Contents

List of Scilab Codes

Chapter 2

THERMAL STATIONS

Scilab code Exa 2.1 Limiting value and Coal per hour

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 2: THERMAL STATIONS

// EXAMPLE : 2.1 :
// Page number 25-26
clear ; clc ; close ; // Clear the work space and
   console

//Given data
M = 15000.0+10.0           // Water evaporated(kg)
C = 5000.0+5.0             // Coal consumption(kg)
time = 8.0                 // Generation shift time(
   hours)

//Calculations
//Case(a)
```

```
M1 = M-15000.0
C1 = C-5000.0
M_C = M1/C1                // Limiting value of water evaporation(kg)
//Case(b)
kWh = 0                    // Station output at no load
consumption_noload = 5000+5*kWh         // Coal consumption at no load(kg)
consumption_noload_hr = consumption_noload/time       // Coal consumption per hour(kg)

//Results
disp("PART I - EXAMPLE : 2.1 : SOLUTION :-")
printf("\nCase(a): Limiting value of water evaporation per kg of coal consumed, M/C = %.f kg", M_C)
printf("\nCase(b): Coal per hour for running station at no load = %.f kg\n", consumption_noload_hr)
```

Scilab code Exa 2.2 `Average load on power plant`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 2: THERMAL STATIONS

// EXAMPLE : 2.2 :
// Page number 26
clear ; clc ; close ; // Clear the work space and console

```

```
//Given data
amount = 25.0*10**5          // Amount spent in 1 year(Rs)
value_heat = 5000.0          // Heating value(kcal/kg)
cost = 500.0                 // Cost of coal per ton(Rs)
n_ther = 0.35                // Thermal efficiency
n_elec = 0.9                 // Electrical efficiency

//Calculations
n = n_ther*n_elec                            // Overall efficiency
consumption = amount/cost*1000               // Coal consumption in 1 year(kg)
combustion = consumption*value_heat          // Heat of combustion(kcal)
output = n*combustion                        // Heat output(kcal)
unit_gen = output/860.0                      // Annual heat generated(kWh). 1 kWh = 860 kcal
hours_year = 365*24.0                        // Total time in a year(hour)
load_average = unit_gen/hours_year           // Average load on the power plant(kW)

//Result
disp("PART I - EXAMPLE : 2.2 : SOLUTION :-")
printf("\nAverage load on power plant = %.2f kW\n",load_average)
printf("\nNOTE: ERROR: Calculation mistake in the final answer in the textbook")
```

Scilab code Exa 2.3 `Heat balance sheet`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 2: THERMAL STATIONS

// EXAMPLE : 2.3 :
// Page number 26
clear ; clc ; close ; // Clear the work space and
    console

//Given data
consumption = 0.5       // Coal consumption per kWh
    output(kg)
cal_value = 5000.0      // Calorific value(kcal/kg)
n_boiler = 0.8          // Boiler efficiency
n_elec = 0.9            // Electrical efficiency

//Calculations
input_heat = consumption*cal_value
    // Heat input(kcal)
input_elec = input_heat/860.0
    // Equivalent electrical energy(kWh). 1 kWh = 860
     kcal
loss_boiler = input_elec*(1-n_boiler)
    // Boiler loss(kWh)
input_steam = input_elec-loss_boiler
    // Heat input to steam(kWh)
input_alter = 1/n_elec
    // Alternator input(kWh)
loss_alter = input_alter*(1-n_elec)
    // Alternate loss(kWh)
loss_turbine = input_steam-input_alter
    // Loss in turbine(kWh)
loss_total = loss_boiler+loss_alter+loss_turbine
    // Total loss(kWh)
```

```
output = 1.0
    // Output(kWh)
Input = output+loss_total
    // Input(kWh)

//Results
disp("PART I - EXAMPLE : 2.3 : SOLUTION :-")
printf("\nHeat Balance Sheet")
printf("\nLOSSES:  Boiler loss       = %.3f kWh",
    loss_boiler)
printf("\n         Alternator loss   = %.2f kWh",
    loss_alter)
printf("\n         Turbine loss      = %.3f kWh",
    loss_turbine)
printf("\n         Total loss        = %.2f kWh",
    loss_total)
printf("\nOUTPUT:  %.1f kWh", output)
printf("\nINPUT:   %.2f kWh\n", Input)
```

Chapter 3

HYDRO ELECTRIC STATIONS

Scilab code Exa 3.1 Firm capacity and Yearly gross output

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 3: HYDRO-ELECTRIC STATIONS

// EXAMPLE : 3.1 :
// Page number 41
clear ; clc ; close ; // Clear the work space and
    console

// Given data
Q = 95.0          // Minimum run-off(m^3/sec)
h = 40.0          // Head(m)

// Calculations
w = 1000.0                         // Density of water(kg/m
```

```
   ^3)
weight = Q*w                    // Weight of water per
   sec(kg)
work_done = weight*h            // Work done in one
   second(kg-mt)
kW_1 = 75.0/0.746               // 1 kW(kg-mt/sec)
power = work_done/kW_1          // Power production(kW)
hours_year = 365.0*24           // Total hours in a year
output = power*365*24.0         // Yearly gross output(
   kWhr)

// Results
disp("PART I - EXAMPLE : 3.1 : SOLUTION :-")
printf("\nFirm capacity = %.f kW", power)
printf("\nYearly gross output = %.2e kWhr.", output)
```

Scilab code Exa 3.3 Available continuous power

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 3: HYDRO-ELECTRIC STATIONS

// EXAMPLE : 3.3 :
// Page number 41
clear ; clc ; close ; // Clear the work space and
   console

// Given data
A = 200.0          // Catchment area(Sq.km)
F = 1000.0         // Annual rainfall(mm)
H = 200.0          // Effective head(m)
```

```
K = 0.5            // Yield factor
n = 0.8            // Plant efficiency

// Calculations
P = 3.14*n*K*A*F*H*10**-4     // Available continuous power(kW)

// Results
disp("PART I - EXAMPLE : 3.3 : SOLUTION :-")
printf("\nAvailable continuous power of hydro-electric station , P = %.f kW", P)
```

Scilab code Exa 3.4 `Minimum flow of river water to operate the plant`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 3: HYDRO-ELECTRIC STATIONS

// EXAMPLE : 3.4 :
// Page number 41-42
clear ; clc ; close ; // Clear the work space and console

// Given data
load_factor = 0.15     // Load factor
P = 10.0*10**3         // Rated installed capacity(kW)
H = 50.0               // Head of plant(m)
n = 0.8                // Efficiency of plant

//Calculation
```

```
units_day = P*load_factor        // Total units
    generated daily on basis of load factor(kWhr)
units_week = units_day*24.0*7    // Total units
    generated for one week(kWhr)
Q = units_week/(9.81*H*n*24*7)   // Minimum flow of
    water(cubic mt/sec)

//Result
disp("PART I - EXAMPLE : 3.4 : SOLUTION :-")
printf("\nMinimum flow of river water to operate the
     plant , Q = %.3f cubic mt/sec", Q)
```

Chapter 7

TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

Scilab code Exa 7.1 Demand factor and Load factor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.1 :
// Page number 73
clear ; clc ; close ; // Clear the work space and console

// Given data
connected_load = 450.0*10**3        // Connected load (kW)
```

```
maximum_demand = 250.0*10**3        // Maximum demand
    (kW)
units_generated = 615.0*10**6       // Units
    generated per annum(kWh)

// Calculations
// Case(i)
demand_factor = maximum_demand/connected_load
            // Demand factor
// Case(ii)
hours_year = 365.0*24
                                    // Total hours in
    a year
average_demand = units_generated/hours_year
              // Average demand(kW)
load_factor = average_demand/maximum_demand*100
          // Load factor(%)

// Results
disp("PART I - EXAMPLE : 7.1 : SOLUTION :-")
printf("\nCase(i) : Demand factor = %.3f ",
    demand_factor)
printf("\nCase(ii): Load factor = %.1f percent",
    load_factor)
```

Scilab code Exa 7.2 Total energy generated annually

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
   GENERATION
```

```
// EXAMPLE : 7.2 :
// Page number 73
clear ; clc ; close ; // Clear the work space and
    console

// Given data
maximum_demand = 480.0*10**3        // Maximum demand
    (kW)
LF = 0.4                            // Annual load
    factor

// Calculation
hours_year = 365.0*24                                  //
     Total hours in a year
energy_gen = maximum_demand*LF*hours_year              //
     Total energy generated annually(kWh)

// Results
disp("PART I - EXAMPLE : 7.2 : SOLUTION :-")
printf("\nTotal energy generated annually = %.5e kWh
    ", energy_gen)
```

