X_C0 = X_C * 10^-3 ; // capacitive reactance in ohm-km
X_C1 = X_C0/1.609 ; // capacitive reactance in ohm-mi
```

```

// DISPLAY RESULTS
disp("EXAMPLE : 4.15 : SOLUTION :-") ;
printf("\n (a) Average inductance per phase , L_a = %.4e H/m \n",L_a) ;
printf("\n (b) Inductive reactance per phase , X_L = %.4f  /km \n",X_L0) ;
printf("\n     Inductive reactance per phase , X_L = %.4f  /mi \n",X_L1) ;
printf("\n (c) Series reactance of line , X_L = %.4f pu \n",X_L2) ;
printf("\n (d) Line-to-neutral capacitance of line , C_n = %.4e F/m \n",C_n);
printf("\n (e) Capacitive reactance to neutral of line , X_C = %.3e  -km \n",X_C0) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
   GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.1 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
d = 2 ; // Diameter  of conductor in cm
```

```
D = 5 ; // Inside diameter of lead sheath in cm
V = 24.9 ; // Line-to-neutral voltage in kV

// CALCULATIONS
// For case (a)
r = d/2 ;
R = D/2 ;
E_max = V/( r * log(R/r) ) ; // Maximum electric
    stress in kV/cm
E_min = V/( R * log(R/r) ) ; // Minimum electric
    stress in kV/cm

// For case (b)
r_1 = R/2.718 ; // Optimum conductor radius in cm .
    From equ 5.15
E_max1 = V/( r_1 * log(R/r_1) ) ; // Min value of
    max stress in kV/cm

// DISPLAY RESULTS
disp("EXAMPLE : 5.1 : SOLUTION :-") ;
printf("\n (a) Maximum value of electric stress ,
    E_max = %.2f kV/cm \n",E_max) ;
printf("\n     Minimum value of electric stress ,
    E_min = %.2f kV/cm \n",E_min) ;
printf("\n (b) Optimum value of conductor radius , r
     = %.2f cm \n",r_1) ;
printf("\n     Minimum value of maximum stress ,
    E_max = %.2f kV/cm \n",E_max1) ;
```

Scilab code Exa 5.2 calculate potential gradient E1

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
```

```
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND GAS–INSULATED TRANSMISSION LINES

// EXAMPLE : 5.2 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
r = 1 ; // Radius of conductor in cm
t_1 = 2 ; // Thickness of insulation layer in cm
r_1 = r + t_1 ;
r_2 = 2 ; // Thickness of insulation layer in cm . r_2 = t_1 = t_2
R = r_1 + r_2 ;
K_1 = 4 ; // Inner layer Dielectric constant
K_2 = 3 ; // Outer layer Dielectric constant
kv = 19.94 ; // potential difference b/w inner & outer lead sheath in kV

// CALCULATIONS
// E_1 = 2q/(r*K_1) & E_2 = 2q/(r_1*K_2) . Let E = E_1/E_2
E = ( r_1 * K_2 )/( r * K_1 ) ; // E = E_1/E_2
V_1 = poly(0,'V_1') ; // defining unknown V_1
E_1 = V_1/( r * log(r_1/r) ) ;
V_2 = poly(0,'V_2') ; // defining unknown V_2
V_2 = kv - (V_1) ;
E_2 = V_2/( r_1 * log(R/r_1) ) ;
E_3 = E_1/E_2 ;
// Equating E = E_3 . we get the value of V_1
V_1 = 12.30891068 ; // Voltage in kV
E_1s = V_1/( r * log(r_1/r) ) ; // Potential gradient at surface of conductor in kV/cm . E_1 = E_1s

// DISPLAY RESULTS
```

```
disp("EXAMPLE : 5.2 : SOLUTION :-") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.3 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
D = 1.235 ; // Inside diameter of sheath in inch
d = 0.575 ; // Conductor diameter in inch
kv = 115 ; // Voltage in kV
l = 6000 ; // Length of cable in feet
r_si = 2000 ; // specific insulation resistance is 2000 M /1000ft . From Table 5.2

// CALCULATIONS
// For case (a)
r_si0 = r_si * l/1000 ;
R_i = r_si0 * log10 (D/d) ; // Total Insulation resistance in M

// For case (b)
P = kv^2/R_i ; // Power loss due to leakage current
```

```
    in W

// DISPLAY RESULTS
disp("EXAMPLE : 5.3 : SOLUTION :-") ;
printf("\n (a) Total insulation resistance at 60
    degree F , R_i= %.2f M \n",R_i) ;
printf("\n (b) Power loss due to leakage current , V
    ^2/R_i = %.4f W \n",P) ;

printf("\n NOTE : ERROR : Mistake in textbook case (
    a) \n") ;
```

Scilab code Exa 5.4 calculate charging current Ic

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
    GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.4 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
C_a = 2 * 10^-6 ; // Capacitance b/w two conductors
    in F/mi
l = 2 ; // length in mi
f = 60 ; // Frequency in Hz
V_L_L = 34.5 * 10^3 ; // Line-to-line voltage in V

// CALCULATIONS
```

```
C_a1 = C_a * l ; // Capacitance for total cable length in F
C_N = 2 * C_a1 ; // capacitance of each conductor to neutral in F . From equ 5.56
V_L_N = V_L_L/sqrt(3) ; // Line-to-neutral voltage in V
I_c = 2 * %pi * f * C_N * (V_L_N) ; // Charging current of cable in A

// DISPLAY RESULTS
disp("EXAMPLE : 5.4 : SOLUTION :-") ;
printf("\n Charging current of the cable , I_c = %.2f A \n",I_c) ;
```

Scilab code Exa 5.5 calculate Ic Is pf

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.5 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
C_a = 0.45 * 10^-6 ; // Capacitance b/w two conductors in F/mi
l = 4 ; // length of cable in mi
f = 60 ; // Freq in Hz
V_L_L = 13.8 * 10^3 ; // Line-to-line voltage in V
```

```
pf = 0.85 ; // lagging power factor
I = 30 ; // Current drawn by load at receiving end in A

// CALCULATIONS
// For case (a)
C_a1 = C_a * l ; // Capacitance for total cable length in F
C_N = 2 * C_a1 ; // capacitance of each conductor to neutral in F
V_L_N = V_L_L/sqrt(3) ; // Line-to-neutral voltage in V
I_c = 2 * %pi * f * C_N * (V_L_N) ; // Charging current in A
I_c1 = %i * I_c ; // polar form of Charging current in A

// For case (b)
phi_r = acosd(pf) ; // pf angle
I_r = I * ( cosd(phi_r) - sind(phi_r) * %i ) ; // Receiving end current in A
I_s = I_r + I_c1 ; // sending end current in A

// For case (c)
pf_s = cosd( atand( imag(I_s),real(I_s) ) ) ; // Lagging pf of sending-end

// DISPLAY RESULTS
disp("EXAMPLE : 5.5 : SOLUTION :-") ;
printf("\n (a) Charging current of feeder , I_c = %.2f A \n",I_c) ;
printf("\n     Charging current of feeder in complex form , I_c = i*%.2f A \n",imag(I_c1)) ;
printf("\n (b) Sending-end current , I_s = %.2f<%.2f A\n",abs(I_s),atand( imag(I_s),real(I_s) )) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
    GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.6 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
f = 60 ; // Freq in Hz
V_L_L = 138 ; // Line-to-line voltage in kV
T = 11/64 ; // Thickness of conductor insulation in
    inches
t = 5/64 ; // Thickness of belt insulation in inches
d =  0.575 ; // Outside diameter of conductor in
    inches

// CALCULATIONS
// For case (a)
T_1 = (T + t)/d ; // To find the value of geometric
    factor G for a single-conductor cable
G_1 = 2.09 ; // From table 5.3 , by interpolation
sf = 0.7858 ; // sector factor obtained for T_1 from
     table 5.3
G = G_1 * sf ; // real geometric factor

// For case (b)
```

```
V_L_N = V_L_L/sqrt(3) ; // Line-to-neutral voltage
    in V
K = 3.3 ; // Dielectric constant of insulation for
    impregnated paper cable
I_c = 3 * 0.106 * f * K * V_L_N/(1000 * G) ; //
    Charging current in A/1000ft

// DISPLAY RESULTS
disp("EXAMPLE : 5.6 : SOLUTION :-") ;
printf("\n (a) Geometric factor of cable using table
     5.3 , G_1 = %.3f \n",G) ;
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 Pdl Pdh

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
    GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.7 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
V_L_N = 7.2 ; // Line-to-neutral voltage in kV
d = 0.814 ; // Conductor diameter in inches
D = 2.442 ; // inside diameter of sheath in inches
K = 3.5 ; // Dielectric constant
pf = 0.03 ; // power factor of dielectric
```

```
l = 3.5 ; // length in mi
f = 60 ; // Freq in Hz
u = 1.3 * 10^7 ; // dielectric resistivity of insulation in M −cm

// CALCULATIONS
// For case (a)
r = d * 2.54/2 ; // conductor radius in cm . [1 inch = 2.54 cm]
R = D * 2.54/2 ; // Inside radius of sheath in cm
E_max = V_L_N/( r * log(R/r) ) ; // max electric stress in kV/cm

// For case (b)
C = 0.0388 * K/( log10 (R/r) ) ; // capacitance of cable in F/mi . From equ 5.29
C_1 = C * l ; // capacitance of cable for total length in F

// For case (c)
V_L_N1 = 7.2 * 10^3 ; // Line−to−neutral voltage in V
C_2 = C_1 * 10^-6 ; // capacitance of cable for total length in F
I_c = 2 * %pi * f * C_2 * (V_L_N1) ; // Charging current in A

// For case (d)
l_1 = l * 5280 * 12 * 2.54 ; // length in cm . [1 mi = 5280 feet] ; [1 feet = 12 inch]
R_i = u * log(R/r)/( 2 * %pi * l_1) ; // Insulation resistance in M

// For case (e)
P_lc = V_L_N^2/R_i ; // power loss in W

// For case (f)
P_dl = 2 * %pi * f * C_1 * V_L_N^2 * pf ; // Total
```

```
dielectric loss in W

// For case (g)
P_dh = P_dl - P_lc ; // dielectric hysteresis loss in W

// DISPLAY RESULTS
disp("EXAMPLE : 5.7 : SOLUTION :-") ;
printf("\n (a) Maximum electric stress occuring in cable dielectric , E_max = %.2f kV/cm \n",E_max) ;
printf("\n (b) Capacitance of cable , C = %.4f  F \n",C_1) ;
printf("\n (c) Charging current of cable , I_c = %.3f A \n",I_c) ;
printf("\n (d) Insulation resistance , R_i = %.2f M  \n",R_i) ;
printf("\n (e) Power loss due to leakage current , P_lc = %.2f W \n",P_lc) ;
printf("\n (f) Total dielectric loss , P_dl = %.2f W \n",P_dl) ;
printf("\n (g) Dielectric hysteresis loss , P_dh = %.2f W \n",P_dh) ;
```

Scilab code Exa 5.8 calculate Rdc Reff percent reduction

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND GAS-INSULATED TRANSMISSION LINES

```

```
// EXAMPLE : 5.8 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
l = 3 ; // underground cable length in mi
f = 60 ; // frequency in hertz

// CALCULATIONS
// For case (a)
R_dc = 0.00539 ; // dc resistance of cable in /1000ft , From table 5.5
R_dc1 = (R_dc/1000) * 5280 * 3 ; // Total dc resistance in  . [1 mi = 5280 feet]

// For case (b)
s_e = 1.233 ; // skin effect coefficient
R_eff = s_e * R_dc1 ; // Effective resistance in
percentage = ( (R_eff - R_dc1)/(R_dc1) ) * 100 ; // skin effect on effective resistance in %

// DISPLAY RESULTS
disp("EXAMPLE : 5.8 : SOLUTION :-") ;
printf("\n (a) Total dc resistance of the conductor , R_dc = %.4f  \n",R_dc1) ;
printf("\n (b) Effective resistance at 60 hz , R_eff = %.4f  \n",R_eff) ;
printf("\n     Skin effect on the Effective resistance in percent at 60 hz , R_eff = %.1f percent greater than for direct current\n", percentage) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.9 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
kV = 35 ; // voltage in kV
f = 60 ; // operating frequency of cable in hertz
d = 0.681 ; // diameter of conductor in inches
t_i = 345 ; // Insulation thickness in cmil
t_s = 105 ; // Metal sheet thickness in cmil
r_c = 0.190 ; // Conductor ac resistance in  /mi
l = 10 ; // Length of cable in mi

// CALCULATIONS
// For case (a)
T_i = t_i/1000 ; // insulation thickness in inch
T_s = t_s/1000 ; // Metal sheet thickness in inch
r_i = (d/2) + T_i ; // Inner radius of metal sheath in inches
r_0 = r_i + T_s ; // Outer radius of metal sheath in inches
S = r_i + r_0 + T_s ; // Spacing b/w conductor centers in inches
X_m = 0.2794 * (f/60) * log10 ( 2*S/(r_0 + r_i) ) ; // Mutual reactance b/w conductor & sheath per phase in  /mi . From Equ 5.78
X_m1 = X_m * l ; // Mutual reactance b/w conductor &
```

```
sheath in /phase

// For case (b)
r_s = 0.2/((r_0+r_i)*(r_0-r_i)) ; // sheet resistance per phase in /mi/phase . From equ 5.79
r_s1 = r_s * l ; // sheet resistance per phase in /phase

// For case (c)
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
d_r1 = d_r * l ; // // increase in conductor resistance due to sheath current in /phase

// For case (d)
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
r_a1 = r_a * l ; // Total positive or negative sequence resistance including sheath current effects in /phase

// For case (e)
ratio = d_r/r_c ; // ratio = sheath loss/conductor loss

// For case (f)
I = 400 ; // conductor current in A ( given for case (f) )
P_s = 3 * (I^2) * ( r_s * X_m^2)/( r_s^2 + X_m^2 ) ; // For three phase loss in W/mi
P_s1 = P_s * l ; // Total sheath loss of feeder in Watts

// DISPLAY RESULTS
```

```
disp("EXAMPLE : 5.9 : SOLUTION :-") ;
printf("\n (a) Mutual reactance b/w conductors & sheath , X_m = %.5f /mi/phase \n",X_m) ;
printf("\n    or Mutual reactance b/w conductors & sheath , X_m = %.4f /phase \n",X_m1) ;
printf("\n (b) Sheath resistance of cable , r_s = %.4f /mi/phase \n",r_s) ;
printf("\n    or Sheath resistance of cable , r_s = %.3f /phase \n",r_s1) ;
printf("\n (c) Increase in conductor resistance due to sheath currents ,  r = %.5f /mi/phase \n",d_r) ;
printf("\n    or Increase in conductor resistance due to sheath currents ,  r = %.4f /phase \n",d_r1) ;
printf("\n (d) Total resistance of conductor including sheath loss , r_a = %.5f /mi/phase \n",r_a) ;
printf("\n    or Total resistance of conductor including sheath loss , r_a = %.4f /phase \n ",r_a1) ;
printf("\n (e) Ratio of sheath loss to conductor loss , Ratio = %.4f \n",ratio) ;
printf("\n (f) Total sheath loss of feeder if current in conductor is 400A , P_s = %.2f W \n",P_s1) ;

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
```

```
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.10 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
f= 60 ; // frequency in hertz
t = 245 ; // insulation thickness in mils
t_s = 95 ; // Lead/metal sheath thickness in mils
d = 0.575 ; // diameter of conductor in inches
r_s = 1.72 ; // sheath resistance in  /mi
r_a = 0.263 ; // Conductor resistance in  /mi
r = 100 ; // earth resistivity in  -mi
D_s = 0.221 ; // GMR of one conductor in inches
D_ab = 24 ; // distance b/w conductor a & b in inch . refer fig 5.30
D_bc = 24 ; // distance b/w conductor b & c in inch . refer fig 5.30
D_ca = 48 ; // distance b/w conductor c & a in inch . refer fig 5.30