Scilab code Exa 7.3 Annual load factors and Capacity factors of two power stations

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
    GENERATION

// EXAMPLE : 7.3 :
```

```
// Page number 73
clear ; clc ; close ; // Clear the work space and console

// Given data
cap_baseload = 400.0*10**3        // Installed capacity of base load plant(kW)
cap_standby = 50.0*10**3          // Installed capacity of standby unit(kW)
output_baseload = 101.0*10**6     // Annual baseload station output(kWh)
output_standby = 87.35*10**6      // Annual standby station output(kWh)
peakload_standby = 120.0*10**3    // Peak load on standby station(kW)
hours_use = 3000.0                // Hours of standby station use/year(hrs)

// Calculations
// Case(i)
LF_1 = output_standby*100/(peakload_standby*hours_use)      // Annual load factor(%)
hours_year = 365.0*24                                        // Total hours in a year
CF_1 = output_standby*100/(cap_standby*hours_year)           // Annual capacity factor(%)
// Case(ii)
peakload_baseload = peakload_standby                         // Peak load on baseload station(kW)
LF_2 = output_baseload*100/(peakload_baseload*hours_use)    // Annual load factor on baseload station(%)
hours_year = 365.0*24                                        // Total hours in a year
CF_2 = output_baseload*100/(cap_baseload*hours_year)
```

```
            // Annual capacity factor on baseload station(%)

// Results
disp("PART I - EXAMPLE : 7.3 : SOLUTION :-")
printf("\nCase(i) : Standby Station")
printf("\n          Annual load factor = %.2f percent", LF_1)
printf("\n          Annual capacity factor = %.2f percent\n", CF_1)
printf("\nCase(ii): Base load Station")
printf("\n          Annual load factor = %.2f percent", LF_2)
printf("\n          Annual capacity factor = %.2f percent\n", CF_2)
printf("\nNOTE: Incomplete solution in the textbook") ;
```

Scilab code Exa 7.4 Reserve capacity of plant

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.4 :
// Page number 74
clear ; clc ; close ; // Clear the work space and console

// Given data
```

```
MD = 500.0              // Maximum demand(MW)
LF = 0.5                // Annual load factor
CF = 0.4                // Annual capacity factor

// Calculations
hours_year = 365.0*24                       // Total
    hours in a year
energy_gen = MD*LF*hours_year               // Energy
    generated/annum(MWh)
plant_cap = energy_gen/(CF*hours_year) // Plant
    capacity(MW)
reserve_cap = plant_cap-MD                  // Reserve
    capacity of plant(MW)

// Results
disp("PART I - EXAMPLE : 7.4 : SOLUTION :-")
printf("\nReserve capacity of plant = %.f MW",
    reserve_cap)
```

Scilab code Exa 7.5 `Number of units supplied annually Diversity factor and Demand`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
   GENERATION

// EXAMPLE : 7.5 :
// Page number 74
clear ; clc ; close ; // Clear the work space and
    console

```

```
// Given data
load_1 = 150.0            // Load supplied by station(MW)
load_2 = 120.0            // Load supplied by station(MW)
load_3 = 85.0             // Load supplied by station(MW)
load_4 = 60.0             // Load supplied by station(MW)
load_5 = 5.0              // Load supplied by station(MW)
MD = 220.0                // Maximum demand(MW)
LF = 0.48                 // Annual load factor

// Calculations
// Case(a)
hours_year = 365.0*24                               // Total hours in a year
units = LF*MD*hours_year                            // Number of units supplied annually
// Case(b)
sum_demand = load_1+load_2+load_3+load_4+load_5     // Sum of maximum demand of individual consumers(MW)
diversity_factor = sum_demand/MD                    // Diversity factor
// Case(c)
DF = MD/sum_demand                                  // Demand factor

// Results
disp("PART I - EXAMPLE : 7.5 : SOLUTION :-")
printf("\nCase(a): Number of units supplied annually = %.2e units", units)
printf("\nCase(b): Diversity factor = %.3f ", diversity_factor)
printf("\nCase(c): Demand factor = %.3f = %.1f percent", DF,DF*100)
```

Scilab code Exa 7.6 Annual load factor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.6 :
// Page number 74
clear ; clc ; close ; // Clear the work space and console

// Given data
power_del_1 = 1000.0     // Power delivered by station(MW)
time_1 = 2.0             // Time for which power is delivered(hours)
power_del_2 = 500.0      // Power delivered by station(MW)
time_2 = 6.0             // Time for which power is delivered(hours)
days_maint = 60.0        // Maintenance days
max_gen_cap = 1000.0     // Maximum generating capacity(MW)

// Calculations
energy_sup_day = (power_del_1*time_1)+(power_del_2*time_2)   // Energy supplied for each working day(MWh)
days_total = 365.0
```

```
                                                          //
    Total days in a year
days_op = days_total-days_maint
                                             // Operating days of
     station in a year
energy_sup_year = energy_sup_day*days_op
                                    // Energy supplied per year(
    MWh)
hours_day = 24.0
                                                            //
    Total hours in a day
working_hours = days_op*hours_day
                                           // Hour of working in
    a year
LF = energy_sup_year*100/(max_gen_cap*working_hours)
                // Annual load factor(%)

// Results
disp("PART I - EXAMPLE : 7.6 : SOLUTION :-")
printf("\nAnnual load factor = %.1f percent", LF)
```

Scilab code Exa 7.7 Diversity factor and Annual load factor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
    GENERATION

// EXAMPLE : 7.7 :
// Page number 74
clear ; clc ; close ; // Clear the work space and
```

```
console

// Given data
load_industry = 750.0            // Industrial consumer load supplied by station(MW)
load_commercial = 350.0          // Commercial establishment load supplied by station(MW)
load_power = 10.0                // Domestic power load supplied by station(MW)
load_light = 50.0                // Domestic light load supplied by station(MW)
MD = 1000.0                      // Maximum demand(MW)
kWh_gen = 50.0*10**5             // Number of kWh generated per year

// Calculations
// Case(i)
sum_demand = load_industry+load_commercial+load_power+load_light   // Sum of max demand of individual consumers(MW)
diversity_factor = sum_demand/MD                    // Diversity factor
// Case(ii)
hours_year = 365.0*24                               // Total hours in a year
average_demand = kWh_gen/hours_year                 // Average demand(MW)
LF = average_demand/MD*100                          // Load factor(%)

// Results
disp("PART I - EXAMPLE : 7.7 : SOLUTION :-")
printf("\nCase(i) : Diversity factor = %.2f ", diversity_factor)
```

```
printf("\nCase(ii): Annual load factor = %.f percent", LF)
```

Scilab code Exa 7.8 Maximum demand and Connected load of each type

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.8 :
// Page number 74-75
clear ; clc ; close ; // Clear the work space and console

// Given data
load_domestic = 15000.0              // Domestic load supplied by station(kW)
diversity_domestic = 1.25            // Diversity factor of domestic load
DF_domestic = 0.7                    // Demand factor of domestic load
load_commercial = 25000.0            // Commercial load supplied by station(kW)
diversity_commercial = 1.2           // Diversity factor of commercial load
DF_commercial = 0.9                  // Demand factor of commercial load
load_industry = 50000.0              // Industrial load supplied by station(kW)
diversity_industry = 1.3             // Diversity
```

```
factor of industrial load
DF_industry = 0.98                    // Demand factor of industrial load
diversity_factor = 1.5                // Overall system diversity factor

// Calculations
// Case(a)
sum_demand = load_domestic+load_commercial+load_industry    // Sum of max demand of individual consumers(MW)
MD = sum_demand/diversity_factor                            // Maximum demand
// Case(b)
MD_domestic = load_domestic*diversity_domestic              // Maximum domestic load demand(kW)
connected_domestic = MD_domestic/DF_domestic                // Connected domestic load(kW)
MD_commercial = load_commercial*diversity_commercial        // Maximum commercial load demand(kW)
connected_commercial = MD_commercial/DF_commercial          // Connected commercial load(kW)
MD_industry = load_industry*diversity_industry              // Maximum industrial load demand(kW)
connected_industry = MD_industry/DF_industry                // Connected industrial load(kW)

// Results
disp("PART I - EXAMPLE : 7.8 : SOLUTION :-")
printf("\nCase(a): Maximum demand = %.f kW", MD)
printf("\nCase(b): Connected domestic load = %.1f kW", connected_domestic)
printf("\n         Connected commercial load = %.1f kW", connected_commercial)
printf("\n         Connected industrial load = %.1f kW", connected_industry)
```

Scilab code Exa 7.9 Size and number of generator units Reserve plant capacity Load

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
   GENERATION

// EXAMPLE : 7.9 :
// Page number 75-76
clear ; clc ; close ; // Clear the work space and
   console

// Given data
MD = 10000.0          // Maximum demand(kW)
load_1 = 2000.0       // Load from 11 PM-6 AM(kW)
t_1 = 7.0             // Time from 11 PM-6 AM(hour)
load_2 = 3500.0       // Load from 6 AM-8 AM(kW)
t_2 = 2.0             // Time from 6 AM-8 AM(hour)
load_3 = 8000.0       // Load from 8 AM-12 Noon(kW)
t_3 = 4.0             // Time from 8 AM-12 Noon(hour)
load_4 = 3000.0       // Load from 12 Noon-1 PM(kW)
t_4 = 1.0             // Time from 12 Noon-1 PM(hour)
load_5 = 7500.0       // Load from 1 PM-5 PM(kW)
t_5 = 4.0             // Time from 1 PM-5 PM(hour)
load_6 = 8500.0       // Load from 5 PM-7 PM(kW)
t_6 = 2.0             // Time from 5 PM-7 PM(hour)
load_7 = 10000.0      // Load from 7 PM-9 PM(kW)
t_7 = 2.0             // Time from 7 PM-9 PM(hour)
load_8 = 4500.0       // Load from 9 PM-11 PM(kW)
t_8 = 2.0             // Time from 9 PM-11 PM(hour)
```

```

// Calculations
energy_gen = (load_1*t_1)+(load_2*t_2)+(load_3*t_3)+(load_4*t_4)+(load_5*t_5)+(load_6*t_6)+(load_7*t_7)+(load_8*t_8) // Energy generated during 24 hours(kWh)
LF = energy_gen/(MD*24.0)                                   // Load factor
no_units = 3.0                                              // Number of generating set
cap_1 = 5000.0                                              // Capacity of first generating unit(kW)
cap_2 = 3000.0                                              // Capacity of second generating unit(kW)
cap_3 = 2000.0                                              // Capacity of third generating unit(kW)
cap_reserve = cap_1                                         // Reserve capacity(kW) i.e largest size of generating unit
cap_installed = cap_1+cap_2+cap_3+cap_reserve               // Installed capacity(kW)
cap_factor = energy_gen/(cap_installed*24.0)                // Plant capacity factor
cap_plant = cap_3*t_1+(cap_3+cap_2)*t_2+(cap_2+cap_1)*t_3+cap_2*t_4+(cap_2+cap_1)*t_5+(cap_3+cap_2+cap_1)*t_6+(cap_3+cap_2+cap_1)*t_7+cap_1*t_8 // Capacity of plant running actually(kWh)
use_factor = energy_gen/cap_plant                           // Plant use factor

// Results
disp("PART I - EXAMPLE : 7.9 : SOLUTION :-")
printf("\nNumber of generator units = %.f", no_units)
```

```
printf("\nSize of generator units required are %.f kW, %.f kW and %.f kW", cap_1,cap_2,cap_3)
printf("\nReserve plant capacity = %.f kW", cap_reserve)
printf("\nLoad factor = %.2f = %.f percent", LF,LF*100)
printf("\nPlant capacity factor = %.4f = %.2f percent", cap_factor,cap_factor*100)
printf("\nPlant use factor = %.3f = %.1f percent", use_factor,use_factor*100)
printf("\n\nNOTE: Capacity of plant is directly taken & operating schedule is not displayed here")
```

Scilab code Exa 7.10 `Cost of generation per kWh at 100 and 50 percent load factor`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.10 :
// Page number 76
clear ; clc ; close ; // Clear the work space and console

// Given data
cap_installed = 210.0*10**3        // Installed capacity of the station(kW)
capital_cost_kW = 1000.0           // Capital cost of station(Rs/kW)
```

```
fixed_cost_per = 0.13                 // Fixed cost = 13% * cost of investment
variable_cost_per = 1.3               // Variable cost = 1.3*fixed cost
LF_1 = 1.0                            // Load factor
LF_2 = 0.5                            // Load factor

// Calculations
MD = cap_installed                                        // Maximum demand(kW)
hours_year = 365.0*24                                     // Total hours in a year
capital_cost = capital_cost_kW*cap_installed              // Capital cost of station(Rs)
// Case(i) At 100% load factor
fixed_cost_1 = capital_cost*fixed_cost_per                // Fixed cost(Rs)
variable_cost_1 = variable_cost_per*fixed_cost_1          // Variable cost(Rs)
operating_cost_1 = fixed_cost_1+variable_cost_1           // Operating cost per annum(Rs)
units_gen_1 = LF_1*MD*hours_year                          // Total units generated(kWh)
cost_gen_1 = operating_cost_1*100/units_gen_1             // Cost of generation per kWh(Paise)
// Case(ii) At 50% load factor
fixed_cost_2 = capital_cost*fixed_cost_per                // Fixed cost(Rs)
units_gen_2 = LF_2*MD*hours_year                          // Total units generated(kWh)
variable_cost_2 = variable_cost_1*units_gen_2/units_gen_1    // Variable cost(Rs)
operating_cost_2 = fixed_cost_2+variable_cost_2
```

```
                    // Operating cost per annum(Rs)
cost_gen_2 = operating_cost_2*100/units_gen_2
                    // Cost of generation per kWh(Paise
    )

// Results
disp("PART I - EXAMPLE : 7.10 : SOLUTION :-")
printf("\nCost of generation per kWh at 100 percent
    load factor = %.2f paise", cost_gen_1)
printf("\nCost of generation per kWh at 50 percent
    load factor = %.1f paise", cost_gen_2)
printf("\nComment: As the load factor is reduced,
    cost of generation is increased\n")
printf("\nNOTE: ERROR: (1) In problem statement,
    Capital cost of station must be Rs. 1000/kW, not
    Rs. 1000/MW")
printf("\n              (2) Calculation mistake in
    Total units generated in Case(i) in textbook")
```

Scilab code Exa 7.11 `Cost per unit generated`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
    GENERATION

// EXAMPLE : 7.11 :
// Page number 76
clear ; clc ; close ; // Clear the work space and
    console

```

```
// Given data
MD = 100.0*10**3                          // Maximum demand(kW)
capital_cost = 200.0*10**6                // Capital cost(Rs)
LF = 0.4                                  // Annual load factor
cost_fueloil = 15.0*10**6                 // Annual cost of fuel and oil(Rs)
cost_tax = 10.0*10**6                     // Cost of taxes, wages and salaries(Rs)
interest = 0.15                           // Interest and depreciation

// Calculations
hours_year = 365.0*24                     // Total hours in a year
units_gen = MD*LF*hours_year              // Units generated per annum(kWh)
fixed_charge = interest*capital_cost      // Annual fixed charges(Rs)
running_charge = cost_fueloil+cost_tax    // Annual running charges(Rs)
annual_charge = fixed_charge+running_charge    // Total annual charges(Rs)
cost_unit = annual_charge*100/units_gen   // Cost per unit(Paise)

// Results
disp("PART I - EXAMPLE : 7.11 : SOLUTION :-")
printf("\nCost per unit generated = %.f paise", cost_unit)
```

Scilab code Exa 7.12 Minimum reserve capacity of station and Cost per kWh generate

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
   GENERATION

// EXAMPLE : 7.12 :
// Page number 76-77
clear ; clc ; close ; // Clear the work space and
   console

// Given data
cap_installed = 500.0           // Installed
   capacity of the station(MW)
CF = 0.45                       // Capacity factor
LF = 0.6                        // Annual laod
   factor
cost_fueloil = 10.0*10**7       // Annual cost of
   fuel,oil etc(Rs)
capital_cost = 10**9            // Capital cost(Rs)
interest = 0.15                 // Interest and
   depreciation

// Calculations
// Case(i)
MD = cap_installed*CF/LF
                                      // Maximum
   demand(MW)
cap_reserve = cap_installed-MD
                                // Reserve capacity(
   MW)
// Case(ii)
hours_year = 365.0*24
                                        // Total
   hours in a year
```

```
units_gen = MD*10**3*LF*hours_year                        // Units generated per annum(kWh)
fixed_charge = interest*capital_cost                      // Annual fixed charges(Rs)
running_charge = cost_fueloil                             // Annual running charges(Rs)
annual_charge = fixed_charge+running_charge               // Total annual charges(Rs)
cost_unit = annual_charge*100/units_gen                   // Cost per kWh generated(Paise)

// Results
disp("PART I - EXAMPLE : 7.12 : SOLUTION :-")
printf("\nCase(i) : Minimum reserve capacity of station = %.f MW", cap_reserve)
printf("\nCase(ii): Cost per kWh generated = %.f paise", cost_unit)
```