// CALCULATIONS
T = t/1000 ; // insulation thickness in inch . [1 mils = 0.001 inch]
T_s = t_s/1000 ; // Lead/metal sheath thickness in mils
r_i = (d/2) + T ; // Inner radius of metal sheath in inches
r_0 = r_i + T_s ; // Outer radius of metal sheath in inches
r_e = 0.00476 * f ; //  AC resistance of earth return in  /mi
D_e = 25920 * sqrt(r/f) ; // Equivalent depth of
```

```
earth return path in inches
D_eq = (D_ab*D_bc*D_ca)^(1/3) ; // Mean distance among conductor centers in inches
Z_0a = (r_a + r_e) + (%i) * (0.36396) * log(D_e/((D_s*D_eq^2)^(1/3))) ;
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
Z_0s = (r_s + r_e) + (%i) * 0.36396 * log (D_e/D_s_3s) ; // Zero sequence impedance of sheath in inches
D_m_3c_3s = D_s_3s ; // Zero sequence mutual impedance b/w conductors & sheaths in inches
Z_0m = r_e + (%i)*(0.36396)*log(D_e/D_m_3c_3s) ;

// For case (a)
Z_00 = Z_0a - (Z_0m^2/Z_0s) ; // Total zero sequence impedance when ground and return paths are present in  /mi/phase

// For case (b)
Z_0 = Z_0a + Z_0s - 2*Z_0m ; // Total zero sequence impedance when there is only sheath return path in  /mi/phase

// For case (c)
Z_01 = Z_0a ; // Total zero sequence impedance when there is only ground return path in  /mi/phase

// DISPLAY RESULTS
disp("EXAMPLE : 5.10 : SOLUTION :-") ;
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))) ;
printf("\n (b) Total zero sequence impedance when there is only sheath return path , Z_0 = %.3f<%.1f  /mi/phase \n",abs(Z_0),atand(imag(Z_0),real(
```

```
Z_0))) ;
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))) ;

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.11 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
f= 60 ; // frequency in hertz
T = 0.175 ; // insulation thickness in inches
d = 0.539 ; // diameter of conductor in inches
G = 0.5 ; // Geometric factor from fig 5.3
K = 3.7 ; // Dielectric constant
V_LL = 13.8 ; // Line-to-line voltage in kV

// CALCULATIONS
D = d + 2 * T ; // Inside diameter of sheath in inches
```

```
G = 2.303 * log10 (D/d) ; // Geometric factor for a single conductor
sf = 0.710 ; // sector factor From Table 5.3 . For (T+t/d) obtained
V_LN = V_LL/sqrt(3) ; // Line-to-neutral voltage in kV

// For case (a)
C_0 = 0.0892 * K/(G * sf) ; // shunt capacitances in F /mi/phase . C_0 = C_1 = C_2 . From equ 5.161

// For case (b)
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

// For case (c)
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

// DISPLAY RESULTS
disp("EXAMPLE : 5.11 : SOLUTION :-") ;
printf("\n (a) Shunt capacitances for zero , positive & negative sequences , C_0 = C_1 = C_2 = %.2f  F /mi/phase \n",C_0) ;
printf("\n (b) Shunt capacitive reactance for zero , positive & negative sequences , X_0 = X_1 = X_2 = %.2e  M /mi/phase \n",X_0) ;
printf("\n (c) Charging current for zero , positive & negative sequences , I_0 = I_1 = I_2 = %.3f A/mi/phase \n",I_0) ;

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND GAS–INSULATED TRANSMISSION LINES

// EXAMPLE : 5.12 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
f= 60 ; // frequency in hertz
r_a = 0.19 ; // Conductor resistance in   /mi
l = 10 ; // length in mi
D_s = 0.262 ; // GMR of one conductor in inches
d = 18 ; // conductors spacing in inches

// CALCULATIONS
// For case (a)
X_a = %i * 0.1213 *log (12/D_s) ; // reactance of individual phase conductor at 12 inch spacing in /mi
Z_aa = l * ( r_a + X_a ) ; // Z_aa = Z_bb = .... = Z_zz
Z_bb = Z_aa ;
Z_zz = Z_aa ;
Z_cc = Z_aa ;
D_eq1 = d * 2 ;
Z_ab = (l) * ( %i * 0.1213 * log(12/D_eq1) ) ;
```

```
Z_bc = Z_ab ;
Z_xy = Z_ab ; // Z_xy = Z_yx
Z_yz = Z_ab ;
Z_ba = Z_ab ;
Z_cb = Z_ab ;
D_eq2 = d * 3 ;
Z_bz = (l) * ( %i * 0.1213 * log(12/D_eq2) ) ;
Z_ay = Z_bz ; // Z_ya = Z_ay
Z_cx = Z_bz ; // Z_cx = Z_xc
Z_yz = Z_bz ; // Z_zy = Z_yz
D_eq3 = d * 4 ;
Z_ac = (l) * ( %i * 0.1213 * log(12/D_eq3) ) ;
Z_ca = Z_ac ; // Z_ac = Z_xz = Z_zx
D_eq4 = d * 1 ;
Z_ax = (l) * ( %i * 0.1213 * log(12/D_eq4) ) ;
Z_bx = Z_ax ; // Z_ax = Z_xa ; Z_bx = Z_xb
Z_by = Z_ax ; // Z_by = Z_yb
Z_cy = Z_ax ; // Z_cy = Z_yc
Z_cz = Z_ax ;
D_eq5 = d * 5 ;
Z_az = (l) * (%i*0.1213*log(12/D_eq5)) ; // Z_za=
    Z_az

Z_s = [Z_aa Z_ab Z_ac ; Z_ba Z_bb Z_bc ; Z_ca Z_cb
    Z_cc] ;
Z_tm = [Z_ax Z_bx Z_cx ; Z_ay Z_by Z_cy ; Z_az Z_bz
    Z_cz] ;
Z_M = [Z_ax Z_ay Z_az ; Z_bx Z_by Z_bz ; Z_cx Z_cy
    Z_cz] ;
Z_N = [Z_aa Z_xy Z_ac ; Z_xy Z_aa Z_ab ; Z_ac Z_ab
    Z_aa] ;
Z_new = (Z_s)-(Z_M)*(Z_N)^(-1)*(Z_tm) ;

// For case (b)
a = 1*exp(%i*120*%pi/180) ; // By symmetrical
    components theory to 3-    system
A = [1 1 1; 1 a^2 a ;1 a a^2] ;
Z_012 = inv(A) * Z_new * A ; // Sequence-impedance
```

```
matrix

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

Scilab code Exa 5.15 calculate PlOH PlGIL ElOH ElGIL ClOH ElavgGIL ClavgOH ClavgGIL Csavings breakeven period

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND GAS-INSULATED TRANSMISSION LINES

// EXAMPLE : 5.15 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
L = 50 ; // length of transmission line in km
P_l_oh = 820 ; // Power loss at peak load for overhead transmission line in kW/km
P_l_g = 254 ; // Power loss at peak load for gas insulated transmission line in kW/km
cost_kwh = 0.10 // cost of electric energy in $ per kWh
lf_ann = 0.7 ; // Annual load factor
plf_ann = 0.7 ; // Annual Power loss factor
```

```
h_yr = 365*24 ; // Time in Hours for a year
total_invest = 200000000 ; // Investment cost of GIL in $ ( for case (j) )

// CALCULATIONS
// For case (a)
Power_loss_OHline = P_l_oh * L ; // Power loss of overhead line at peak load in kW

// For case (b)
Power_loss_GILline = P_l_g * L ; // Power loss of gas-insulated transmission line at peak load in kW

// For case (c)
energy_loss_OH = Power_loss_OHline * h_yr ; // Total annual energy loss of OH line at peak load in kWh/yr

// For case (d)
energy_loss_GIL = Power_loss_GILline * h_yr ; // Total annual energy loss of GIL at peak load in kWh/yr

// For case (e)
energy_ann_OH = lf_ann * energy_loss_OH ; // Average energy loss of OH line at peak load in kWh/yr

// For case (f)
energy_ann_GIL = lf_ann * energy_loss_GIL ; // Average energy loss of GIL line at peak load in kWh/yr

// For case (g)
cost_ann_OH = cost_kwh * energy_ann_OH ; // Average annual cost of losses of OH line in $ per year

// For case (h)
```

```
cost_ann_GIL = cost_kwh * energy_ann_GIL ; // Average annual cost of losses of GIL line in $ per year

// For case (i)
P_loss_ann = cost_ann_OH - cost_ann_GIL ; // Annual resultant savings of losses per yr

// For case (j)
break_period = total_invest/P_loss_ann ; // Payback period if GIL alternative period is selected

// DISPLAY RESULTS
disp("EXAMPLE : 5.15 : SOLUTION :-") ;
printf("\n (a) Power loss of Overhead line at peak load , (Power loss)_OH_line = %d kW \n",Power_loss_OHline) ;
printf("\n (b) Power loss of Gas-insulated transmission line , (Power loss)_GIL_line = %d kW \n",Power_loss_GILline) ;
printf("\n (c) Total annual energy loss of Overhead transmission line at peak load = %.4e kWh/yr \n",energy_loss_OH) ;
printf("\n (d) Total annual energy loss of Gas-insulated transmission line at peak load = %.5e kWh/yr \n",energy_loss_GIL);
printf("\n (e) Average energy loss of Overhead transmission line = %.5e kWh/yr \n",energy_ann_OH);
printf("\n (f) Average energy loss of Gas-insulated transmission line at peak load = %.5e kWh/yr \n",energy_ann_GIL);
printf("\n (g) Average annual cost of losses of Overhead transmission line  = $ %.5e/yr \n",cost_ann_OH);
printf("\n (h) Average annual cost of losses of Gas-insulated transmission line = $ %.5e/yr \n",cost_ann_GIL);
```

```
printf("\n (i) Annual resultant savings in losses
    using Gas−insulated transmission line = $ %.6e/yr
    \n",P_loss_ann);
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 5 : UNDERGROUND POWER TRANSMISSION AND
   GAS−INSULATED TRANSMISSION LINES

// EXAMPLE : 5.16 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
n = 40 ; // useful life in years
i = 10/100 ; // carrying charge rate
A_P = (i*(1+i)^n)/((1 + i)^n - 1) ; // Refer page
    642
A_F = 0.00226 ; // A_F = A/F
pr_tax = 3/100 ; // Annual ad property taxes is 3%
    of 1st costs of each alternative

// FOR OVERHEAD TRANSMISSION
L_OH = 50 ; // length of route A in mi
cost_b_A = 1 * 10^6 ; // cost per mile to bulid in $
salvage_A = 2000 ; // salvage value per mile at end
```

```
of 40 years
cost_mait_OH = 500 ; // cost in $ per mile to maintain

// SUBMARINE TRANSMISSION LINE
L_S = 30 ; // length of route B in mi
cost_b_B = 4*10^6 ; // cost per mile to bulid in $
salvage_B = 6000 ; // salvage value per mile at end of 40 years
cost_mait_S = 1500 ; // cost in $ per mile to maintain

// GIL TRANSMISSION
L_GIL = 20 ; // length of route C in mi
cost_b_C = 7.6*10^6 ; // cost per mile to bulid in $
salvage_C = 1000 ; // salvage value per mile at end of 40 years
cost_mait_GIL = 200 ; // cost in $ per mile to maintain
savings = 17.5*10^6 ; // relative savings in power loss per year in $


// CALCULATIONS
n = 25 ; // useful life in years
i = 20/100 ; // carrying charge rate
p = ((1 + i)^n - 1)/(i*(1+i)^n) ; // p = P/A
// FOR OVERHEAD TRANSMISSION
P_OH = cost_b_A * L_OH ; // first cost of 500 kV OH line in $
F_OH = salvage_A * L_OH ; // Estimated salvage value in $
A_1 = P_OH * A_P - F_OH * A_F ; // Annual equivalent cost of capital in $
A_2 = P_OH * pr_tax + cost_mait_OH * L_OH ; // annual equivalent cost of tax and maintainance in $
A = A_1 + A_2 ; // total annual equi cost of OH line
```

```
in $

// SUBMARINE TRANSMISSION LINE
P_S = cost_b_B * L_S ; // first cost of 500 kV OH line in $
F_S = salvage_B * L_S ; // Estimated salvage value in $
B_1 = P_S * A_P - F_S * A_F ; // Annual equivalent cost of capital in $
B_2 = P_S * pr_tax + cost_mait_S * L_S ; // annual equivalent cost of tax and maintainance in $
B = B_1 + B_2 ; // total annual equi cost of OH line in $

// GIL TRANSMISSION
P_GIL = cost_b_C * L_GIL ; // first cost of 500 kV OH line in $
F_GIL = salvage_C * L_GIL ; // Estimated salvage value in $
C_1 = P_GIL * A_P - F_GIL * A_F ; // Annual equivalent cost of capital in $
C_2 = P_GIL * pr_tax + cost_mait_GIL * L_GIL ; // annual equivalent cost of tax and maintainance in $
C = C_1 + C_2 ; // total annual equi cost of OH line in $
A_net = C - savings ; // Total net annual equi cost of GIL

// DISPLAY RESULTS
disp("EXAMPLE : 5.16 : SOLUTION :-") ;
printf("\n OVERHEAD TRANSMISSION LINE : \n") ;
printf("\n  Annual equivalent cost of capital invested in line , A_1 = $ %d \n",A_1) ;
printf("\n  Annual equivalent cost of Tax and maintainance , A_2 = $ %d \n",A_2) ;
printf("\n  Total annual equivalent cost of OH transmission , A = $ %d \n",A) ;
```

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 6 : DIRECT CURRENT POWER TRANSMISSION

// EXAMPLE : 6.1 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
K_1 = 2.5 ; // Factor
K_2 = 1.7 ; // Factor

// CALCULATIONS
// For case (b)
I_d = poly(0,'I_d') ; // since P_loss(dc) = P_loss (
```

```
ac)
I_L = poly(0,'I_L') ; // i.e 2*I_d^2*R_dc = 3*I_L^2*R_ac
I_d = sqrt(3/2)*I_L ; // Ignoring skin effects R_dc = R_ac
I_d1 = 1.225*I_L ; // Refer Equ 6.23

// For case (a)
V_d = poly(0,'V_d') ; // Defining a ploynomial V_d
E_p = poly(0,'E_p') ; // since P_dc = P_ac (or) V_d*I_d = 3*E_p*I_L
V_d = 2.45*E_p ; // Refer Equ 6.25

// For case (c)
ins_lvl = (K_2*(V_d/2))/(K_1*E_p) ; // Ratio of dc insulation level to ac insulation level
ins_lvl_1 = (K_2*2.45/2)/K_1 ; // simplifying above equ
dc_i = poly(0,'dc_i') ; // dc_i = dc insulation level
ac_i = poly(0,'ac_i') ; // ac_i = ac insulation level
dc_i = ins_lvl_1 * ac_i ;

// DISPLAY RESULTS
disp("EXAMPLE : 6.1 : SOLUTION :-") ;
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) ;
printf("\n (b) The dc line current I_d in terms of ac line current I_L , I_d = \n"); disp(I_d1) ;
printf("\n (c) Ratio of dc insulation level to ac insulation level =  \n") ; disp(dc_i/ac_i) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 6 : DIRECT CURRENT POWER TRANSMISSION

// EXAMPLE : 6.2 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
K = 3 ; // factor

// CALCULATIONS
// For case (a)
V_d = poly(0,'V_d') ; // defining a polynomial
E_p = poly(0,'E_p') ;
V_d = K*2*E_p ; // From equ 6.18

// For case (b)
P_dc = poly(0,'P_dc') ;
P_ac = poly(0,'P_ac') ;
P_dc = 2*P_ac ;

// For case (c)
P_ld = poly(0,'P_ld') ; // P_loss(dc)
P_la = poly(0,'P_la') ; // P_loss(ac)
P_ld = (2/3)*P_la ;

```

```
// DISPLAY RESULTS
disp("EXAMPLE : 6.2 : SOLUTION :-") ;
printf("\n (a) Maximum operating V_d in terms of voltage E_p , V_d = \n") ; disp(V_d) ;
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) ;
printf("\n (or) P_dc = \n") ; disp(P_dc) ;
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) ;
printf("\n (or) P_loss(dc) = \n") ; disp(P_ld) ;
```

Scilab code Exa 6.3 calculate KVA rating Wye side KV rating

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION

// EXAMPLE : 6.3 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
V_d0 = 125 ; // voltage rating of bridge rectifier in kV
V_dr0 = V_d0 ; // Max continuos no-load direct voltage in kV
I = 1600 ; // current rating of bridge rectifier in A
I_d = I ; // Max continuous current in A
```

```
// CALCULATIONS
// For case (a)
S_B = 1.047 * V_d0 * I_d ; // 3-phase kVA rating of
    rectifier transformer

// For case (b)
// SINCE   V_d0 = 2.34*E_LN
E_LN = V_d0/2.34 ; // Wye side kV rating

// DISPLAY RESULTS
disp("EXAMPLE : 6.3 : SOLUTION :-") ;
printf("\n (a) Three-phase kilovolt-ampere rating ,
    S_B = %d kVA \n",S_B) ;
printf("\n (b) Wye-side kilovolt rating , E_L-N = %
    .4f kV \n",E_LN) ;
```

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION

// EXAMPLE : 6.4 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
E_LN = 53.418803 ; // Wye-side kV rating . From exa
    6.3
```

```
I = 1600 ; // current rating of bridge rectifier in A
I_d = I ; // Max continuous current in A
X_tr = 0.10 ; // impedance of rectifier transformer in pu

// For case (a)
sc_MVA1 = 4000 ; // short-ckt MVA

// For case (b)
sc_MVA2 = 2500 ; // short-ckt MVA

// For case (c)
sc_MVA3 = 1000 ; // short-ckt MVA

// CALCULATIONS
nom_kV = sqrt(3) * E_LN ; // Nominal kV_L-L
I_1ph = sqrt(2/3) * I_d ; // rms value of wye-side phase current
E_LN1 = E_LN * 10^3 ; // Wye-side rating in kV
X_B = (E_LN1/I_1ph) ; // Associated reactance base in

// For case (a)
X_sys1 = nom_kV^2/sc_MVA1 ; // system reactance in

X_tra = X_tr * X_B ; // Reactance of rectifier transformer
X_C = X_sys1 + X_tra ; // Commutating reactance in


// For case (b)
X_sys2 = nom_kV^2/sc_MVA2 ; // system reactance in

X_C2 = X_sys2 + X_tra ; // Commutating reactance in


// For case (b) When breaker 1 & 2 are open
```

```
X_sys3 = nom_kV^2/sc_MVA3 ; // system reactance in

X_C3 = X_sys3 + X_tra ; // Commutating reactance in


// DISPLAY RESULTS
disp("EXAMPLE : 6.4 : SOLUTION :-") ;
printf("\n (a) Commutating reactance When all three breakers are closed, X_C = %.4f \n",X_C) ;
printf("\n (b) Commutating reactance When breaker 1 is open, X_C = %.4f \n",X_C2) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION

// EXAMPLE : 6.5 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
X_C = 6.2292017 ; // commutating reactance when all 3 breakers are closed
E_LN = 53.418803 * 10^3 ; // Wye-side volt rating
V_d0 = 125 * 10^3 ; // voltage rating of bridge rectifier in V
V_dr0 = V_d0 ; // Max continuos no-load direct
```

```
voltage in V
I = 1600 ; // current rating of bridge rectifier in A
I_d = I ; // Max continuous current
nom_kV = sqrt(3) * E_LN ; // Nominal kV_L-L
X_tr = 0.10 ; //impedance of rectifier transformer in pu
alpha = 0 ; // delay angle    = 0 degree

// CALCULATIONS
// For case (a)
E_m = sqrt(2) * E_LN ;
u = acosd(1 - (2*X_C*I_d)/(sqrt(3)*E_m)); // overlap angle when delay angle    = 0 degree

// For case (b)
R_C = (3/%pi) * X_C ; // Equ commutation resistance per phase
V_d = V_d0 * cosd(alpha) - R_C * I_d ; // dc voltage of rectifier in V

// For case (c)
cos_theta = V_d/V_d0 ; // Displacement or power factor of rectifier

// For case (d)
Q_r = V_d * I_d * tand( acosd(cos_theta) ) ; // magnetizing var I/P

// DISPLAY RESULTS
disp("EXAMPLE : 6.5 : SOLUTION :-") ;
printf("\n (a) Overlap angle u of rectifier , u = %.2f degree\n",u) ;
printf("\n (b) The dc voltage V_dr of rectifier , V_dr = %.2f V \n",V_d) ;
printf("\n (c) Displacement factor of rectifier , cos   = %.3f  \n",cos_theta) ;
printf("\n     and      = %.1f degree \n ",acosd(
```

```
    cos_theta)) ;
printf("\n (d) Magnetizing var input to rectifier ,
    Q_r = %.4e var \n",Q_r) ;

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION

// EXAMPLE : 6.6 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
I_d = 1600 ; // Max continuous dc current in A
V_d0 = 125 * 10^3 ; // voltage rating of bridge
    rectifier in V
V_d = 100 * 10^3 ; // dc voltage of rectifier in V
X_C = 6.2292017 ; // commutating reactance when all
    3 breakers are closed

// CALCULATIONS
// For case (a)
R_C = (3/%pi) * X_C ;
cos_alpha = (V_d + R_C*I_d)/V_d0 ; // Firing angle

alpha = acosd(cos_alpha) ;
```

```

// For case (b)
// V_d = (1/2)*V_d0*(cos_alpha + cos_delta)
cos_delta = (2 * V_d/V_d0) - cos_alpha ;
delta = acosd(cos_delta) ;
u = delta - alpha ; // Overlap angle u in degree

// For case (c)
cos_theta = V_d/V_d0 ; // power factor
theta = acosd(cos_theta) ;

// For case (d)
Q_r = V_d * I_d * tand(theta) ; // magnetizing var I/P

// DISPLAY RESULTS
disp("EXAMPLE : 6.6 : SOLUTION :-") ;
printf("\n (a) Firing angle  of rectifier,  = %.2f degree\n",alpha) ;
printf("\n (b) Overlap angle u of rectifier, u = %.2f degree\n",u) ;
printf("\n (c) Power factor, cos  = %.2f \n",cos_theta) ;
printf("\n   and   = %.2f degree \n ",theta) ;
printf("\n (d) Magnetizing var input, Q_r = %.2e var \n",Q_r) ;
```

Scilab code Exa 6.7 determine u mode Id or Vdr

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

```

```
// CHAPTER : 6 : DIRECT-CURRENT POWER TRANSMISSION

// EXAMPLE : 6.7 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
X_C = 12.649731 ; // commutating reactance when 2 breakers are open
alpha = 0 ;
I_d = 1600 ; // DC current in A
E_LN = 53.4188 * 10^3 ; // Wye-side rating in V
V_d0 = 125 * 10^3 ; // voltage rating of bridge rectifier in V

// CALCULATIONS
// For case (a)
E_m = sqrt(2) * E_LN ;
u = acosd(1 - (2 * X_C * I_d)/(sqrt(3) * E_m)) ; // overlap angle u =

// For case (b)
// since rectifier operates in first mode i.e doesn't operate in second mode
R_C = (3/%pi) * X_C ;
V_dr = ( V_d0 * cosd(alpha) ) - (R_C*I_d) ; // dc voltage of rectifier in V

// DISPLAY RESULTS
disp("EXAMPLE : 6.7 : SOLUTION :-") ;
printf("\n (a) u = %.1f degree \n",u) ;
printf("\n since u < 60 degree . The rectifier operates at FIRST mode , the normal operating mode \n") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 6 : DIRECT—CURRENT POWER TRANSMISSION

// EXAMPLE : 6.10 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
X_C = 6.2292 ; // commutating reactance when all 3
   breakers are closed
I_db = 1600 ; // dc current base in A
V_db = 125 * 10^3 ; // dc voltage base in V
I_d = I_db ; // Max continuous current in A
V_d = 100 * 10^3 ; // dc voltage in V
alpha = 0 ; // Firing angle   = 0 degree

// CALCULATIONS
// For case (a)
R_c = (3/%pi) * X_C ;
R_cb = V_db/I_db ; // Resistance base in
V_d_pu = V_d/V_db ; // per unit voltage
I_d_pu = I_d/I_db ; // per unit current
R_c_pu = R_c/R_cb ; // per unit
E_pu = (V_d_pu + R_c_pu * I_d_pu)/cosd(alpha) ; //
   Open ckt dc voltage in pu
V_d0 = E_pu * V_db ; // Open ckt dc voltage in V

```

```
// For case (b)
E = V_d0/2.34; // Open ckt ac voltage on wye side of transformer in V

// For case (c)
E_1LN = 92.95 * 10^3 ; // voltage in V
E_1B = E_1LN ;
E_LN = 53.44 * 10^3 ; // voltage in V
a = E_1LN/E_LN ;
n = a ; // when LTC on neutral
X_c_pu = 2 * R_c_pu ;
E_1_pu = E_1LN / E_1B ; // per unit voltage
cos_delta = cosd(alpha) - ( (X_c_pu * I_d_pu)/( (a/n) *E_1_pu) ) ;
delta = acosd(cos_delta) ;
u = delta - alpha ;

// For case (d)
cos_theta = V_d/V_d0 ; // pf of rectifier
theta = acosd(cos_theta) ;

// For case (e)
Q_r = V_d*I_d*tand(theta) ; // magnetizing var I/P

// For case (f)
d_V = E_LN - E ; // necessary change in voltage in V
p_E_LN = 0.00625 * E_LN ; // one buck step can change in V/step
no_buck = d_V / p_E_LN ; // No. of steps of buck

// DISPLAY RESULTS
disp("EXAMPLE : 6.10 : SOLUTION :-") ;
printf("\n (a) Open circuit dc Voltage , V_d0 = %.2f V \n",V_d0);
printf("\n (b) Open circuit ac voltage on wye side of transformer , E = %.2f V \n",E);
printf("\n (c) Overlap angle , u = %.2f degree \n",u)
```

```
printf("\n (d) Power factor , cos   = %.3f  \n",
   cos_theta);
printf("\n     and       = %.2f degree \n ",theta);
printf("\n (e) Magnetizing var input to rectifier ,
   Q_r = %.4e var \n",Q_r);
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND
   INSULATION COORDINATION

// EXAMPLE : 7.1 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
V = 1000 ; // surge voltage in kV
```

```
Z_c = 500 ; // surge impedance in

// CALCULATIONS
// For case (a)
P = V^2/Z_c ; // Total surge power in MW

// For case (b)
V1 = V*10^3 ; // surge voltage in V
i = V1/Z_c ;// surge current in A

// DISPLAY RESULTS
disp("EXAMPLE : 7.1 : SOLUTION :-") ;
printf("\n (a) Total surge power in line , P = %d MW \n",P) ;
printf("\n (b) Surge current in line , i = %d A \n",i) ;
```

Scilab code Exa 7.2 determine surge Power surge current

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND INSULATION COORDINATION

// EXAMPLE : 7.2 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
V = 1000 ; // surge voltage in kV
Z_c = 50 ; // surge impedance in
```

```
// CALCULATIONS
// For case (a)
P = V^2/Z_c ; // Total surge power in MW

// For case (b)
V1 = V*10^3 ; // surge voltage in V
i = V1/Z_c ;// surge current in A

// DISPLAY RESULTS
disp("EXAMPLE : 7.1 : SOLUTION :-") ;
printf("\n (a) Total surge power in line , P = %d MW
    \n",P) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND
   INSULATION COORDINATION

// EXAMPLE : 7.4 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
R = 500 ; // Resistance in
Z_c = 400 ; // characteristic impedance in
v_f = 5000 ; // Forward travelling voltage wave in V
```

```
i_f = 12.5 ; // Forward travelling current wave in A

// CALCULATIONS
// For case (a)
r_v = (R - Z_c)/(R + Z_c) ; // Reflection coefficient of voltage wave

// For case (b)
r_i = -(R - Z_c)/(R + Z_c) ; // Reflection coefficient of current wave

// For case (c)
v_b = r_v * v_f ; // Backward-travelling voltage wave in V

// For case (d)
v = v_f + v_b ; // Voltage at end of line in V
v1 = (2 * R/(R + Z_c)) * v_f ; // (or) Voltage at end of line in V

// For case (e)
t1 = (2 * R/(R + Z_c)) ; // Refraction coefficient of voltage wave

// For case (f)
i_b = -( v_b/Z_c ) ; // backward-travelling current wave in A
i_b1 = -r_v * i_f ; // (or) backward-travelling current wave in A


// For case (g)
i = v/R ; // Current flowing through resistor in A

// For case (h)
t2 = (2 * Z_c/(R + Z_c)) ; // Refraction coefficient of current wave