Scilab code Exa 7.13 Two part tariff to be charged from consumers

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.13 :
// Page number 77
```

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

// Given data
gen_expense = 850000.0                  // Annual generation expense(Rs)
fuel_expense = 2800000.0                // Annual fuel expense(Rs)
trans_expense = 345000.0                // Annual transmission expense(Rs)
dist_expense = 2750000.0                // Annual distribution expense(Rs)
repair_expense = 300000.0               // Annual repairs,etc expense(Rs)
unit_gen = 600.0*10**6                  // Number of units generated per year(kWh)
MD = 75.0*10**3                         // Maximum demand(kW)
gen = 0.9                               // Fixed charges for generation
fuel = 0.15                             // Fixed charges for fuel
transm = 0.85                           // Fixed charges for transmission
dist = 0.95                             // Fixed charges for distribution
repair = 0.5                            // Fixed charges for repairs,etc
loss_dist = 0.2                         // Losses in transmission and distribution

// Calculations
fixed_gen = gen_expense*gen                     // Fixed charge on generation(Rs)
running_gen = gen_expense*(1-gen)               // Running charge on generation(Rs)
fixed_fuel = fuel_expense*fuel                  // Fixed charge on fuel(Rs)
```

```
running_fuel = fuel_expense*(1-fuel)            // Running charge on fuel(Rs)
fixed_trans = trans_expense*transm              // Fixed charge on transmission(Rs)
running_trans = trans_expense*(1-transm)        // Running charge on transmission(Rs)
fixed_dist = dist_expense*dist                  // Fixed charge on distribution(Rs)
running_dist = dist_expense*(1-dist)            // Running charge on distribution(Rs)
fixed_repair = repair_expense*repair            // Fixed charge on repairs,etc(Rs)
running_repair = repair_expense*(1-repair)      // Running charge on repairs,etc(Rs)
fixed_charge = fixed_gen+fixed_fuel+fixed_trans+fixed_dist+fixed_repair       // Total fixed charges(Rs)
running_charge = running_gen+running_fuel+running_trans+running_dist+running_repair   // Total running charges(Rs)
fixed_unit = fixed_charge/MD                    // Fixed charges per unit(Rs)
units_dist = unit_gen*(1-loss_dist)             // Total number of units distributed(kWh)
running_unit = running_charge*100/units_dist    // Running charges per unit(Paise)

// Results
disp("PART I - EXAMPLE : 7.13 : SOLUTION :-")
printf("\nTwo part tariff is Rs %.3f per kW of maximum demand plus %.3f paise per kWh",fixed_unit,running_unit)
```

Scilab code Exa 7.14 Generation cost in two part form

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
   GENERATION

// EXAMPLE : 7.14 :
// Page number 77
clear ; clc ; close ; // Clear the work space and
   console

// Given data
cap_installed = 100.0*10**3          // Installed
   capacity of the station(kW)
capital_cost_kW = 1000.0             // Capital
   cost(Rs/kW)
depreciation = 0.15                  // Annual
   depreciation charge
royalty_kW = 2.0                     // Royalty per
    kW per year(Rs)
royalty_kWh = 0.03                   // Royalty per
    kWh per year(Rs)
MD = 70.0*10**3                      // Maximum
   demand(kW)
LF = 0.6                             // Annual load
    factor
cost_salary = 1000000.0              // Annual cost
    of salaries,maintenance charges etc(Rs)
cost_salary_per = 0.2                // Annual cost
    of salaries,maintenance charges etc charged as
   fixed charges

// Calculations
hours_year = 365.0*24
                                                    //
```

```
Total hours in a year
unit_gen = MD*LF*hours_year                                // Units generated/annum(kWh)
capital_cost = cap_installed*capital_cost_kW               // Capital cost of plant(Rs)
depreciation_charge = depreciation*capital_cost            // Depreciation charges(Rs)
salary_charge = cost_salary_per*cost_salary                // Cost on salaries , maintenance etc(Rs)
fixed_charge = depreciation_charge+salary_charge           // Total annual fixed charges(Rs)
cost_kW_fixed = (fixed_charge/MD)+royalty_kW               // Cost per kW(Rs)
salary_charge_running = (1-cost_salary_per)*cost_salary    // Annual running charge on salaries , maintenance etc(Rs)
cost_kWh_running = (salary_charge_running/unit_gen)+royalty_kWh   // Cost per kWh(Rs)

// Results
disp("PART I - EXAMPLE : 7.14 : SOLUTION :-")
printf("\nGeneration cost in two part form is given by, Rs. (%.2f*kW + %.3f*kWh) ", cost_kW_fixed, cost_kWh_running)
```

Scilab code Exa 7.15 Overall generating cost per unit at 50 and 100 percent capaci

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
```

```
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.15 :
// Page number 78
clear ; clc ; close ; // Clear the work space and console

// Given data
cap_installed = 100.0*10**3  // Installed capacity of station(kW)
cost_gen = 30.0              // Generating cost per annum(Rs/kW)
cost_fixed = 4000000.0       // Fixed cost per annum(Rs)
cost_fuel = 60.0             // Cost of fuel(Rs/tonne)
calorific = 5700.0           // Calorific value of fuel(kcal/kg)
rate_heat_1 = 2900.0         // Plant heat rate at 100% capacity factor(kcal/kWh)
CF_1 = 1.0                   // Capacity factor
rate_heat_2 = 4050.0         // Plant heat rate at 50% capacity factor(kcal/kWh)
CF_2 = 0.5                   // Capacity factor

// Calculations
cost_fixed_kW = cost_fixed/cap_installed                // Fixed cost per kW(Rs)
cost_fixed_total = cost_gen+cost_fixed_kW               // Fixed cost per kW capacity(Rs)
average_demand_1 = CF_1*cap_installed                   // Average demand at 100% capacity factor(kW)
average_demand_2 = CF_2*cap_installed                   // Average demand at 50% capacity factor(kW)
hours_year = 365.0*24                                   // Total hours in a year
unit_gen_1 = CF_1*hours_year
```

```
// Energy generated per annum with average demand of 1 kW(kWh)
unit_gen_2 = CF_2*hours_year    // Energy generated per annum with average demand of 0.5 kW(kWh)
cost_kWh_fixed_1 = cost_fixed_total*100/unit_gen_1    // Cost per kWh due to fixed charge with 100% CF(Paise)
cost_kWh_fixed_2 = cost_fixed_total*100/unit_gen_2    // Cost per kWh due to fixed charge with 50% CF(Paise)
kg_kWh_1 = rate_heat_1/calorific    // Weight(kg)
kg_kWh_2 = rate_heat_2/calorific    // Weight(kg)
cost_coal_1 = kg_kWh_1*cost_fuel*100/1000.0    // Cost due to coal at 100% CF(Paise/kWh)
cost_coal_2 = kg_kWh_2*cost_fuel*100/1000.0    // Cost due to coal at 50% CF(Paise/kWh)
cost_total_1 = cost_kWh_fixed_1+cost_coal_1    // Total cost per unit with 100% CF(Paise)
cost_total_2 = cost_kWh_fixed_2+cost_coal_2    // Total cost per unit with 50% CF(Paise)

// Results
disp("PART I - EXAMPLE : 7.15 : SOLUTION :-")
printf("\nOverall generating cost per unit at 100 percent capacity factor = %.3f paise", cost_total_1)
printf("\nOverall generating cost per unit at 50 percent capacity factor = %.3f paise\n", cost_total_2)
printf("\nNOTE: Slight changes in obtained answer from that of textbook answer is due to more precision here")
```

Scilab code Exa 7.16 Yearly cost per kW demand and Cost per kWh supplied at substa