```

```
// DISPLAY RESULTS
disp("EXAMPLE : 7.4 : SOLUTION :-") ;
printf("\n (a) Reflection coefficient of voltage wave ,    = %.4f \n",r_v) ;
printf("\n (b) Reflection coefficient of current wave ,    = %.4f \n",r_i) ;
printf("\n (c) Backward-travelling voltage wave , v_b = %.3f V \n",v_b) ;
printf("\n (d) Voltage at end of line , v = %.3f V \n",v) ;
printf("\n     From alternative method ")
printf("\n     Voltage at end of line , v = %.3f V \n",v) ;
printf("\n (e) Refraction coefficient of voltage wave ,    = %.4f \n",t1) ;
printf("\n (f) Backward-travelling current wave , i_b = %.4f A \n",i_b) ;
printf("\n (g) Current flowing through resistor , i = %.4f A \n",i) ;
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 surges

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND INSULATION COORDINATION

// EXAMPLE : 7.5 :
```

```
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
Z_c1 = 400 ; // Surge impedance of line in
Z_c2 = 40 ; // Surge impedance of cable in
v_f = 200 ; // Forward travelling surge voltage in kV

// CALCULATIONS
// For case (a)
v_f1 = v_f * 10^3 ; // surge voltage in V
i_f = v_f1/Z_c1 ; // Magnitude of forward current wave in A

// For case (b)
r = (Z_c2 - Z_c1)/(Z_c2 + Z_c1) ; // Reflection coefficient

// For case (c)
t = 2 * Z_c2/(Z_c2 + Z_c1) ; // Refraction coefficient

// For case (d)
v = t * v_f ; // Surge voltage transmitted forward into cable in kV

// For case (e)
v1 = v * 10^3 ; // Surge voltage transmitted forward into cable in V
I = v1/Z_c2 ; // Surge current transmitted forward into cable in A

// For case (f)
v_b = r * v_f ; // surge voltage reflected back along overhead line in kV

// For case (g)
```

```
i_b = -r * i_f ; // surge current reflected back along overhead line in A

// For case (h)
// Arbitrary values are taken in graph.Only for reference not for scale
T = 0:0.1:300 ;

for i = 1:int(length(T)/3) ; // plotting Voltage values
    vo(i) = 3;
end
for i = int(length(T)/3):length(T)
    vo(i) = 1 ;
end
for i = int(length(T))
    vo(i) = 0 ;
end


a=gca() ;
ylabel("CURRENT                    SENDING END
                    VOLTAGE                ") ;
b = newaxes() ; // creates new axis
b.y_location = "right" ; // Position of axis
ylabel ("RECEIVING END") ; // Labelling y-axis
b.axes_visible = ["off","off","off"] ;
e = newaxes() ;
e.y_location = "middle" ;
e.y_label.text = "JUNCTION" ;
subplot(2,1,1) ;
plot2d(T,vo,2,'012','',[0,0,310,6]) ;

for i = 1:int(length(T)/3) ; // Plotting current surges value
    io(i) = 1 ;
end
for i = int(length(T)/3):length(T)
```

```
    io(i) = 3 ;
end
for i = int(length(T))
    io(i) = 0 ;
end


c=gca() ;
d = newaxes() ;
d.y_location = "right" ;
d.filled = "off" ;
f.y_location = "middle" ;
f.y_label.text = "JUNCTION" ;
subplot(2,1,2) ;
plot2d(T,io,5,'012','',[0,0,310,6]) ;

// DISPLAY RESULTS
disp("EXAMPLE : 7.5 : SOLUTION :-") ;
printf("\n (a) Magnitude of forward current wave , i_f = %d A \n",i_f) ;
printf("\n (b) Reflection coefficient ,  = %.4f \n",r) ;
printf("\n (c) Refraction coefficient ,  = %.4f \n",t) ;
printf("\n (d) Surge voltage transmitted forward into cable , v = %.2f kV \n",v) ;
printf("\n (e) Surge current transmitted forward into cable , i = %.f A \n",I) ;
printf("\n (f) Surge voltage reflected back along the OH line , v_b = %.2f kV \n",v_b) ;
printf("\n (g) Surge current reflected back along the OH line , i_b = %.f A \n",i_b) ;
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 end voltage with time

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 7 : TRANSIENT OVERVOLTAGES AND INSULATION COORDINATION

// EXAMPLE : 7.6 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
v = 1000 ; // ideal dc voltage source in V
Z_s = 0 ; // internal impedance in
Z_c = 40 ; // characteristic impedance in
Z_r = 60 ; // Cable is terminated in 60   resistor

// CALCULATIONS
// For case (a)
r_s = (Z_s - Z_c)/(Z_s + Z_c) ; // Reflection coefficient at sending end

// For case (b)
r_r = (Z_r - Z_c)/(Z_r + Z_c) ; // Reflection coefficient at receiving end

// For case (c)
T = 0:0.001:10.6 ; // // plotting values
for i = 1:length(T) ;
    if(T(i)<=1)
        x(i) = (1.2)*T(i) - 1 ;
    elseif(T(i)>=1 & T(i)<=2)
        x(i) = (-1.2)*T(i) + 1.4 ;
```

```
    elseif(T(i)>=2 & T(i)<=3)
        x(i) = (1.2)*T(i)- 3.4 ;
    elseif(T(i)>=3 & T(i)<=4)
        x(i) = (-1.2)*T(i) + 3.8 ;
    elseif(T(i)>=4 & T(i)<=5)
        x(i) = (1.2)*T(i)- 5.8 ;
    elseif(T(i)>=5 & T(i)<=6)
        x(i) = (-1.2)*T(i) + 6.2 ;
    elseif(T(i)>=6 & T(i)<=7)
        x(i) = (1.2)*T(i)- 8.2 ;
    elseif(T(i)>=7 & T(i)<=8)
        x(i) = (-1.2)*T(i) + 8.6 ;
    elseif(T(i)>=8 & T(i)<=9)
        x(i) = (1.2)*T(i)- 10.6 ;
    elseif(T(i)>=9 & T(i)<=10)
        x(i) = (-1.2)*T(i) + 11 ;
    elseif(T(i)>=10 & T(i)<=10.6)
        x(i) = (1.2)*T(i) - 13 ;
        end
end

subplot(2,1,1) ; // Plotting two graph in same window
plot2d(T,x,5,'012','',[0,-1,11,0.2]) ;

a = gca() ;
xlabel("TIME") ;
ylabel("  _s = −1              DISTANCE
   _r = 0.2") ;
xtitle("Fig 7.6 (c) Lattice diagram") ;
a.thickness = 2 ; // sets thickness of plot
xset('thickness',2) ; // sets thickness of axes
xstring(1,-1,'T') ;
xstring(2,-1,'2T') ;
xstring(3,-1,'3T') ;
xstring(4,-1,'4T') ;
xstring(5,-1,'5T') ;
xstring(6,-1,'6T') ;
```

```
xstring(7,-1,'7T') ;
xstring(8,-1,'8T') ;
xstring(9,-1,'9T') ;
xstring(10,-1,'10T') ;
xstring(0.1,0.1,'0V') ;
xstring(2,0.1,'1200V') ;
xstring(4,0.1,'960V') ;
xstring(6,0.1,'1008V') ;
xstring(8,0.1,'998.4V') ;
xstring(1,-0.88,'1000V') ;
xstring(3,-0.88,'1000V') ;
xstring(5,-0.88,'1000V') ;
xstring(7,-0.88,'1000V') ;
xstring(9,-0.88,'1000V') ;

// For case (d)
q1 = v ; // Refer Fig 7.11 in textbook
q2 = r_r * v ;
q3 = r_s * r_r * v ;
q4 = r_s * r_r^2 * v ;
q5 = r_s^2 * r_r^2 * v ;
q6 = r_s^2 * r_r^3 * v ;
q7 = r_s^3 * r_r^3 * v ;
q8 = r_s^3 * r_r^4 * v ;
q9 = r_s^4 * r_r^4 * v ;
q10 = r_s^4 * r_r^5 * v ;
q11 = r_s^5 * r_r^5 * v ;
V_1 = v - q1 ;
V_2 = v - q3 ;
V_3 = v - q5 ;
V_4 = v - q7 ; // voltage at t = 6.5T & x = 0.25l in
    Volts
V_5 = v - q9 ;

// For case (e)
t = 0:0.001:9 ;

for i= 1:length(t)
```

```
    if(t(i)>=0 & t(i)<=1)
        y(i) = V_1 ;
    elseif(t(i)>=1 & t(i)<=3)
        y(i) = V_2 ;
    elseif(t(i)>=3 & t(i)<=5)
        y(i)= V_3 ;
    elseif(t(i)>=5 & t(i)<=7)
        y(i)= V_4 ;
    elseif(t(i)>=7 & t(i)<=9)
        y(i)= V_5 ;
    end
end
subplot(2,1,2) ;
a = gca() ;
a.thickness = 2 ; // sets thickness of plot
plot2d(t,y,2,'012','',[0,0,10,1300]) ;
a.x_label.text = 'TIME (T)' ; // labels x-axis
a.y_label.text = 'RECEIVING-END  VOLTAGE (V)' ; // labels y-axis
xtitle("Fig 7.6 (e) . Plot of Receiving end Voltage v/s Time") ;
xset('thickness',2); // sets thickness of axes
xstring(1,0,'1T') ; // naming points
xstring(3,0,'3T') ;
xstring(5,0,'5T') ;
xstring(7,0,'7T') ;
xstring(1,1200,'1200 V') ;
xstring(4,960,'960 V') ;
xstring(6,1008,'1008 V') ;
xstring(8,998.4,'998.4 V') ;


// DISPLAY RESULTS
disp("EXAMPLE : 7.6 : SOLUTION :-") ;
printf("\n (a) Reflection coefficient at sending end ,  _s = %.f \n",r_s) ;
printf("\n (b) Reflection coefficient at sending end ,  _r = %.1f \n",r_r)
```

```
printf("\n (c) The lattice diagram is shown in Fig
    7.6 (c) \n") ;
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) ;
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 Vv

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 8 : LIMITING FACTORS FOR EXTRA-HIGH AND
    ULTRAHIGH VOLTAGE TRANSMISSION

// EXAMPLE : 8.1 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
```

```
m_0 = 0.90 ; // Irregularity factor
p = 74 ; // Atmospheric pressure in Hg
t = 10 ; // temperature in degree celsius
D = 550 ; // Equilateral spacing b/w conductors in
    cm
d = 3 ; // overall diameter in cm

// CALCULATIONS
// For case (a)
r = d/2 ;
delta = 3.9211 * p/( 273 + t ) ; // air density
    factor
V_0_ph = 21.1 * delta * m_0 * r * log(D/r) ; //
    disruptive critical rms line voltage in kV/phase
V_0 = sqrt(3) * V_0_ph ; // disruptive critical rms
    line voltage in kV

// For case (b)
m_v = m_0 ;
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
V_v = sqrt(3)*V_v_ph ; // visual critical rms line
    voltage in kV

// DISPLAY RESULTS
disp("EXAMPLE : 8.1 : SOLUTION :-") ;
printf("\n (a) Disruptive critical rms line voltage
    , V_0 = %.1f kV \n",V_0) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
```

```
ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 8 : LIMITING FACTORS FOR EXTRA−HIGH AND ULTRAHIGH VOLTAGE TRANSMISSION

// EXAMPLE : 8.2 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
f = 60 ; // freq in Hz
d = 3 ; // overall diameter in cm
D = 550 ; // Equilateral spacing b/w conductors in cm
V1 = 345 ; // operating line voltage in kV
V_0 = 172.4 ; // disruptive critical voltage in kV
L = 50 ; // line length in mi
p = 74 ; // Atmospheric pressure in Hg
t = 10 ; // temperature in degree celsius
m_0 = 0.90 ; // Irregularity factor

// CALCULATIONS
r = d/2 ;
delta = 3.9211 * p/( 273 + t ) ; // air density factor
V_0 = 21.1 * delta * m_0 * r * log(D/r) ; // disruptive critical rms line voltage in kV/phase
V =V1/sqrt(3) ; // Line to neutral operating voltage in kV
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
P_cT = P_c * L ; // For total line length corona loss in kW/phase
T_P_c = 3 * P_cT ; // Total corona loss of line in
```

```
kW

// DISPLAY RESULTS
disp("EXAMPLE : 8.2 : SOLUTION :-") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
   ANALYSIS

// EXAMPLE : 9.1 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
V_a = 7.3 * exp(%i*12.5*%pi/180) ; // Phase voltage
   in V
```

```
V_b = 0.4 * exp(%i*(-100)*%pi/180) ; // Phase voltage in V
V_c = 4.4 * exp(%i*154*%pi/180) ; // Phase voltage in V
a = 1 * exp(%i*120*%pi/180) ; // operator 'a' by application of symmetrical components theory to 3−   system . Refer section 9.3 for details

// CALCULATIONS
V_a0 = (1/3) * (V_a + V_b + V_c) ; // Analysis equ in V
V_a1 = (1/3) * (V_a + a*V_b + a^2*V_c) ;
V_a2 = (1/3) * (V_a + a^2*V_b + a*V_c) ;
V_b0 = V_a0 ;
V_b1 = a^2 * V_a1 ;
V_b2 = a * V_a2 ;
V_c0 = V_a0 ;
V_c1 = a * V_a1 ;
V_c2 = a^2 * V_a2 ;

// DISPLAY RESULTS
disp("EXAMPLE : 9.1 : SOLUTION :-") ;
printf("\n The symmetrical components for the phase voltages V_a , V_b & V_c are\n") ;
printf("\n V_a0 = %.2f<%.1f V \n",abs(V_a0),atand(imag(V_a0),real(V_a0) )) ;
printf("\n V_a1 = %.2f<%.1f V \n",abs(V_a1),atand(imag(V_a1),real(V_a1) )) ;
printf("\n V_a2 = %.2f<%.1f V \n",abs(V_a2),atand(imag(V_a2),real(V_a2) )) ;
printf("\n V_b0 = %.2f<%.1f V \n",abs(V_b0),atand(imag(V_b0),real(V_b0) )) ;
printf("\n V_b1 = %.2f<%.1f V \n",abs(V_b1),atand(imag(V_b1),real(V_b1) )) ;
printf("\n V_b2 = %.2f<%.1f V \n",abs(V_b2),atand(imag(V_b2),real(V_b2) )) ;
printf("\n V_c0 = %.2f<%.1f V \n",abs(V_c0),atand(imag(V_c0),real(V_c0) )) ;
```

```
printf("\n V_c1 = %.2f<%.1f V \n",abs(V_c1),atand(imag(V_c1),real(V_c1) )) ;
printf("\n V_c2 = %.2f<%.1f V \n",abs(V_c2),atand(imag(V_c2),real(V_c2) )) ;

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") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.2 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
V_abc = [0 ; 50 ; -50] ; // Phase voltages of a 3-system in V
I_abc = [-5 ; 5*%i ; -5] ; // Phase current of a 3-system in A

// CALCULATIONS
```

```
// For case (a)
S_3ph = (V_abc)' * conj(I_abc) ; // 3-  complex power in VA

// For case (b)
a = 1*exp(%i*120*%pi/180) ; // By symmetrical components theory to 3-  system
A = [1 1 1; 1 a^2 a ;1 a a^2] ;
V_012 = inv(A) * (V_abc) ; // Sequence voltage matrices in V
I_012 = inv(A) * (I_abc) ; // Sequence current matrices in A

// For case (c)
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)

// DISPLAY RESULTS
disp("EXAMPLE : 9.2 : SOLUTION :-") ;
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) )) ;
printf("\n (b) Sequence Voltage matrices , [V_012] = V \n") ;
printf("\n     %.f<%.f ",abs(V_012(1,1)),atand( imag(V_012(1,1)),real(V_012(1,1)) )) ;
printf("\n     %.4f<%.f ",abs(V_012(2,1)),atand(imag(V_012(2,1)),real(V_012(2,1)) )) ;
printf("\n     %.4f<%.f ",abs(V_012(3,1)),atand(imag(V_012(3,1)),real(V_012(3,1)) )) ;
printf("\n \n    Sequence current matrices , [I_012] = A \n") ;
printf("\n     %.4f<%.1f ",abs(I_012(1,1)),atand(imag(I_012(1,1)),real(I_012(1,1)) )) ;
printf("\n     %.4f<%.f ",abs(I_012(2,1)),atand(imag(I_012(2,1)),real(I_012(2,1)) )) ;
printf("\n     %.4f<%.f ",abs(I_012(3,1)),atand(
```

```
    imag(I_012(3,1)),real(I_012(3,1)) )) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
    ANALYSIS

// EXAMPLE : 9.3 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
l = 40 ; // line length in miles
// Conductor parameter from Table A.3
r_a = 0.206 ; // Ohms per conductor per mile in   /
    mi
r_b = r_a ; // r_a = r_b = r_c in   /mi
D_s = 0.0311 ; // GMR in ft where D_s = D_sa = D_sb
    = D_sc
D_ab = sqrt(2^2 + 8^2) ; // GMR in ft
D_bc = sqrt(3^2 + 13^2) ; // GMR in ft
D_ac = sqrt(5^2 + 11^2) ; // GMR in ft
D_e = 2788.5 ; // GMR in ft since earth resistivity
    is zero
r_e = 0.09528 ; // At 60 Hz in   /mi
```

```
// CALCULATIONS
// For case (a)
Z_aa =[(r_a + r_e) + %i * 0.1213*log(D_e/D_s)]*l ;
    // Self impedance of line conductor in
Z_bb = Z_aa ;
Z_cc = Z_bb ;
Z_ab = [r_e + %i * 0.1213*log(D_e/D_ab)]*l ; //
    Mutual impedance in
Z_ba = Z_ab ;
Z_bc = [r_e + %i * 0.1213*log(D_e/D_bc)]*l ;
Z_cb = Z_bc ;
Z_ac = [r_e + %i * 0.1213*log(D_e/D_ac)]*l ;
Z_ca = Z_ac ;
Z_abc = [Z_aa Z_ab Z_ac ; Z_ba Z_bb Z_bc ; Z_ca Z_cb
     Z_cc] ; // Line impedance matrix

// For case (b)
a = 1*exp(%i*120*%pi/180) ; // By symmetrical
    components theory to 3-   system
A = [1 1 1; 1 a^2 a ;1 a a^2] ;
Z_012 = inv(A) * Z_abc*A ; // Sequence impedance
    matrix

// DISPLAY RESULTS
disp("EXAMPLE : 9.3 : SOLUTION :-") ;
printf("\n (a) Line impedance matrix , [Z_abc] = \n"
    ) ; disp(Z_abc) ;
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 transposed line

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.4 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
l = 40 ; // line length in miles
// Conductor parameter from Table A.3
r_a = 0.206 ; // Ohms per conductor per mile in  /mi
r_b = r_a ; // r_a = r_b = r_c in  /mi
D_s = 0.0311 ; // GMR in ft where D_s = D_sa = D_sb = D_sc
D_ab = sqrt(2^2 + 8^2) ; // GMR in ft
D_bc = sqrt(3^2 + 13^2) ; // GMR in ft
D_ac = sqrt(5^2 + 11^2) ; // GMR in ft
D_e = 2788.5 ; // GMR in ft since earth resistivity is zero
r_e = 0.09528 ; // At 60 Hz in  /mi

// CALCULATIONS
// For case (a)
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
D_eq = (D_ab * D_bc * D_ac)^(1/3) ; // Equ GMR
Z_m = [r_e + %i*0.1213*log(D_e/D_eq)]*l ; // From equ 9.50
Z_abc = [Z_s Z_m Z_m ; Z_m Z_s Z_m ; Z_m Z_m Z_s] ; // Line impedance matrix
```

```
// For case (b)
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

// DISPLAY RESULTS
disp("EXAMPLE : 9.4 : SOLUTION :-") ;
printf("\n (a) Line impedance matrix when line is completely transposed , [Z_abc] = \n") ; disp(Z_abc) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.5 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
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
Y_012 = inv(Z_012) ; // Sequence admittance of line

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

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

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

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

```
// DISPLAY RESULTS
disp("EXAMPLE : 9.5 : SOLUTION :-") ;
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) )) ;
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) )) ;
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) )) ;
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

```

// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.6 :
clear ; clc ; close ; // Clear the work space and console
```

```
// GIVEN DATA
kv = 115 ; // Line voltage in kV

// For case (a)
h_11 = 90 ; // GMD b/w ground wires & their images
r_a = 0.037667 ; // Radius in metre
p_aa = 11.185 * log(h_11/r_a) ; // unit is F^(-1)m
p_bb = p_aa ;
p_cc = p_aa ;
l_12 = sqrt(22 + (45 + 37)^2) ;
D_12 = sqrt(2^2 + 8^2) ; // GMR in ft
p_ab = 11.185*log(l_12/D_12) ; // unit is F^(-1)m
p_ba = p_ab ;
D_13 = sqrt(3^2 + 13^2) ; // GMR in ft
l_13 = 94.08721051 ;
p_ac = 11.185 * log(l_13/D_13) ; // unit is F^(-1)m
p_ca = p_ac ;
l_23 = 70.72279912 ;
D_23 = sqrt(5^2 + 11^2) ; // GMR in ft
p_bc = 11.185 * log(l_23/D_23) ; // unit is F^(-1)m
p_cb = p_bc ;
P_abc = [p_aa p_ab p_ac ; p_ba p_bb p_bc ; p_ca p_cb
    p_cc] ; // Matrix of potential coefficients

// For case (b)
C_abc = inv(P_abc) ; // Matrix of maxwells
    coefficients

// For case (c)
a = 1*exp(%i*120*%pi/180) ; // By symmetrical
    components theory to 3-    system
A = [1 1 1; 1 a^2 a ;1 a a^2] ;
C_012 = inv(A) * C_abc * A ; // Matrix of sequence
    capacitances

// For case (d)
C_01 = C_012(1,2) ;
```

```
C_11 = C_012(2,2) ;
C_21 = C_012(3,2) ;
d_0 = C_01/C_11 ; // Zero-sequence electrostatic unbalances . Refer equ 9.115
d_2 = -C_21/C_11 ; // Negative-sequence electrostatic unbalances . Refer equ 9.116

// DISPLAY RESULTS
disp("EXAMPLE : 9.6 : SOLUTION :-") ;
printf("\n (a) Matrix of potential coefficients , [P_abc] = \n") ; disp(P_abc) ;
printf("\n (b) Matrix of maxwells coefficients , [C_abc] = \n") ; disp(C_abc) ;
printf("\n (c) Matrix of sequence capacitances , [C_012] = \n") ; disp(C_012) ;
printf("\n (d) Zero-sequence electrostatic unbalances , d_0 = %.4f<%.1f \n",abs(d_0),atand(imag(d_0),real(d_0) )) ;
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 Line-toLineVoltages at Faultpoints

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.9 :
```

```
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
kv = 230 ; // Line voltage in kV
Z_0 = 0.56 * %i ; // impedance in
Z_1 = 0.2618 * %i ; // Impedance in
Z_2 = 0.3619 * %i ; // Impedance in
z_f = 5 + 0*%i ; // fault impedance in
v = 1 * exp(%i*0*%pi/180) ;

// CALCULATIONS
// For case (a)
Z_B = kv^2/200 ; // Imedance base on 230 kV line
Z_f = z_f/Z_B ; // fault impedance in pu
I_a0 = v/(Z_0 + Z_1 + Z_2 + 3*Z_f) ; // Sequence currents in pu A
I_a1 = I_a0 ;
I_a2 = I_a0 ;
a = 1 * exp(%i*120*%pi/180) ; // By symmetrical components theory to 3-  system
A = [1 1 1; 1 a^2 a ;1 a a^2] ;
I_f = A * [I_a0 ; I_a1 ; I_a2] ; // Phase currents in pu A

// For case (b)
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
V_f = A*V_a ; // Phase voltage in pu V

// For case (c)
V_abf = V_f(1,1) - V_f(2,1) ; // Line-to-line voltages at fault points in pu V
V_bcf = V_f(2,1) - V_f(3,1) ; // Line-to-line voltages at fault points in pu V
V_caf = V_f(3,1) - V_f(1,1) ; // Line-to-line voltages at fault points in pu V

```

```
// DISPLAY RESULTS
disp("EXAMPLE : 9.9 : SOLUTION :-") ;
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) )) ;
printf("\n  Phase currents in pu A , [I_af ; I_bf ; I_cf] = pu A \n") ;
printf("\n    %.4f<%.1f ",abs(I_f),atand(imag(I_f),real(I_f) )) ;
printf("\n \n (c) Sequence voltages are , [V_a0 ; V_a1 ; V_a2 ] = pu V \n") ;
printf("\n    %.4f<%.1f ",abs(V_a),atand(imag(V_a),real(V_a) )) ;
printf("\n \n    Phase voltages are , [V_af ; V_bf ; V_cf ] = pu V \n") ;
printf("\n    %.4f<%.1f ",abs(V_f),atand(imag(V_f),real(V_f) )) ;
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) )) ;
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) )) ;
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) )) ;

printf("\n NOTE : ERROR : Calclation mistake in textbook from case(c) onwards \n") ;
```

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
```

```
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.10 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
Z_0 = 0.2619 * %i ;
Z_1 = 0.25 * %i ;
Z_2 = 0.25 * %i ;
v = 1 * exp(%i*0*%pi/180) ;
a = 1 * exp(%i*120*%pi/180) ; // By symmetrical components theory to 3−   system
A = [1 1 1; 1 a^2 a ;1 a a^2] ;

// CALCULATIONS
// For case (b)
I_a0 = v/(Z_0 + Z_1 + Z_2) ; // Sequence currents at fault point F in pu A
I_a1 = I_a0 ;
I_a2 = I_a0 ;

// For case (c)
I_a1g1 = (1/2) * I_a1 ; // Sequence current at terminals of generator G1 in pu A
I_a2g1 = (1/2) * I_a2 ;
I_a0g1 = 0.5/(0.55 + 0.5)*I_a0 ; // By current division in pu A

// For case (d)
I_f = [A] * [I_a0g1 ; I_a1g1 ; I_a2g1] ; // Phase current at terminal of generator G1 in pu A

```

```
// For case (e)
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

// For case (f)
V_f = [A]*[V_a] ; // Phase voltage at terminal of generator G1 in pu V

// For case (g)
I_a1g2 = (1/2) * I_a1 ; // By symmetry for Generator G2
I_a2g2 = (1/2) * I_a2 ;
I_a0g2 = 0 ; // By inspection
// V_a1(HV) leads V_a1(LV) by 30 degree & V_a2(HV) lags V_a2(LV) by 30 degree
I_a0G2 = I_a0g2 ;
I_a1G2 = abs(I_a1g2)*exp(%i * (atand( imag(I_a1g2),real(I_a1g2) ) - 30) * %pi/180) ; // (-90-30) = (-120)
I_a2G2 = abs(I_a2g2)*exp(%i *(atand( imag(I_a2g2),real(I_a2g2) ) + 30) * %pi/180)  ; // (-90+30) = (-60)

I_f2 = [A] * [I_a0G2 ; I_a1G2 ; I_a2G2] ; // Phase current at terminal of generator G2 in pu A

 // Sequence voltage at terminal of generator G2 in pu V
V_a0G2 = 0 ;
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)
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)

V_f2 = A * [V_a0G2 ; V_a1G2 ; V_a2G2] ; // Phase
```

```
voltage at terminal of generator G2 in pu V

// DISPLAY RESULTS
disp("EXAMPLE : 9.10 : SOLUTION :-") ;
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) )) ;
printf("\n (c) Sequence currents at the terminals of generator G1 , \n") ;
printf("\n      I_a0,G_1 = %.4f<%.f pu A ",abs(I_a0g1),atand( imag(I_a0g1),real(I_a0g1) )) ;
printf("\n      I_a1,G_1 = %.4f<%.f pu A ",abs(I_a1g1),atand( imag(I_a1g1),real(I_a1g1) )) ;
printf("\n      I_a2,G_1 = %.4f<%.f pu A ",abs(I_a2g1),atand( imag(I_a2g1),real(I_a2g1) )) ;
printf("\n \n (d) Phase currents at terminal of generator G1 are , [I_af ; I_bf ; I_cf] = pu A \n") ;
printf("\n        %.4f<%.f ",abs(I_f),atand(imag(I_f),real(I_f) )) ;
printf("\n \n (e) Sequence voltages at the terminals of generator G1 , [V_a0 ; V_a1 ; V_a2 ] = pu V \n") ;
printf("\n        %.4f<%.1f ",abs(V_a),atand(imag(V_a),real(V_a) )) ;
printf("\n \n (f) Phase voltages at terminal of generator G1 are , [V_af ; V_bf ; V_cf] = pu V \n") ;
printf("\n        %.4f<%.1f ",abs(V_f),atand(imag(V_f),real(V_f) )) ;
printf("\n \n (g) Sequence currents at the terminals of generator G2 , \n") ;
printf("\n      I_a0,G_2 = %.f<%.f pu A ",abs(I_a0G2),atand( imag(I_a0G2),real(I_a0G2) )) ;
printf("\n      I_a1,G_2 = %.4f<%.f pu A",abs(I_a1G2),atand( imag(I_a1G2),real(I_a1G2) )) ;
printf("\n      I_a2,G_2 = %.4f<%.f pu A",abs(I_a2G2),atand( imag(I_a2G2),real(I_a2G2) )) ;
```

```
printf("\n \n      Phase currents at terminal of generator G2 are , [I_af ; I_bf ; I_cf] = pu A \n");
printf("\n        %.4f<%.f ",abs(I_f2),atand(imag(I_f2),real(I_f2) )) ;
printf("\n \n      Sequence voltages at the terminals of generator G2 , [V_a0 ; V_a1 ; V_a2 ] = pu V\n");
printf("\n          %.f<%.f ",abs(V_a0G2),atand( imag(V_a0G2),real(V_a0G2) )) ;
printf("\n          %.4f<%.f  ",abs(V_a1G2),atand( imag(V_a1G2),real(V_a1G2) )) ;
printf("\n          %.4f<%.f  ",abs(V_a2G2),atand( imag(V_a2G2),real(V_a2G2) )) ;
printf("\n \n      Phase voltages at terminal of generator G2 are , [V_af ; V_bf ; V_cf] = pu V \n");
printf("\n          %.4f<%.1f  ",abs(V_f2),atand(imag(V_f2),real(V_f2) )) ;

printf("\n \n NOTE : ERROR : Calclation mistake in textbook case(f) ") ;
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 Line-toLineVoltages at Faultpoints

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

```

```
// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.11 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
kv = 230 ; // Line voltage in kV from Exa 9.9
Z_0 = 0.56*%i ; // Zero-sequence impedance in pu
Z_1 = 0.2618*%i ; // Zero-sequence impedance in pu
Z_2 = 0.3619*%i ; // Zero-sequence impedance in pu
z_f = 5 ; // Fault impedance in
v = 1*exp(%i*0*%pi/180) ; //
a = 1*exp(%i*120*%pi/180) ; // By symmetrical components theory to 3-  system
A = [1 1 1; 1 a^2 a ;1 a a^2] ;

// CALCULATIONS
// For case (b)
I_a0 = 0 ; // Sequence current in A
Z_B = kv^2/200 ; // Base impedance of 230 kV line
Z_f = z_f/Z_B ; // fault impedance in pu
I_a1 = v/(Z_1 + Z_2 + Z_f) ; // Sequence current in pu A
I_a2 = - I_a1 ; // Sequence current in pu A
I_f = [A] * [I_a0 ; I_a1 ; I_a2] ; // Phase current in pu A

// For case (c)
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
V_f = A*V_a ; // Phase voltages in pu V

// For case (d)
V_abf = V_f(1,1) - V_f(2,1) ; // Line-to-line voltages at fault points in pu V
```

```
V_bcf = V_f(2,1) - V_f(3,1) ; // Line-to-line voltages at fault points in pu V
V_caf = V_f(3,1) - V_f(1,1) ; // Line-to-line voltages at fault points in pu V



// DISPLAY RESULTS
disp("EXAMPLE : 9.11 :SOLUTION :-") ;
printf("\n (b) Sequence currents are , \n") ;
printf("\n  I_a0 = %.f pu A ",I_a0) ;
printf("\n  I_a1 = %.4f<%.2f pu A ",abs(I_a1),atand( imag(I_a1),real(I_a1) )) ;
printf("\n  I_a2 = %.4f<%.2f pu A ",abs(I_a2),atand( imag(I_a2),real(I_a2) )) ;
printf("\n \n Phase currents are , [I_af ; I_bf ; I_cf] =  pu A \n") ;
printf("\n       %.4f<%.1f ",abs(I_f),atand(imag(I_f),real(I_f) )) ;
printf("\n \n (c) Sequence voltages are , [V_a0 ; V_a1 ; V_a2] =  pu V \n") ;
printf("\n       %.4f<%.1f ",abs(V_a),atand(imag(V_a),real(V_a) )) ;
printf("\n \n  Phase voltages are , [V_af ; V_bf ; V_cf] =  pu V \n") ;
printf("\n       %.4f<%.1f ",abs(V_f),atand(imag(V_f),real(V_f) )) ;
printf("\n \n (d) Line-to-line voltages at the fault points are \n") ;
printf("\n     V_abf = %.4f<%.1f pu V \n",abs(V_abf),atand( imag(V_abf),real(V_abf) )) ;
printf("\n     V_bcf = %.4f<%.1f pu V \n",abs(V_bcf),atand( imag(V_bcf),real(V_bcf) )) ;
printf("\n     V_caf = %.4f<%.1f pu V \n",abs(V_caf),atand( imag(V_caf),real(V_caf) )) ;

printf("\n \n NOTE : ERROR : Minor calclation mistake in textbook ") ;
```

Scilab code Exa 9.12 determine Iphase Isequence Vphase Vsequence LinetoLineVoltages at Faultpoints

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.12 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
z_f = 5 ; // Fault-impedance in
z_g = 10 ; // Ground-impedance in
kv = 230 ; // Line voltage in kV from Exa 9.9
Z_0 = 0.56*%i ; // Zero impedance in pu
Z_1 = 0.2618*%i ; // Positive sequence Impedance in pu
Z_2 = 0.3619*%i ; // Negative sequence Impedance in pu
v = 1*exp(%i*0*180/%pi) ;
a = 1*exp(%i*120*%pi/180) ; // By symmetrical components theory to 3-  system
A = [1 1 1; 1 a^2 a ;1 a a^2] ;

// CALCULATIONS
// For case (b)
Z_B = kv^2/200 ; // Base impedance of 230 kV line
Z_f = z_f/Z_B ; // fault impedance in pu
```

```
Z_g = z_g/Z_B ;
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
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
I_a0 = -[(Z_2 + Z_f)/( (Z_2 + Z_f)+(Z_0 + Z_f + 3*Z_g) )]*I_a1 ; // Sequence current in pu A
I_f = A*[I_a0 ; I_a1 ; I_a2] ; // Phase currents in pu A

// For case (c)
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
V_f = A*[V] ; // Phase voltages in pu V

// For case (d)
V_abf = V_f(1,1) - V_f(2,1) ; // Line-to-line voltages at fault points a & b
V_bcf = V_f(2,1) - V_f(3,1) ; // Line-to-line voltages at fault points b & c
V_caf = V_f(3,1) - V_f(1,1) ; // Line-to-line voltages at fault points c & a

// DISPLAY RESULTS
disp("EXAMPLE : 9.12 : SOLUTION :-") ;
printf("\n (b) Sequence currents are , \n") ;
printf("\n   I_a0 = %.4f∠%.2f pu A ",abs(I_a0),atand( imag(I_a0),real(I_a0) )) ;
printf("\n   I_a1 = %.4f∠%.2f pu A ",abs(I_a1),atand( imag(I_a1),real(I_a1) )) ;
printf("\n   I_a2 = %.4f∠%.2f pu A ",abs(I_a2),atand( imag(I_a2),real(I_a2) )) ;
printf("\n \n   Phase currents are , [I_af ; I_bf ; I_cf] = pu A \n ") ;
printf("\n       %.4f∠%.1f ",abs(I_f),atand(imag(I_f
```

```
),real(I_f) ));
printf("\n \n (c) Sequence voltages , [V_a0 ; V_a1 ; V_a2] = pu V \n ");
printf("\n       %.4f<%.1f ",abs(V),atand(imag(V),real(V) ));
printf("\n \n  Phase voltages , [V_af ; V_bf ; V_cf] = pu V \n ");
printf("\n       %.4f<%.1f ",abs(V_f),atand(imag(V_f),real(V_f) ));
printf("\n \n (d) Line-to-line voltages at the fault points are , \n");
printf("\n   V_abf = %.4f<%.1f pu V \n",abs(V_abf),atand( imag(V_abf),real(V_abf) ));
printf("\n   V_bcf = %.4f<%.1f pu V \n",abs(V_bcf),atand( imag(V_bcf),real(V_bcf) ));
printf("\n   V_caf = %.4f<%.1f pu V \n",abs(V_caf),atand( imag(V_caf),real(V_caf) ));
```