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.16 :
// Page number 78
clear ; clc ; close ; // Clear the work space and console

// Given data
MD = 75.0*10**3                       // Maximum demand(kW)
LF = 0.4                              // Yearly load factor
cost_capital = 60.0                   // Capital cost(Rs/annum/kW)
cost_kWh = 1.0                        // Cost per kWh transmitted(Paise)
charge_trans = 2000000.0              // Annual capital charge for transmission(Rs)
charge_dist = 1500000.0               // Annual capital charge for distribution(Rs)
diversity_trans = 1.2                 // Diversity factor for transmission
diversity_dist = 1.25                 // Diversity factor for distribution
n_trans = 0.9                         // Efficiency
```

```
of transmission system
n_dist = 0.85                                    // Efficiency of distribution system

// Calculations
// Case(a)
capital_cost = cost_capital*MD                   // Annual capital cost(Rs)
fixed_charge_sub = capital_cost+charge_trans     // Total fixed charges for supply to substation per annum(Rs)
sum_MD_sub = MD*diversity_trans                  // Sum of all maximum demand of substation(kW)
cost_kW_sub = fixed_charge_sub/sum_MD_sub        // Yearly cost per kW demand at substation(Rs)
running_cost_unit_sub = 1/n_trans                // Running cost per unit supplied at substation(Paise)
// Case(b)
sum_MD_con = sum_MD_sub*diversity_dist           // Sum of all maximum demand of consumer(kW)
fixed_charge_con = capital_cost+charge_trans+charge_dist    // Total fixed charges for supply to cosnumers(Rs)
cost_kW_con = fixed_charge_con/sum_MD_con        // Yearly cost per kW demand on consumer premises(Rs)
running_cost_unit_con = running_cost_unit_sub/n_dist    // Running cost per unit supplied to consumer(Paise)

// Results
disp("PART I - EXAMPLE : 7.16 : SOLUTION :-")
printf("\nCase(a): Yearly cost per kW demand at the
```

```
substations = Rs. %.2f ", cost_kW_sub)
printf("\n          Cost per kWh supplied at the substations = %.2f paise\n", running_cost_unit_sub)
printf("\nCase(b): Yearly cost per kW demand at the consumer premises = Rs. %.2f ", cost_kW_con)
printf("\n          Cost per kWh supplied at the consumer premises = %.3f paise", running_cost_unit_con)
```

Scilab code Exa 7.17 `Number of working hours per week above which the HV supply is`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.17 :
// Page number 79
clear ; clc ; close ; // Clear the work space and console

// Given data
kVA_tariff_hv = 60.0          // HV supply per kVA per annum(Rs)
kWh_tariff_hv = 3.0/100       // HV supply per kWh annum(Rs)
kVA_tariff_lv = 65.0          // LV supply per kVA per annum(Rs)
kWh_tariff_lv = 3.3/100       // LV supply per kWh annum(Rs)
```

```
cost_equip_kVA = 50.0        // Cost of transformers and switchgear per kVA(Rs)
loss_full_load = 0.02        // Full load transformation loss
fixed_charge_per = 0.2       // Fixed charges per annum
no_week = 50.0               // Number of working weeks in a year

// Calculations
rating_equip = 1000/(1-loss_full_load)          // Rating of transformer and switchgear(kVA)
cost_equip = cost_equip_kVA*rating_equip        // Cost of transformers and switchgear(Rs)
fixed_charge = fixed_charge_per*cost_equip      // Fixed charges per annum on HV plant(Rs)
X = poly(0,"X")                                 // Number of working hours per week
units_consumed = (no_week*X)*1000.0             // Yearly units consumed by load
total_units = units_consumed/(1-loss_full_load) // Total units to be paid on HV supply
// Case(a)
annual_cost_hv = (kVA_tariff_hv*rating_equip)+(kWh_tariff_hv*cost_equip*X)+fixed_charge   // Annual cost(Rs)
// Case(b)
annual_cost_lv = (kVA_tariff_lv*1000.0)+(kWh_tariff_lv*units_consumed)     // Annual cost(Rs)
p = annual_cost_hv-annual_cost_lv                                       // Finding unknown value i.e working hours in terms of X
x = roots(p)                                    // Finding unknown value i.e working hours

// Results
```

```
disp("PART I - EXAMPLE : 7.17 : SOLUTION :-")
printf("\nAbove %.1f working hours per week the H.V
    supply is cheaper ", x)
```

Scilab code Exa 7.18 Cheaper alternative to adopt and by how much

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
    GENERATION

// EXAMPLE : 7.18 :
// Page number 79-80
clear ; clc ; close ; // Clear the work space and
    console

// Given data
load_1 = 10.0*10**3          // Load per annum(kVA)
time_1 = 1800.0              // Time(hours)
load_2 = 6.0*10**3           // Load per annum(kVA)
time_2 = 600.0               // Time(hours)
load_3 = 0.25*10**3          // Load per annum(kVA)
time_3 = 400.0               // Time(hours)
rating_trans = 10.0*10**3    // Transformer rating(kVA
    )
pf = 0.8                     // Lagging power factor
n_fl_A = 98.3/100.0          // Full load efficiency
    of transformer A
n_fl_B = 98.8/100.0          // Full load efficiency
    of transformer B
loss_A = 70.0                // Core loss at rated
```

```
    voltage of transformer A(kW)
loss_B = 40.0                // Core loss at rated
    voltage of transformer B(kW)
cost_A = 250000.0            // Cost of transformer A(
    Rs)
cost_B = 280000.0            // Cost of transformer B(
    Rs)
interest_per = 0.1           // Interest and
    depreciation charges
cost_energy_unit = 3.0       // Energy costs per unit(
    Paise)

// Calculations
// Transformer A
output_A = rating_trans*pf
                                 // kW output at full
    load(kW)
input_A = output_A/n_fl_A
                                  // Input at full
    load(kW)
cu_loss_fl_A = input_A-output_A-loss_A
                     // Copper loss at full load(kW)
cu_loss_2_A = (load_2/load_1)**2*cu_loss_fl_A
              // Copper loss at 6 MVA output(kW)
cu_loss_3_A = (load_3/load_1)**2*cu_loss_fl_A
              // Copper loss at 0.25 MVA output(kW)
ene_iron_loss_A = loss_A*(time_1+time_2+time_3)
          // Energy consumed due to iron losses(kWh)
ene_cu_loss_A = time_1*cu_loss_fl_A+time_2*
    cu_loss_2_A+time_3*cu_loss_3_A  // Energy
    consumed due to copper losses(kWh)
total_loss_A = ene_iron_loss_A+ene_cu_loss_A
               // Total loss per annum(kWh)
cost_energy_A = cost_energy_unit/100*total_loss_A
        // Energy cost per annum due to losses(Rs)
// Transformer B
output_B = rating_trans*pf
                                 // kW output at full
```

```
load(kW)
input_B = output_B/n_fl_B                                 // Input at full load(kW)
cu_loss_fl_B = input_B-output_B-loss_B                    // Copper loss at full load(kW)
cu_loss_2_B = (load_2/load_1)**2*cu_loss_fl_B             // Copper loss at 6 MVA output(kW)
cu_loss_3_B = (load_3/load_1)**2*cu_loss_fl_B             // Copper loss at 0.25 MVA output(kW)
ene_iron_loss_B = loss_B*(time_1+time_2+time_3)           // Energy consumed due to iron losses(kWh)
ene_cu_loss_B = time_1*cu_loss_fl_B+time_2*
    cu_loss_2_B+time_3*cu_loss_3_B   // Energy
    consumed due to copper losses(kWh)
total_loss_B = ene_iron_loss_B+ene_cu_loss_B              // Total loss per annum(kWh)
cost_energy_B = cost_energy_unit/100*total_loss_B         // Energy cost per annum due to losses(Rs)
diff_capital = cost_B-cost_A                              // Difference in capital costs(Rs)
annual_charge = interest_per*diff_capital                 // Annual charge due to this amount(Rs)
diff_cost_energy = cost_energy_A-cost_energy_B            // Difference in energy cost per annum(Rs)
cheap = diff_cost_energy-annual_charge                    // Cheaper in cost(Rs)

// Results
disp("PART I - EXAMPLE : 7.18 : SOLUTION :-")
printf("\nTransformer B is cheaper by Rs. %.f per
    year \n", cheap)
printf("\nNOTE: ERROR: Full load efficiency for
    transformer B is 98.8 percent, not 98.3 percent
    as given in problem statement")
```

```
printf("\n      Changes in obtained answer from that
    of textbook answer is due to more precision")
```

Scilab code Exa 7.19 Valuation halfway based on Straight line Reducing balance and

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
   GENERATION

// EXAMPLE : 7.19 :
// Page number 80-81
clear ; clc ; close ; // Clear the work space and
   console

// Given data
fixed_cost = 4.0*10**4          // Fixed cost of
   plant(Rs)
salvage_value = 4.0*10**3       // Salvage value(Rs)
n = 20.0                        // Useful life(years)
r = 0.06                        // Sinking fund
   depreciation compounded annually

// Calculations
n_2 = n/2                                     //
   Halfway of useful life(years)
// Case(a)
total_dep_A = fixed_cost-salvage_value        //
   Total depreciation in 20 years(Rs)
dep_10_A = total_dep_A/2                      //
   Depreciation in 10 years(Rs)
```

```
value_10_A = fixed_cost-dep_10_A                    // Value at the end of 10 years(Rs)
// Case(b)
P_B = fixed_cost                                    // Capital outlay(Rs)
q_B = (salvage_value/fixed_cost)**(1/n)             // q = (1-p)
value_10_B = P_B*(q_B)**n_2                         // Value at the end of 10 years(Rs)
// Case(c)
P_C = fixed_cost                                    // Capital cost of plant(Rs)
P__C = salvage_value                                // Scrap value(Rs)
Q_C = P_C-P__C                                      // Cost of replacement(Rs)
q_C = Q_C/((((1+r)**n-1)/r)                          // Yearly charge(Rs)
amount_dep = q_C*((1+r)**n_2-1)/r                   // Amount deposited at end of 10 years(Rs)
value_10_C = P_C-amount_dep                         // Value at the end of 10 years(Rs)

// Results
disp("PART I - EXAMPLE : 7.19 : SOLUTION :-")
printf("\nCase(a): Valuation halfway through its life based on Straight line depreciation method = Rs %.1e ", value_10_A)
printf("\nCase(b): Valuation halfway through its life based on Reducing balance depreciation method = Rs %.2e ", value_10_B)
printf("\nCase(c): Valuation halfway through its life based on Sinking fund depreciation method = Rs %.2e ", value_10_C)
```

Scilab code Exa 7.20 `Type and hp ratings of two turbines for the station`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.20 :
// Page number 81
clear ; clc ; close ; // Clear the work space and console

// Given data
h = 30.0                    // Mean head(m)
area_catch = 250.0          // Catchment area(Square km)
average_rain = 1.25         // Average rainfall per annum(m)
utilized_rain = 0.7         // Rainfall utilized
LF = 0.8                    // Expected load factor
n_turbine = 0.9             // Mechanical efficiency of turbine
n_gen = 0.95                // Efficiency of generator

// Calculations
water_avail = utilized_rain*area_catch*10**6*average_rain     // Water available(m^3)
sec_year = 365.0*24*60*60                                     // Total seconds in a year
Q = water_avail/sec_year                                      // Quantity available per second(m^3) i.e Discharge(m^3/sec)
w = 1000.0
```

```
// Density of water(kg/m^3)
n = n_turbine*n_gen                                                    // Overall efficiency
P = 0.736/75*Q*w*h*n                                                   // Average output of generator units(kW)
rating_gen = P/LF                                                      // Rating of generator(kW)
rating_gen_each = rating_gen/2.0                                       // Rating of each generator(kW)
rating_turbine = rating_gen/2*(1/(0.736*n_gen))                        // Rating of each turbine(metric hp)

// Results
disp("PART I - EXAMPLE : 7.20 : SOLUTION :-")
printf("\nChoice of units are:")
printf("\n  2 generators each having maximum rating of %.f kW ", rating_gen_each)
printf("\n  2 propeller turbines each having maximum rating of %.f metric hp \n", rating_turbine)
printf("\nNOTE: Changes in obtained answer from that of textbook answer is due to more precision here')
```

Scilab code Exa 7.21 Plot of chronological load curve and Load duration curve

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
```

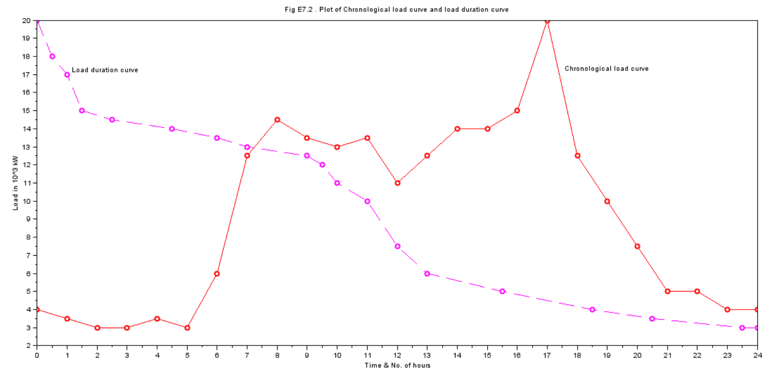

Figure 7.1: Plot of chronological load curve and Load duration curve

```
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
    GENERATION

// EXAMPLE : 7.21 :
// Page number 81-82
clear ; clc ; close ; // Clear the work space and
    console

// Given data
t0 = 0.0                          // Time 12 morning
l0 = 4.0                          // Load at 12 morning(kW
    *1000)
t1 = 1.0                          // Time 1 a.m
l1 = 3.5                          // Load at 1 a.m(kW*1000)
t2 = 2.0                          // Time 2 a.m
l2 = 3.0                          // Load at 2 a.m(kW*1000)
t3 = 3.0                          // Time 3 a.m
l3 = 3.0                          // Load at 3 a.m(kW*1000)
t4 = 4.0                          // Time 4 a.m
```

```
l4 = 3.5                    // Load at 4 a.m(kW*1000)
t5 = 5.0                    // Time 5 a.m
l5 = 3.0                    // Load at 5 a.m(kW*1000)
t6 = 6.0                    // Time 6 a.m
l6 = 6.0                    // Load at 6 a.m(kW*1000)
t7 = 7.0                    // Time 7 a.m
l7 = 12.5                   // Load at 7 a.m(kW*1000)
t8 = 8.0                    // Time 8 a.m
l8 = 14.5                   // Load at 8 a.m(kW*1000)
t9 = 9.0                    // Time 9 a.m
l9 = 13.5                   // Load at 9 a.m(kW*1000)
t10 = 10.0                  // Time 10 a.m
l10 = 13.0                  // Load at 10 a.m(kW*1000)
t11 = 11.0                  // Time 11 a.m
l11 = 13.5                  // Load at 11 a.m(kW*1000)
t113 = 11.50                // Time 11.30 a.m
l113 = 12.0                 // Load at 11.30 am(kW
    *1000)
t12 = 12.0                  // Time 12 noon
l12 = 11.0                  // Load at 12 noon(kW*1000)
t123 = 12.50                // Time 12.30 noon
l123 = 5.0                  // Load at 12.30 noon(kW
    *1000)
t13 = 13.0                  // Time 1 p.m
l13 = 12.5                  // Load at 1 p.m(kW*1000)
t133 = 13.50                // Time 1.30 p.m
l133 = 13.5                 // Load at 1.30 p.m(kW
    *1000)
t14 = 14.0                  // Time 2 p.m
l14 = 14.0                  // Load at 2 p.m(kW*1000)
t15 = 15.0                  // Time 3 p.m
l15 = 14.0                  // Load at 3 p.m(kW*1000)
t16 = 16.0                  // Time 4 p.m
l16 = 15.0                  // Load at 4 p.m(kW*1000)
t163 = 16.50                // Time 4.30 p.m
l163 = 18.0                 // Load at 4.30 p.m(kW
    *1000)
t17 = 17.0                  // Time 5 p.m
```

```
l17 = 20.0                      // Load at 5 p.m(kW*1000)
t173 = 17.50                    // Time 5.30 p.m
l173 = 17.0                     // Load at 5.30 p.m(kW*1000)
t18 = 18.0                      // Time 6 p.m
l18 = 12.5                      // Load at 6 p.m(kW*1000)
t19 = 19.0                      // Time 7 p.m
l19 = 10.0                      // Load at 7 p.m(kW*1000)
t20 = 20.0                      // Time 8 p.m
l20 = 7.5                       // Load at 8 p.m(kW*1000)
t21 = 21.0                      // Time 9 p.m
l21 = 5.0                       // Load at 9 p.m(kW*1000)
t22 = 22.0                      // Time 10 p.m
l22 = 5.0                       // Load at 10 p.m(kW*1000)
t23 = 23.0                      // Time 11 p.m
l23 = 4.0                       // Load at 11 p.m(kW*1000)
t24 = 24.0                      // Time 12 morning
l24 = 4.0                       // Load at 12 morning(kW*1000)

// Calculations
t = [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,t16,t17,t18,t19,t20,t21,t22,t23,t24]
l = [l0,l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16,l17,l18,l19,l20,l21,l22,l23,l24]
a = gca() ;
a.thickness = 2                 // sets thickness of plot
plot(t,l,'ro-')                 // Plot of Chronological load curve
T = [0,0.5,1,1.5,2.5,4.5,6,7,9,9.5,10,11,12,13,15.5,18.5,20.5,23.5,24]    // Solved time
L = [20,18,17,15,14.5,14,13.5,13,12.5,12,11,10,7.5,6,5,4,3.5,3,3]         // Solved load
plot(T,L,'--mo')                // Plot of load duration curve
```

```
a.x_label.text = 'Time & No. of hours'   // labels x-axis
a.y_label.text = 'Load in 10^3 kW'   // labels y-axis
xtitle("Fig E7.2 . Plot of Chronological load curve and load duration curve")
xset('thickness',2)   // sets thickness of axes
xstring(17.5,17,'Chronological load curve')
xstring(1.1,17,'Load duration curve')

// Results
disp("PART I - EXAMPLE : 7.21 : SOLUTION :-")
printf("\nThe chronological load curve and the load duration curve is shown in the Figure E7.2\n")
printf("\nNOTE: The time is plotted in 24 hours format')
```