Scilab code Exa 9.13 determine Iphase Isequence Vphase Vsequence LinetoLineVoltages at Faultpoints

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.13 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
```

```
z_f = 5 ; // Fault-impedance in
Z_0 = 0.56*%i ; // Zero impedance in pu
Z_1 = 0.2618*%i ; // Positive sequence Impedance in pu
Z_2 = 0.3619*%i ; // Negative sequence Impedance in pu
kv = 230 ; // Line voltage in kV from Exa 9.9
a = 1 * exp(%i*120*%pi/180) ; // By symmetrical components theory to 3-   system
A = [1 1 1; 1 a^2 a ;1 a a^2] ;

// CALCULATIONS
// For case (b)
Z_B = kv^2/200 ; // Base impedance of 230 kV line
Z_f = z_f/Z_B ; // fault impedance in pu
v = 1*exp(%i*0*%pi/180) ;
I_a0 = 0 ; // Sequence current in pu A
I_a1 = v/(Z_1 + Z_f) ; // Sequence current in pu A
I_a2 = 0 ; // Sequence current in pu A
I_f = A*[I_a0 ; I_a1 ; I_a2] ; // Phase-current in pu A

// For case (c)
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
V_f = A*[V] ; // Phase voltages in pu V

// For case (d)
V_abf = V_f(1,1) - V_f(2,1) ; // Line-to-line voltages at fault points a & b
V_bcf = V_f(2,1) - V_f(3,1) ; // Line-to-line voltages at fault points b & c
V_caf = V_f(3,1) - V_f(1,1) ; // Line-to-line voltages at fault points c & a

// DISPLAY RESULTS
disp("EXAMPLE : 9.13 : SOLUTION :-") ;
```

```
printf("\n (b) Sequence currents are , \n") ;
printf("\n      I_a0 = %.1f pu A ",I_a0) ;
printf("\n      I_a1 = %.4f<%.1f pu A ",abs(I_a1),
   atand( imag(I_a1),real(I_a1) )) ;
printf("\n      I_a2 = %.1f pu A ",I_a2) ;
printf("\n \n   Phase currents are , [I_af ; I_bf ;
   I_cf] = pu A \n ") ;
printf("\n       %.4f<%.1f ",abs(I_f),atand(imag(I_f
   ),real(I_f) )) ;
printf("\n \n (c) Sequence voltages , [V_a0 ; V_a1 ;
    V_a2] = pu V \n ") ;
printf("\n       %.4f<%.1f ",abs(V),atand(imag(V),
   real(V) )) ;
printf("\n \n    Phase voltages , [V_af ; V_bf ;
   V_cf] = pu V \n ") ;
printf("\n       %.4f<%.1f ",abs(V_f),atand(imag(V_f
   ),real(V_f) )) ;
printf("\n \n (d) Line-to-line voltages at the fault
    points are , \n") ;
printf("\n     V_abf = %.4f<%.1f pu V \n",abs(V_abf),
   atand( imag(V_abf),real(V_abf) )) ;
printf("\n     V_bcf = %.4f<%.1f pu V \n",abs(V_bcf),
   atand( imag(V_bcf),real(V_bcf) )) ;
printf("\n     V_caf = %.4f<%.1f pu V \n",abs(V_caf),
   atand( imag(V_caf),real(V_caf) )) ;

printf("\n \n NOTE : ERROR : Calclation mistake in
   textbook case(d) ") ;
```

Scilab code Exa 9.14 determine admittance matrix

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
```

```
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT
   ANALYSIS

// EXAMPLE : 9.14 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
VG_1 = 1*exp(%i*0*%pi/180) ;
VG_2 = 1*exp(%i*0*%pi/180) ;

// CALCULATIONS
// For case (a)
I_1 = 1*exp(%i*0*%pi/180) ;
I_2 = 1*exp(%i*0*%pi/180) ;
V_1 = 0.4522*exp(%i*90*%pi/180) ;
V_2 = 0.4782*exp(%i*90*%pi/180) ;
Y_11 = I_1/V_1 ; // When V_2 = 0
Y_21 = (-0.1087)*Y_11 ; // When V_2 = 0
Y_22 = I_2/V_2 ; // When V_1 = 0
Y_12 = Y_21 ;
Y = [Y_11 Y_12 ; Y_21 Y_22] ; // Admittance matrix
   associated with positive-sequence n/w

// For case (b)
I_S1_12 = 2.0193*exp(%i*90*%pi/180) ; // Short-ckt F
    & F' to neutral & by superposition theorem
I_S1_10 = 0.2884*exp(%i*90*%pi/180) ; // Short-ckt F
    & F' to neutral & by superposition theorem
I_S2_12 = 0.4326*exp(%i*90*%pi/180) ;
I_S2_10 = 1.4904*exp(%i*90*%pi/180) ;
I_S1 = I_S1_12 + I_S1_10 ;
I_S2 = I_S2_12 + I_S2_10 ;

// DISPLAY RESULTS
disp("EXAMPLE : 9.14 :SOLUTION :-") ;
```

```
printf("\n (a) Admittance matrix associated with positive-sequence network , Y = \n") ; disp(Y) ;
printf("\n (b) Source currents Two-port Thevenin equivalent positive sequence network are , \n") ;
printf("\n     I_S1 = %.4f<%.f pu ",abs(I_S1),atand( imag(I_S1),real(I_S1) )) ;
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

```

// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.15 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
Y_11 = -2.2115*%i ;
Y_12 = 0.2404*%i ;
Y_21 = 0.2404*%i ;
Y_22 = -2.0912*%i ;
Y = [Y_11 Y_12 ; Y_21 Y_22] ;
I_S1 = 2.3077*%i ;
I_S2 = 1.9230*%i ;

I_a1 = poly(0,'I_a1') ;
```

```
I_a2 = poly(0,'I_a2') ;
a = Y_12*I_S2 - Y_22*I_S1 ;
b = (Y_12+Y_22)*I_a1 ;
c = Y_12*I_S1 - Y_11*I_S2 ;
d = (Y_12 + Y_11)*I_a1 ;
V1 = (1/det(Y))*[(a-b) ; (c+d)] ; // Gives the uncoupled positive sequence N/W
A = (Y_12+Y_22)*I_a2 ;
B = (Y_12 + Y_11)*I_a2 ;
V2 = (1/det(Y))*[A ; B] ; // Gives the uncoupled negative sequence N/W

// DISPLAY RESULTS
disp("EXAMPLE : 9.15 : SOLUTION :-") ;
printf("\n (a) [V_a1 ; V_a11] = ") ; disp(V1) ;
printf("\n      Values of Uncoupled positive-sequence network \n") ;
printf("\n (b) [V_a2 ; V_a22] = ") ; disp(V2) ;
printf("\n      Values of Uncoupled negative-sequence network \n") ;
```

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 9 : SYMMETRICAL COMPONENTS AND FAULT ANALYSIS

// EXAMPLE : 9.16 :
clear ; clc ; close ; // Clear the work space and console
```

```

// GIVEN DATA
H_aa = 81.5 ;
D_aa = 1.658 ;
f = 60 ; // Freq in Hz
I = 20 ;
kV = 69 ; // Line voltage in kV
MVA = 25 ; // Transformer T1 rating in MVA

// CALCULATIONS
// For case (a)
C_0 = 29.842*10^-9/(log(H_aa/D_aa)) ; // Capacitance in F/mi
b_0 = 2*%pi*f*C_0 ; // Susceptance in S/mi
B_0 = b_0*I ; // For total system
X_C0 = (1/B_0) ; // Total zero-sequence reactance in
TC_0 = B_0/(2*%pi*f) ; // Total zero-sequence capacitance in F

// For case (c)
X_1 = 0.05 ; // Leakage reactance of transformer T1 in pu
X_0 = X_1 ;
X_2 = X_1 ;
Z_B = kV^2/MVA ;
X_01 = X_0*Z_B ; // Leakage reactance in
V_F = 69*10^3/sqrt(3) ;
I_a0PC = V_F/(17310.8915*%i) ; // Zero-sequence current flowing through PC in A
I_PC = 3*abs(I_a0PC) ; // Continuous-current rating of the PC in A

// For case (d)
X_PC = (17310.8915 - X_01)/3 ; // Required reactance value for PC in

// For case (e)
```

```
L_PC = X_PC/(2*%pi*f) ; // Inductance in H

// For case (f)
S_PC = (I_PC^2)*X_PC ; // Rating in VA
S_PC1 = S_PC*10^-3 ; // Continuous kVA rating in kVA

// For case (g)
V_PC = I_PC * X_PC ; // continuous-voltage rating for PC in V

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 10 : PROTECTIVE EQUIPMENT AND
   TRANSMISSION SYSTEM PROTECTION

// EXAMPLE : 10.1 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
X_d = 0.14*%i ; // Reactance of generator in pu
E_g = 1*exp(%i*0*%pi/180) ;
```

```
S_B = 25*10^3 ; // voltage in kVA
V_BL_V = 13.8 ; // low voltage in kV

// CALCULATIONS
I_f = E_g/X_d ; // Subtransient fault current in pu
I_BL_V = S_B/( sqrt(3)*V_BL_V) ; // Current base for
    low-voltage side
I_f1 = abs(I_f)*I_BL_V ; // magnitude of fault
    current in A

// DISPLAY RESULTS
disp("EXAMPLE : 10.1 : SOLUTION :-") ;
printf("\n Subtransient fault current for 3-   fault
    in per units = pu \n") ; disp(I_f) ;
printf("\n Subtransient fault current for 3-   fault
    in ampere = %.f A \n",I_f1) ;
```

Scilab code Exa 10.2 determine max Idc Imax Imomentary Sinterrupting Smomentary

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 10 : PROTECTIVE EQUIPMENT AND
    TRANSMISSION SYSTEM PROTECTION

// EXAMPLE : 10.2 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
// For case (a)
```

```
I_f = 7.1428571 ; // Subtransient fault current in pu . Result of exa 10.1

// For case (d)
V_pf = 13800 ; // voltage in V
zeta = 1.4 ;
I_f1 = 7471 ; // magnitude of fault current in A

// CALCULATIONS
// For case (a)
I_fdc_max = sqrt(2)*I_f ; // Max dc current in pu

// For case (b)
I_f_max = 2*I_fdc_max ; // Total max instantaneous current in pu

// For case (c)
I_momt = 1.6*I_f ; // Total rms momentary current

// For case (d)
S_int = sqrt(3)*(V_pf)*I_f1*zeta*10^-6 ; // Interrupting rating in MVA

// For case (e)
S_momt = sqrt(3)*(V_pf)*I_f1*1.6*10^-6 ; // Momentary duty of CB in MVA

// DISPLAY RESULTS
disp("EXAMPLE : 10.2 : SOLUTION :-") ;
printf("\n (a) Maximum possible dc current component , I_fdc_max = %.1f pu \n",I_fdc_max) ;
printf("\n (b) Total maximum instantaneous current , I_max = %.1f pu \n",I_f_max) ;
printf("\n (c) Momentary current , I_momentary = %.2f pu \n",I_momt) ;
printf("\n (d) Interrupting rating of a 2-cycle CB , S_interrupting = %.f MVA \n",S_int) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 10 : PROTECTIVE EQUIPMENT AND TRANSMISSION SYSTEM PROTECTION

// EXAMPLE : 10.4 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
z_l = 0.2 + %i * 0.7 ; // Line impedance in pu
f_l = 0.7 ; // Fault point at a distance from A in pu
f_m = 1.2 ; // magnitude of fault current in pu
l = 10.3 ; // Line spacing in ft
p = 100 ; // Power in MVA
v = 138 ; // voltage in kV
i = 418.4 ; // current in A
z = 190.4 ; // Impedance in

// CALCULATIONS
// For case (a)
I = f_m * i ; // Current in arc in A
R_arc = 8750 * l/(I^1.4) ; // Arc resistance in
R_arc1 = R_arc/z ; // Arc resistance in pu

```

```
// For case (b)
Z_L = z_l * f_l ;
Z_r = Z_L + R_arc1 ; // Impedance seen by the relay
   in pu

// For case (c)
phi_1 = atand( imag(Z_L),real(Z_L) ) ; // Line
   impedance angle without arc resistance in degree
phi_2 = atand( imag(Z_r),real(Z_r) ) ; // Line
   impedance angle with arc resistance in degree

// DISPLAY RESULTS
disp("EXAMPLE : 10.4 : SOLUTION :-") ;
printf("\n (a) Value of arc resistance at fault
   point in    , R_arc = %.2f    \n",R_arc) ;
printf("\n     Value of arc resistance at fault
   point in pu , R_arc = %.2f pu \n",R_arc1) ;
printf("\n (b) Value of line impedance including the
    arc resistance , Z_L + R_arc = pu \n") ; disp(
   Z_r) ;
printf("\n (c) Line impedance angle without arc
   resistance ,   = %.2f degree \n",phi_1) ;
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 impedance

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 10 : PROTECTIVE EQUIPMENT AND
```

```
    TRANSMISSION SYSTEM PROTECTION

// EXAMPLE : 10.5 :
clear ; clc ; close ; // Clear the work space and console

// CALCULATIONS
// For case (a)
// Coordinate Values taken here are only for reference . Refer exa 10.5

T = 0:0.01:300 ;

for i = 1:int(length(T)/1.1) ;
    po(i) = 4 ;
end
for i = int(length(T)/1.1):length(T)
    po(i) = 5 ;
end
for i = 1:int(length(T)/1.1)
    io(i) = 4 ;
    end
for i = int(length(T)/1.1):length(T)
    io(i) = 3 ;
end

a= gca() ;
subplot(2,1,1) ; // To plot 2 graph in same graphic window
a.thickness = 2 ; // sets thickness of plot of points
plot2d(T,po,3,'012','',[0 0 310 7]) ;
plot2d(T,io,3,'012','',[0 0 310 7]) ;
xtitle("Fig 10.5 (a)   Zones of protection for relay R_12") ;
xset('thickness',2); // sets thickness of axes
xstring(25,3.8,'[]') ;
xstring(45,4.2,'(1)') ;
```