Scilab code Exa 7.22 Daily energy produced Reserve capacity and Maximum energy pro

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.22 :
// Page number 82
clear ; clc ; close ; // Clear the work space and console

// Given data
```

```
MD = 20.0*10**3                         // Maximum demand(kW)
LF = 0.6                                // Load factor
CF = 0.48                               // Plant capacity factor
UF = 0.8                                // Plant use factor

// Calculations
// Case(a)
avg_demand = LF*MD                       // Average demand(kW)
ene_daily = avg_demand*24.0              // Daily energy produced(kWh)
// Case(b)
cap_installed = avg_demand/CF            // Installed capacity(kW)
cap_reserve = cap_installed-MD           // Reserve capacity(kW)
// Case(c)
max_ene_C = cap_installed*24.0           // Maximum energy that could be produced daily(kWh)
// Case(d)
max_ene_D = ene_daily/UF                 // Maximum energy that could be produced daily as per schedule(kWh)

// Results
disp("PART I - EXAMPLE : 7.22 : SOLUTION :-")
printf("\nCase(a): Daily energy produced = %.f kWh", ene_daily)
printf("\nCase(b): Reserve capacity of plant = %.f kW", cap_reserve)
printf("\nCase(c): Maximum energy that could be produced daily when plant runs at all time = %.f kWh", max_ene_C)
printf("\nCase(d): Maximum energy that could be produced daily when plant runs fully loaded = %.f
```

```
kWh", max_ene_D)
```

Scilab code Exa 7.23 Rating Annual energy produced Total fixed and variable cost C

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
   GENERATION

// EXAMPLE : 7.23 :
// Page number 83-84
clear ; clc ; close ; // Clear the work space and
   console

// Given data
cap_3sets = 600.0                  // Capacity of 3
   generators(kW)
no_3 = 3.0                         // Number of sets
   of 600 kW
cap_4thset = 400.0                 // Capacity of 4th
    generator set(kW)
no_4 = 1.0                         // Number of sets
   of 400 kW
MD = 1600.0                        // Maximum demand(
   kW)
LF = 0.45                          // Load factor
cost_capital_kW = 1000.0           // Capital cost
   per kW installed capacity(Rs)
cost_annual_per = 0.15             // Annual cost =
   15% of capital cost
cost_operation = 60000.0           // Annual
```

```
operation cost(Rs)
cost_maintenance = 30000.0          // Annual maintenance cost(Rs)
fixed_maintenance = 1.0/3           // Fixed cost
variable_maintenance = 2.0/3        // Variable cost
cost_fuel_kg = 40.0/100             // Cost of fuel oil(Rs/kg)
cost_oil_kg = 1.25                  // Cost of lubricating oil(Rs/kg)
calorific = 10000.0                 // Calorific value of fuel(kcal/kg)
oil_consum = 1.0/400                // Consumption of lubricating oil. 1kg for every 400kWh generated
fuel_consum = 1.0/2                 // Consumption of fuel. 1kg for every 2kWh generated
n_gen = 0.92                        // Generator efficiency
heat_lost = 1.0/3                   // Heat lost in the fuel to cooling water
theta = 11.0                        // Difference of temperature between inlet and outlet( C )

// Calculations
// Case(a)
rating_3set_A = cap_3sets/n_gen                                    // Rating of first 3 sets(kW)
rating_4th_A = cap_4thset/n_gen                                    // Rating of 4th set(kW)
// Case(b)
avg_demand_B = LF*MD                // Average demand(kW)
hours_year = 365.0*24               // Total hours in a year
energy_B = avg_demand_B*hours_year
```

```
                                                        //
   Annual energy produced(kWh)
// Case(c)
total_invest = (no_3*cap_3sets+cap_4thset*no_4)*
   cost_capital_kW                  // Total
   investment(Rs)
annual_cost = cost_annual_per*total_invest
                                              // Annual
   cost(Rs)
maintenance_cost = fixed_maintenance*
   cost_maintenance                                 //
   Maintenance cost(Rs)
fixed_cost_total = annual_cost+maintenance_cost
                                         // Total fixed
   cost per annum(Rs)
fuel_consumption = energy_B*fuel_consum
                                                 // Fuel
   consumption(Kg)
cost_fuel = fuel_consumption*cost_fuel_kg
                                               // Cost of
   fuel(Rs)
oil_consumption = energy_B*oil_consum
                                                   //
   Lubrication oil consumption(Kg)
cost_oil = oil_consumption*cost_oil_kg
                                                  // Cost
   of Lubrication oil(Rs)
var_maintenance_cost = variable_maintenance*
   cost_maintenance                       // Variable
   part of maintenance cost(Rs)
variable_cost_total = cost_fuel+cost_oil+
   var_maintenance_cost+cost_operation   // Total
   variable cost per annum(Rs)
cost_total_D = fixed_cost_total+variable_cost_total
                                     // Total cost per
   annum(Rs)
cost_kWh_gen = cost_total_D/energy_B*100
                                                // Cost per
```

```
kWh generated(Paise)
// Case(c)
n_overall = energy_B*860/(fuel_consumption*calorific)*100                    // Overall efficiency(%)
// Case(d)
weight_water_hr = heat_lost*fuel_consumption/(hours_year*theta)*calorific       // Weight of cooling water required(kg/hr)
weight_water_min = weight_water_hr/60.0                                         // Weight of cooling water required(kg/min)
capacity_pump = weight_water_min*MD/avg_demand_B                                // Capacity of cooling water pump(kg/min)

// Results
disp("PART I - EXAMPLE : 7.23 : SOLUTION :-")
printf("\nCase(a): Rating of first 3 sets of diesel engine = %.f kW", rating_3set_A)
printf("\n          Rating of 4th set of diesel engine = %.f kW", rating_4th_A)
printf("\nCase(b): Annual energy produced = %.1e kWh", energy_B)
printf("\nCase(c): Total fixed cost = Rs %.f ", fixed_cost_total)
printf("\n          Total variable cost = Rs %.f ", variable_cost_total)
printf("\n          Cost per kWh generated = %.f paise", cost_kWh_gen)
printf("\nCase(d): Overall efficiency of the diesel plant = %.1f percent", n_overall)
printf("\nCase(e): Quantity of cooling water required per round = %.2e kg/hr = %.f kg/min", weight_water_hr,weight_water_min)
printf("\n          Capacity of cooling-water pumps under maximum load = %.f kg/min \n", capacity_pump)
```

```
printf("\nNOTE: Changes in obtained answer from that of textbook answer is due to more precision here ')
```

Scilab code Exa 7.24 Turbine rating Energy produced Average steam consumption Evap