```
plot(45,4,'+') ;
xstring(60,3.8,'[]') ;
xstring(60,4.2,'B_12') ;
xstring(120,3.8,'[]') ;
xstring(120,4.2,'B_21') ;
xstring(140,4.2,'(2)') ;
plot(140,4,'+') ;
xstring(155,3.8,'[]') ;
xstring(155,4.2,'B_23') ;
xstring(220,3.8,'[]') ;
xstring(220,4.2,'B_32') ;
xstring(270,5.0,'(3)') ;
xstring(285,2.8,'[]') ;
xstring(285,3.2,'B_35') ;
xstring(285,4.8,'[]') ;
xstring(285,5.2,'B_34') ;
xstring(85,3.4,'TL_12') ;
xstring(180,3.4,'TL_23') ;
xstring(60,3,'ZONE 1') ;
xstring(100,2,'ZONE 2') ;
xstring(190,1,'ZONE 3') ;

// For case (b)

for i = 1:int(length(T)/4) ;
    vo(i) = 0.5;
end
for i = int(length(T)/4):length(T/1.7)
    vo(i) = 2;
end
for i = int(length(T)/1.7):length(T)
    vo(i) = 4
end

for i = int(length(T)/2.14):length(T/1.35) ; // plotting Voltage values
    uo(i) = 0.5;
end
```

```
for i = int(length(T)/1.35):length(T)
    uo(i) = 2;
end

a = gca() ;
a.thickness = 2 ;
subplot(2,1,2)
plot2d(T,vo,2,'012','',[0 0 310 7]) ;
plot2d(T,uo,2,'012','',[0 0 310 7]) ;
ylabel("OPERATING TIME") ;
xlabel("IMPEDANCE") ;
xtitle("Fig 10.5 (b) Coordination of distance relays, Operating time v/s Impedance") ;
xset('thickness',2); // sets thickness of axes
xstring(0.1,0.3,'T_1') ;
xstring(30,0.6,'R_12') ;
xstring(58,1.3,'T_2') ;
xstring(100,2.0,'R_12') ;
xstring(160,3.0,'T_3') ;
xstring(230,4.0,'R_12') ;
xstring(160,0.6,'R_23') ;
xstring(260,2.1,'R_23') ;

// DISPLAY RESULTS
disp("EXAMPLE : 10.5 : SOLUTION :-") ;
printf("\n (a) The zone of protection for relay R_12 is shown in Fig 10.5 (a) \n") ;
printf("\n  ZONE 1 lies b/w (1) & B_21 \n") ;
printf("\n  ZONE 2 lies b/w (1) & TL_23 \n") ;
printf("\n  ZONE 3 lies after (1) \n") ;
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 Imax CT VT ZLoad Zr

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 10 : PROTECTIVE EQUIPMENT AND TRANSMISSION SYSTEM PROTECTION

// EXAMPLE : 10.6 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
kv = 230 * 10^3 ; // transmission system voltage in V
VA = 100 * 10^6 ; // Maximum peak load supplied by TL_12 in VA
ZTL_12 = 2 + %i * 20 ; // Positive-sequence impedances of line TL_12
ZTL_23 = 2.5 + %i * 25 ; // Positive-sequence impedances of line TL_23
pf = 0.9 ; // Lagging pf

// CALCULATIONS
// For case (a)
I_max = VA/(sqrt(3)*kv) ; // Maximum load current in A

// For case (b)
CT = 250/5 ; // CT ratio which gives about 5A in secondary winding under the maximum loading

// For case (c)
vr = 69 ; // selecting Secondary voltage of 69 V line to neutral
VT = (kv/sqrt(3))/vr ; // Voltage ratio

```

```
// For case (d)
Z_r = CT/VT ; // impedance measured by relay . Z_r =
    (V/VT)/(I/CT)
Z_TL_12 = Z_r * ZTL_12 ; // Impedance of lines TL_12
    as seen by relay
Z_TL_23 = Z_r * ZTL_23 ; // Impedance of lines TL_23
    as seen by relay

// For case (e)
Z_load = vr * CT * (pf + %i*sind(acosd(pf)))/(I_max)
    ; // Load impedance based on secondary ohms

// For case (f)
Z_r1 = 0.80 * Z_TL_12 ; // Zone 1 setting of relay
   R_12

// For case (g)
Z_r2 = 1.20 * Z_TL_12 ; // Zone 2 setting of relay
   R_12

// For case (h)
Z_r3 = Z_TL_12 + 1.20*(Z_TL_23) ; // Zone 3 setting
   of relay R_12

// DISPLAY RESULTS
disp("EXAMPLE : 10.6 : SOLUTION :-") ;
printf("\n (a) Maximum load current , I_max = %.2f A
    \n",I_max) ;
printf("\n (b) CT ratio , CT = %.1f \n",CT) ;
printf("\n (c) VT ratio , VT = %.1f \n",VT) ;
printf("\n (d) Impedance measured by relay = %.3f
   Z_line \n",Z_r) ;
printf("\n (e) Load impedance based on secondary
   ohms , Z_load = (secondary) \n") ; disp(Z_load)
    ;
printf("\n (f) Zone 1 setting of relay R_12 , Z_r =
     (secondary) \n") ; disp(Z_r1) ;
printf("\n (g) Zone 2 setting of relay R_12 , Z_r =
```

```
    (secondary) \n") ; disp(Z_r2) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 10 : PROTECTIVE EQUIPMENT AND
   TRANSMISSION SYSTEM PROTECTION

// EXAMPLE : 10.7 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
Z_r1 = 0.0415692 + %i*0.4156922 ; // Required zone 1
     setting . From result of exa 10.6
Z_r2 = 0.0623538 + %i*0.6235383 ; // Required zone 2
     setting . From result of exa 10.6
Z_r3 = 0.1299038 + %i*1.2990381 ; // Required zone 3
     setting . From result of exa 10.6

// CALCULATIONS
// For case (a)
theta1 = atand(imag(Z_r1),real(Z_r1)) ;
Z_1 = abs(Z_r1)/cosd(theta1 - 30) ; // Zone 1
    setting of mho relay R_12

// For case (b)
```

```
theta2 = atand(imag(Z_r2),real(Z_r2)) ;
Z_2 = abs(Z_r2)/cosd(theta2 - 30) ; // Zone 2 setting of mho relay R_12

// For case (b)
theta3 = atand(imag(Z_r3),real(Z_r3)) ;
Z_3 = abs(Z_r3)/cosd(theta3 - 30) ; // Zone 3 setting of mho relay R_12

// DISPLAY RESULTS
disp("EXAMPLE : 10.7 : SOLUTION :-") ;
printf("\n (a) Zone 1 setting of mho relay R_12 = %.4f  (secondary) \n",Z_1) ;
printf("\n (b) Zone 2 setting of mho relay R_12 = %.4f  (secondary) \n",Z_2) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 12 : CONSTRUCTION OF OVERHEAD LINES

// EXAMPLE : 12.1 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
cost_avg = 1500 ; // Average cost on each repair in
    $
r_0 = 0 ; // No. of times repair required for damage
     to line
r_1 = 1 ; // No. of times repair required
r_2 = 2 ; // No. of times repair required
```

```
r_3 = 3 ; // No. of times repair required
P_r_0 = 0.4 ; // Probability of exactly no. of repairs for r_0
P_r_1 = 0.3 ; // Probability of exactly no. of repairs for r_1
P_r_2 = 0.2 ; // Probability of exactly no. of repairs for r_2
P_r_3 = 0.1 ; // Probability of exactly no. of repairs for r_3
R_0 = 0 ; // No. of times repair required for relocating & rebuilding
R_1 = 1 ; // No. of times repair required
P_R_0 = 0.9 ; // Probability of exactly no. of repairs for R_0
P_R_1 = 0.1 ; // Probability of exactly no. of repairs for R_1
n = 25 ; // useful life in years
i = 20/100 ; // carrying charge rate
p = ((1 + i)^n - 1)/(i*(1+i)^n) ; // p = P/A . Refer page 642

// CALCULATIONS
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

// DISPLAY RESULTS
disp("EXAMPLE : 12.1 : SOLUTION :-") ;
printf("\n Affordable cost of relocating line , B = $ %.1f \n",B) ;
printf("\n Since actual relocating & rebuilding of line would cost much more than amount found \n") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 12 : CONSTRUCTION OF OVERHEAD LINES

// EXAMPLE : 12.2 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
V = 40 ; // Actual wind velocity in mi/hr
c_pg = 40 ; // Circumference at ground level in inches
c_pt = 28 ; // Circumference at pole top in inches
l = 35 ; // height of pole in feet
l_g = 6 ; // Height of pole set in ground in feet
d_c = 0.81 ; // dia. of copper conductor in inches
span_avg = 120 ; // Average span in ft
no_c = 8 ; // NO. of conductors

// CALCULATIONS
// For case (a)
p = 0.00256 * (V^2) ; // Buck's Formula to find wind pressure on cylindrical surface in lb/ft^2
d_pg = c_pg/(%pi) ; // dia. of pole at ground line in inches
d_pt = c_pt/(%pi) ; // dia. of pole at pole top in inches
h_ag = ( l - l_g ) * 12 ; // Height of pole above
```

```
    ground in inch
S_pni = (1/2) * (d_pg + d_pt) * h_ag ; // projected
    area of pole in square inch
S_pni_ft = S_pni * 0.0069444 ; // projected area of
    pole in square ft
P = S_pni_ft * p ; // Total pressure of wind on pole
     in lb

// For case (b)
S_ni = d_c * span_avg * 12 ; // Projected area of
    conductor in square inch . [1 feet = 12 inch]
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]
P_C = S_ni_ft * p * no_c ; // Total pressure of wind
     on conductor in lb

// DISPLAY RESULTS
disp("EXAMPLE : 12.2 : SOLUTION :-");
printf("\n (a) Total pressure of wind on pole , P =
   %.2f lb \n",P);
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 12 : CONSTRUCTION OF OVERHEAD LINES

```

```
// EXAMPLE : 12.3 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
a = 45 ; // OH line to be bulit on wood poles in ft
b = 6.5 ; // Ground depth in ft
c = 1 ; // Top cross-arm below pole top in ft
d = 3 ; // Lower cross-arm below pole top in ft
m_t = 0.6861 ; // Transverse wind load on top cross-arm in lb/ft
m_l = 0.4769 ; // Transverse wind load on lower cross-arm in lb/ft
u_s = 8000 ; // Ultimate strength of wood pole in lb/sq.in
sf = 2 ; // Safety factor
span_avg = 250 ; // Average span in ft
p = 9 ; // Transverse wind load on wood poles in clb/sq.ft

// CALCULATIONS
h_1j = a - b - c ; // Moment arms for top arm in ft
h_2j = a - b - d ; // Moment arms for top arm in ft
M_tc1 = 1 * 4* m_t * span_avg * h_1j ; // Total bending moment for top arm in lb-ft
M_tc2 = 1 * 4* m_l * span_avg * h_2j ; // Total bending moment for lower arm in lb-ft
M_tc = M_tc1 + M_tc2 ; // Total bending moment for both cross-arms together in lb-ft
S = u_s/sf ; // Allowable max fiber stress in pounds per sq.inch
c_pg = ( M_tc/( 2.6385*10^-4*S ) )^(1/3) ; // circumference of pole at ground line in inch

c_pt = 22 ; // From proper tables , for 8000 psi ,
h_ag = a - b ; // Height of pole above ground in ft
d_pg = c_pg/(%pi) ; // circumference of pole at ground line in inches
```

```
d_pt = c_pt/(%pi) ; // circumference of pole at pole
    top in inches
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
M_T = M_tc + M_gp ; // Total bending moment due to
    wind on conductor & pole
c_pg1 = (M_T/( 2.6385 * 10^-4 * S ) )^(1/3) ; //
    using equ 12.11

// DISPLAY RESULTS
disp("EXAMPLE : 12.3 : SOLUTION :-") ;
printf("\n Minimum required pole circumference at
    the ground line , c = %.1f in \n",c_pg1) ;
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) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 12 : CONSTRUCTION OF OVERHEAD LINES

// EXAMPLE : 12.4 :
clear ; clc ; close ; // Clear the work space and
    console

```

```
// GIVEN DATA
T1 = 3000 ; // Bending moments in lb
T2 = 2500 ; // Bending moments in lb
h1 = 37.5 ; // Bending moments at heights in ft
h2 = 35.5 ; // Bending moments at heights in ft
h_g = 36.5 ; // Height at which Guy is attached to pole in ft
L = 15 ; // Lead of guy in ft

// CALCULATIONS
// For case (a)
T_h = ( T1*h1 + T2*h2 )/h_g ; // Horizontal component of tension in guy wire in lb . From equ 12.26

// For case (b)
bet = atand(h_g/L) ; // beta angle in degree . From equ 12.28

// For case (c)
T_v = T_h * tand(bet) ; // Vertical component of tension in guy wire in lb . From equ 12.34

// For case (d)
T_g = T_h/( cosd(bet )) ; // Tension in guy wire in lb . From equ 12.29
T_g1 = sqrt( T_h^2 + T_v^2 ) ; // Tension in guy wire in lb

// DISPLAY RESULTS
disp("EXAMPLE : 12.4 : SOLUTION :-") ;
printf("\n (a) Horizontal component of tension in guy wire , T_h = %.1f lb \n",T_h) ;
printf("\n (b) Angle    ,    = %.2f degree \n",bet) ;
printf("\n (c) Vertical component of tension in guy wire , T_v = %.2f lb \n",T_v) ;
printf("\n (d) Tension in guy wire , T_g = %.1f lb \
```

```
    n",T_g) ;
printf("\n      (or)  From another equation , \n") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 13 : SAG AND TENSION ANALYSIS

// EXAMPLE : 13.1 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
c = 1600 ; // Length of conductor in feet
L = 500 ; // span b/w conductors in ft
w1 = 4122 ; // Weight of conductor in lb/mi

// CALCULATIONS
// For case (a)
```

```
l = 2 * c *( sinh(L/(2*c)) ) ; // Length of conductor in ft using eq 13.6
l_1 = L * (1 + (L^2)/(24*c^2) ) ; // Length of conductor in ft using eq 13.8

// For case (b)
d = c*( cosh( L/(2*c) ) - 1 ) ; // sag in ft

// For case (c)
w = w1/5280 ; // Weight of conductor in lb/ft . [1 mile = 5280 feet]
T_max = w * (c + d) ; // Max conductor tension in lb
T_min = w * c ; // Min conductor tension in lb

// For case (d)
T = w * (L^2)/(8*d) ; // Appr value of tension in lb using parabolic method

// DISPLAY RESULTS
disp("EXAMPLE : 13.1 : SOLUTION :-") ;
printf("\n (a) Length of conductor using eq 13.6 , l = %.3f ft \n",l) ;
printf("\n & Length of conductor using eq 13.8 , l = %.4f ft \n",l_1) ;
printf("\n (b) Sag , d = %.1f ft \n",d) ;
printf("\n (c) Maximum value of conductor tension using catenary method , T_max = %.1f lb \n",T_max) ;
printf("\n     Minimum value of conductor tension using catenary method , T_min = %.1f lb \n",T_min) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// CHAPTER : 13 : SAG AND TENSION ANALYSIS

// EXAMPLE : 13.2 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
L = 500 ; // span b/w conductors in ft
p = 4 ; // Horizontal wind pressure in lb/sq ft
t_i = 0.50 ; // Radial thickness of ice in inches
d_c = 1.093 ; // outside diameter of ACSR conductor in inches
w1 = 5399 ; // weight of conductor in lb/mi
s = 28500 ; // ultimate strength in lb

// CALCULATIONS
// For case (a)
w_i = 1.25 * t_i * (d_c + t_i) ; // Weight of ice in pounds per feet

// For case (b)
w = w1/5280 ; // weight of conductor in lb/ft . [1 mile = 5280 feet]
W_T = w + w_i ; // Total vertical load on conductor in pounds per feet

// For case (c)
P = ( (d_c + 2*t_i)/(12) )*p ; // Horizontal wind force in lb/ft

// For case (d)
w_e = sqrt( P^2 + (w + w_i)^2 ) ; // Effective load
```

```
    on conductor in lb/ft

// For case (e)
T = s/2 ;
d = w_e * L^2/(8*T) ; // sag in feet

// For case (f)
d_v = d * W_T/w_e ; // vertical sag in feet

// DISPLAY RESULTS
disp("EXAMPLE :13.2 : SOLUTION :-") ;
printf("\n (a) Weight of ice in pounds per feet ,
    w_i = %.4f lb/ft \n",w_i) ;
printf("\n (b) Total vertical load on conductor in
    pounds per feet , W_T = %.4f lb/ft \n",W_T) ;
printf("\n (c) Horizontal wind force in pounds per
    feet , P = %.4f lb/ft \n",P) ;
printf("\n (d) Effective load acting in pounds per
    feet , w_e = %.4f lb/ft \n",w_e) ;
printf("\n (e) Sag in feet , d = %.2f ft \n",d) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// APPENDIX C : REVIEW OF BASICS

// EXAMPLE : C.1 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
z = 100 * exp(60*%i*%pi/180) ; // Impedance of
   transmission line in
v1 = 73034.8 * exp(30*%i*%pi/180) ; // Bus voltages
   in V
v2 = 66395.3 * exp(20*%i*%pi/180) ; // Bus voltages
   in V
```

```
// CALCULATIONS
// For case (a)
S_12 = v1 * ( conj(v1) - conj(v2) )/( conj(z) ) ; // Complex power per phase in VA


// For case (b)
P_12 = real(S_12) ; // Active power per phase in W

// For case (c)
Q_12 = imag(S_12) ; // Reactive power per phase in vars

// DISPLAY RESULTS
disp("EXAMPLE : C.1 : SOLUTION :-") ;
printf("\n (a) Complex power per phase that is being transmitted from bus 1 to bus 2 , S12 = %.2f<%.2f VA \n",abs(S_12), atan(imag(S_12),real(S_12))*(180/%pi)) ;
printf("\n (b) Active power per phase that is being transmitted , P12 = %.2f W \n",P_12) ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// APPENDIX C : REVIEW OF BASICS

```

```
// EXAMPLE : C.2 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
X_pu = 12/100 ; // Leakage reactance in pu
kV_B_HV = 345 ; // HV side ratings in Y kV
kV_B_LV = 34.5 ; // LV side ratings in Y kV
MVA_B = 20 ; // selected Base on HV side in MVA

// CALCULATIONS
// For case (a)
X_pu = 12/100 ; // Reactance of transformer in pu

// For case (b)
Z_B_HV = (kV_B_HV)^2/MVA_B ; // HV side base
    impedance in

// For case (c)
Z_B_LV = (kV_B_LV)^2/MVA_B ; // LV side base
    impedance in

// For case (d)
X_HV = X_pu * Z_B_HV ; // Reactance referred to HV
    side in

// For case (e)
X_LV = X_pu * Z_B_LV ; // Reactance referred to LV
    side in
n = (kV_B_HV/sqrt(3))/(kV_B_LV/sqrt(3)) ; // Turns
    ratio of winding
X_LV1 = X_HV/n^2 ; // From equ C.89

// DISPLAY RESULTS
disp("EXAMPLE : C.2 : SOLUTION :-") ;
printf("\n (a) Reactance of transformer in pu , X_pu
     = %.2f pu \n",X_pu) ;
printf("\n (b) High-voltage side base impedance ,
```

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// APPENDIX C : REVIEW OF BASICS

// EXAMPLE : C.3 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
X_pu = 12/100 ; // Leakage reactance in pu
kV_B_HV = 345 ; // HV side ratings in Y kV
kV_B_LV = 34.5 ; // LV side ratings in    kV
MVA_B = 20 ; // Base on HV side in MVA

// CALCULATIONS
// For case (a)
n = ( kV_B_HV/sqrt(3) )/kV_B_LV ; // Turns ratio of windings
```

```

// For case (b)
Z_B_HV = (kV_B_HV)^2/MVA_B ; // HV side base impedance in
X_HV = X_pu * Z_B_HV ; // Reactance referred to HV side in
X_LV = X_HV/(n^2) ; // transformer reactance referred to delta LV side in

// For case (c)
Z_dt = X_LV ;
Z_Y = Z_dt/3 ; // Reactance of equi wye connection
Z_B_LV = kV_B_LV^2/MVA_B ; // LV side base impedance in
X_pu1 = Z_Y/Z_B_LV ; // reactance in pu referred to LV side

// Alternative method For case (c)
n1 = kV_B_HV/kV_B_LV ; // Turns ratio if line-to-line voltages are used
X_LV1 = X_HV/(n1^2) ; // Reactance referred to LV side in
X_pu2 = X_LV1/Z_B_LV ; // reactance in pu referred to LV side

// DISPLAY RESULTS
disp("EXAMPLE : C.3 : SOLUTION :-") ;
printf("\n (a) Turns ratio of windings , n = %.4f \n",n) ;
printf("\n (b) Transformer reactance referred to LV side in ohms ,X_LV = %.4f     \n",X_LV) ;
printf("\n (c) Transformer reactance referred to LV side in per units ,X_pu = %.2f pu \n",X_pu1) ;
printf("\n    (or) From another equation if line-to-line voltages are used ,") ;
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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
   ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// APPENDIX C : REVIEW OF BASICS

// EXAMPLE : C.4 :
clear ; clc ; close ; // Clear the work space and
   console

// GIVEN DATA
I_1 = 1000 ; // Physical current in A for 2.4 kV
   circuit
Z_pu = 0.04 ; // Leakage reactance in pu
I_pu = 2.08*exp(%i*(-90)*%pi/180) ; // Generator
   supply for pure inductive load
kVA_Bg1 = 6000 ; // Rated kVA values for T1
kVA_Bg2 = 4000 ; // Rated kVA values for T2
N2 = 2.4 ; // N2 = V2 in Y kV ,refer fig C.4
N1 = 24 ; // N1 = V1 in Y kV ,refer fig C.4
N3 = 24 ; // N3 = V3 = N1 in Y kV ,refer fig C.4
N4 = 12 ; // N4 = V4 in Y kV ,refer fig C.4

// CALCULATIONS
// For case (a)
kVA_B = 2080 ; // arbitrarily selected kVA values
   for all 3 ckt

// For case (b)
```

```
n1 = N2/N1 ; // Turns ratio of transformer T1 & T2 i
    .e N2/N1
n2 = N3/N4 ; // Turns ratio N1'/N2'
kV_BL_L1 = 2.5 ; // arbitrarily selected Base
    voltage for 2.4 kV ckt in kV
kV_BL_L2 = kV_BL_L1/n1 ; // arbitrarily selected
    Base voltage for 24 kV ckt in kV
kV_BL_L3 = kV_BL_L2/n2 ; // arbitrarily selected
    Base voltage for 12 kV ckt in kV

// For case (c)
Z_B1 = (kV_BL_L1)^(2) * 1000/(kVA_B) ; // Base
    impedance in   for 2.4 kV ckt
Z_B2 = (kV_BL_L2)^(2) * 1000/(kVA_B) ; // Base
    impedance in   for 24 kV ckt
Z_B3 = (kV_BL_L3)^(2) * 1000/(kVA_B) ; // Base
    impedance in   for 12 kV ckt

// For case (d)
I_B1 = kVA_B/(sqrt(3)*kV_BL_L1) ; // Base current in
     A for 2.4 kV ckt
I_B2 = kVA_B/(sqrt(3)*kV_BL_L2) ; // Base current in
     A for 24 kV ckt
I_B3 = kVA_B/(sqrt(3)*kV_BL_L3) ; // Base current in
     A for 12 kV ckt

// For case (e)
I_2 = (n1) * I_1 ; // Physical current in A for 24
    kV circuit
I_4 = (n2) * I_2 ; // Physical current in A for 12
    kV circuit

// For case (f)
I_pu_3ckt = abs(I_pu) ; // per-unit current values
    for all 3-ckt

// For case (g)
kV_B1 = N2 ; // Given voltage in kV
```

```
kV_B2 = N4 ; // Given voltage in kV
Z_pu_T1 = (%i)*Z_pu*(kVA_B/kVA_Bg1)*(kV_B1/kV_BL_L1)^(2) ; // New reactance of T1
Z_pu_T2 = (%i)*Z_pu*(kVA_B/kVA_Bg2)*(kV_B2/kV_BL_L3)^(2) ; // New reactance of T2

// For case (h)
V1 = kV_B1/kV_BL_L1 ; // voltage in pu at bus 1
V2 = V1 - I_pu * (Z_pu_T1) ; // voltage in pu at bus 2
V4 = V2 - I_pu * (Z_pu_T2) ; // voltage in pu at bus 3

// For case (i)
S1 = V1 * abs(I_pu) ; // Apparent power value at bus 1 in pu
S2 = V2 * abs(I_pu) ; // Apparent power value at bus 2 in pu
S4 = V4 * abs(I_pu) ; // Apparent power value at bus 4 in pu

// DISPLAY RESULTS
disp("EXAMPLE : C.3 : SOLUTION :-") ;
printf("\n (a) Base kilovoltampere value for all 3-circuits is , kVA_B = %.1f kVA \n",kVA_B) ;
printf("\n (b) Base line-to-line kilovolt value for 2.4 kV circuit , kV_BL_L = %.1f kV \n",kV_BL_L1) ;
printf("\n      Base line-to-line kilovolt value for 24 kV circuit , kV_BL_L = %.1f kV \n",kV_BL_L2) ;
printf("\n      Base line-to-line kilovolt value for 24 kV circuit , kV_BL_L = %.1f kV \n",kV_BL_L3) ;
printf("\n (c) Base impedance value of 2.4 kV circuit , Z_B = %.3f     \n",Z_B1) ;
printf("\n      Base impedance value of 24 kV circuit , Z_B = %.1f     \n",Z_B2) ;
printf("\n      Base impedance value of 12.5 kV circuit , Z_B = %.1f     \n",Z_B3) ;
```

```
printf("\n (d) Base current value of 2.4 kV circuit , I_B = %d A \n",I_B1) ;
printf("\n     Base current value of 24 kV circuit , I_B = %d A \n",I_B2) ;
printf("\n     Base current value of 2.4 kV circuit , I_B = %d A \n",I_B3) ;
printf("\n (e) Physical current of 2.4 kV circuit , I = %.f A \n",I_1) ;
printf("\n     Physical current of 24 kV circuit , I = %.f A \n",I_2) ;
printf("\n     Physical current of 12 kV circuit , I = %.f A \n",I_4) ;
printf("\n (f) Per unit current values for all 3 circuits , I_pu = %.2f pu \n",I_pu_3ckt) ;
printf("\n (g) New transformer reactance of T1 , Z_pu_T1 =  j%.4f pu \n",abs(Z_pu_T1)) ;
printf("\n     New transformer reactance of T2 , Z_pu_T2 =  j%.4f pu \n",abs(Z_pu_T2)) ;
printf("\n (h) Per unit voltage value at bus 1 ,V1 = %.2f<%.1f pu \n",abs(V1),atand(imag(V1),real(V1))) ;
printf("\n     Per unit voltage value at bus 2 ,V2 = %.4f<%.1f pu \n",abs(V2),atand(imag(V2),real(V2))) ;
printf("\n     Per unit voltage value at bus 4 ,V4 = %.4f<%.1f pu \n",abs(V4),atand(imag(V4),real(V4))) ;
printf("\n (i) Per-unit apparent power value at bus 1 , S1 = %.2f pu \n",S1) ;
printf("\n     Per-unit apparent power value at bus 2 , S2 = %.4f pu \n",S2) ;
printf("\n     Per-unit apparent power value at bus 4 , S4 = %.4f pu \n",S4) ;
printf("\n (j) TABLE C.2 \n") ;
printf("\n     Results Of Example C.4 \n") ;
printf("\n ----------------------------------------------------------------------") ;
```

```
printf("\n      QUANTITY         \t   2.4-kV  circuit     \t   24-kV  circuit    \t   12-kV  circuit    ");
printf("\n ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- ") ;
printf("\n      kVA_B(3-    )      \t   %d kVA                \t   %d kVA                \t   %d kVA \n",kVA_B,kVA_B,kVA_B) ;
printf("\n      kV_B(L-L)         \t   %.1f  kV              \t   %d kV                 \t   %.1f  kV \n",kV_BL_L1,kV_BL_L2,kV_BL_L3) ;
printf("\n      Z_B               \t   %.3f                  \t   %.1f                  \t   %.1f      \n",Z_B1,Z_B2,Z_B3) ;
printf("\n      I_B               \t   %d  A                 \t   %d  A                 \t   %d  A \n",I_B1,I_B2,I_B3) ;
printf("\n      I_physical        \t   %d  A                 \t   %.f  A                \t   %.f  A \n",I_1,I_2,I_4) ;
printf("\n      I_pu              \t   %.2f  pu              \t   %.2f  pu              \t   %.2f  pu \n",I_pu_3ckt,I_pu_3ckt,I_pu_3ckt) ;
printf("\n      V_pu              \t   %.2f  pu              \t   %.4f  pu              \t   %.4f  pu \n",abs(V1),abs(V2),abs(V4)) ;
printf("\n      S_pu              \t   %.2f  pu              \t   %.4f  pu              \t   %.4f  pu \n",S1,S2,S4) ;
printf(" ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- ") ;
```

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING ANALYSIS AND DESIGN
```

```
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// APPENDIX C : REVIEW OF BASICS

// EXAMPLE : C.5 :
clear ; clc ; close ; // Clear the work space and console

// GIVEN DATA
D_ab = 6.8 ; // distance b/w conductors center-to-center in ft
D_bc = 5.5 ; // distance b/w conductors center-to-center in ft
D_ca = 4 ; // distance b/w conductors center-to-center in ft

// CALCULATIONS
// For case (a)
D_eq = (D_ab * D_bc * D_ca)^(1/3) ; // Equi spacing for pole top in ft
D_s = 0.01579 ; // GMR in ft From Table A.1
X_L = 0.1213 * log(D_eq/D_s) ; // Inductive reactance in  /mi . From equ C.135

// For case (b)
X_a = 0.503 ; // Inductive reactance in  /mi From Table A.1
X_d = 0.2026 ; // From Table A.8 for D_eq,by linear interpolation in  /mi
X_L1 = X_a + X_d ; // Inductive reactance in  /mi

// DISPLAY RESULTS
disp("EXAMPLE : C.5 : SOLUTION :-") ;
printf("\n (a) Inductive reactance using equation C.135 , X_L = %.4f  /mi \n",X_L );
printf("\n (b) Inductive reactance using tables ,
```

```
    X_L = %.4 f    /mi \n",X_L1) ;
```

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

```
// ELECTRIC POWER TRANSMISSION SYSTEM ENGINEERING
    ANALYSIS AND DESIGN
// TURAN GONEN
// CRC PRESS
// SECOND EDITION

// APPENDIX C : REVIEW OF BASICS

// EXAMPLE : C.6 :
clear ; clc ; close ; // Clear the work space and
    console

// GIVEN DATA
D_ab = 6.8 ; // distance b/w conductors center-to-
    center in ft
D_bc = 5.5 ; // distance b/w conductors center-to-
    center in ft
D_ca = 4 ; // distance b/w conductors center-to-
    center in ft
l = 100 ; // Line length in miles

// CALCULATIONS
// For case (a)
D_m = (D_ab * D_bc * D_ca)^(1/3) ; // Equi spacing
    for pole top in ft
r = 0.522/(2 * 12) ; // feet
X_C = 0.06836 * log10 (D_m/r) ; // Shunt capacitive
    reactance in M *mi

// For case (b)
```

```
X_a = 0.1136 ; // Shunt capacitive reactance in M *mi , From table A.1
X_d = 0.049543 ; // Shunt capacitive reactance spacing factor in M *mi , From table A.9
X_C1 = X_a + X_d ; // Shunt capacitive reactance in M *mi
X_C2 = X_C1/l ; // Capacitive reactance of 100 mi line in M

// DISPLAY RESULTS
disp("EXAMPLE : C.6 : SOLUTION :-") ;
printf("\n (a) Shunt capacitive reactance using equation C.156 , X_C = %.6f M *mi \n",X_C) ;
printf("\n (b) Shunt capacitive reactance using tables , X_C = %.6f M *mi \n",X_C1) ;
printf("\n (c) Capacitive reactance of total line , X_C = %.5e M \n",X_C2) ;
```