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.24 :
// Page number 84
clear ; clc ; close ; // Clear the work space and console

// Given data
cap_installed = 30.0*10**3       // Rating of each generators(kW)
no = 4.0                         // Number of installed generators
MD = 100.0*10**3                 // Maximum demand(kW)
LF = 0.8                         // Load factor
cost_capital_kW = 800.0          // Capital cost per kW installed capacity(Rs)
depreciation_per = 0.125         // Depreciation,etc = 12.5% of capital cost
cost_operation = 1.2*10**6       // Annual operation cost(Rs)
cost_maintenance = 600000.0      // Annual
```

```
    maintenance cost(Rs)
fixed_maintenance = 1.0/3         // Fixed cost
variable_maintenance = 2.0/3      // Variable cost
cost_miscellaneous = 100000.0     // Miscellaneous cost(Rs)
cost_fuel_kg = 32.0/1000          // Cost of fuel oil(Rs/kg)
calorific = 6400.0                // Calorific value of fuel(kcal/kg)
n_gen = 0.96                      // Generator efficiency
n_thermal = 0.28                  // Thermal efficiency of turbine
n_boiler = 0.75                   // Boiler efficiency
n_overall = 0.2                   // Overall thermal efficiency

// Calculations
// Case(a)
rating_turbine = cap_installed/(n_gen*0.736)          // Rating of each steam turbine(metric hp)
// Case(b)
avg_demand_B = LF*MD                                   // Average demand(kW)
hours_year = 365.0*24                                  // Total hours in a year
energy_B = avg_demand_B*hours_year                     // Annual energy produced(kWh)
// Case(c)
steam_consumption_C = (0.8+3.5*LF)/LF                  // Average steam consumption(kg/kWh)
// Case(d)
LF_D = 1.0
```

```
   // Assumption that Load factor for boiler
steam_consumption_D = (0.8+3.5*LF_D)/LF_D          // Steam consumption(kg/kWh)
energy_D = cap_installed*1.0          // Energy output per hour per set(kWh)
evaporation_cap = steam_consumption_D*energy_D          // Evaporation capacity of boiler(kg/hr)
// Case(e)
total_invest = no*cap_installed*cost_capital_kW          // Total investment(Rs)
capital_cost = depreciation_per*total_invest          // Capital cost(Rs)
maintenance_cost = fixed_maintenance*cost_maintenance          // Maintenance cost(Rs)
fixed_cost_total = capital_cost+maintenance_cost          // Total fixed cost per annum(Rs)
var_maintenance_cost = variable_maintenance*cost_maintenance     // Variable part of maintenance cost(Rs)
input_E = energy_B/n_overall          // Input into system per annum(kWh)
weight_fuel = input_E*860/calorific          // Weight of fuel(kg)
cost_fuel = weight_fuel*cost_fuel_kg          // Cost of fuel(Rs)
variable_cost_total = cost_operation+var_maintenance_cost+cost_miscellaneous+cost_fuel     // Total variable cost per annum(Rs)
cost_total_E = fixed_cost_total+variable_cost_total          // Total cost per annum(Rs)
cost_kWh_gen = cost_total_E/energy_B*100          // Cost per kWh generated(
```

```
    Paise)

// Results
disp("PART I - EXAMPLE : 7.24 : SOLUTION :-")
printf("\nCase(a): Rating of each steam turbine = %.
    f metric hp", rating_turbine)
printf("\nCase(b): Energy produced per annum = %.3e
    kWh", energy_B)
printf("\nCase(c): Average steam consumption per kWh
     = %.1f kg/kWh", steam_consumption_C)
printf("\nCase(d): Evaporation capacity of boiler =
    %.f kg/hr", evaporation_cap)
printf("\nCase(e): Total fixed cost = Rs %.2e ",
    fixed_cost_total)
printf("\n          Total variable cost = Rs %.2e ",
    variable_cost_total)
printf("\n          Cost per kWh generated = %.2f
    paise\n", cost_kWh_gen)
printf("\nNOTE: Changes in obtained answer from that
     of textbook answer is due to more precision here
    ')
```

Scilab code Exa 7.25 Plot of hydrograph and Average discharge available

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER
    GENERATION

```

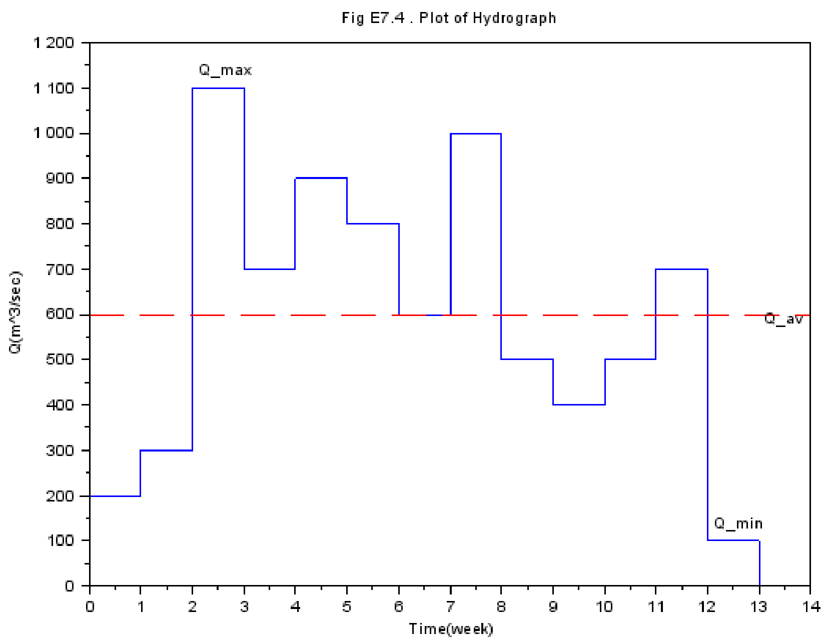

Figure 7.2: Plot of hydrograph and Average discharge available

```
// EXAMPLE : 7.25 :
// Page number 85
clear ; clc ; close ; // Clear the work space and console

// Given data
w1 = 1.0          // Week 1
Q1 = 200.0        // Discharge during week 1(m^2/sec)
w2 = 2.0          // Week 2
Q2 = 300.0        // Discharge during week 2(m^2/sec)
w3 = 3.0          // Week 3
Q3 = 1100.0       // Discharge during week 3(m^2/sec)
w4 = 4.0          // Week 4
Q4 = 700.0        // Discharge during week 4(m^2/sec)
w5 = 5.0          // Week 5
Q5 = 900.0        // Discharge during week 5(m^2/sec)
w6 = 6.0          // Week 6
Q6 = 800.0        // Discharge during week 6(m^2/sec)
w7 = 7.0          // Week 7
Q7 = 600.0        // Discharge during week 7(m^2/sec)
w8 = 8.0          // Week 8
Q8 = 1000.0       // Discharge during week 8(m^2/sec)
w9 = 9.0          // Week 9
Q9 = 500.0        // Discharge during week 9(m^2/sec)
w10 = 10.0        // Week 10
Q10 = 400.0       // Discharge during week 10(m^2/sec)
w11 = 11.0        // Week 11
Q11 = 500.0       // Discharge during week 11(m^2/sec)
w12 = 12.0        // Week 12
Q12 = 700.0       // Discharge during week 12(m^2/sec)
w13 = 13.0        // Week 13
Q13 = 100.0       // Discharge during week 13(m^2/sec)
no_week = 13.0 // Total weeks of discharge

// Calculations
Q_average = (Q1+Q2+Q3+Q4+Q5+Q6+Q7+Q8+Q9+Q10+Q11+Q12+Q13)/no_week      // Average weekly discharge(m^3/sec)
```

```
// Hydrograph
W = [0,w1,w1,w2,w2,w3,w3,w4,w4,w5,w5,w6,w6,w7,w7,w8,w8,w9,w9,w10,w10,w11,w11,w12,w12,w13,w13,w13]
Q = [200,Q1,Q2,Q2,Q3,Q3,Q4,Q4,Q5,Q5,Q6,Q6,Q7,Q7,Q8,Q8,Q9,Q9,Q10,Q10,Q11,Q11,Q12,Q12,Q13,Q13,Q13,0]
a = gca()
a.thickness = 2
    // sets thickness of plot
plot(W,Q)
    // Plotting hydrograph
q = Q_average
w = [0,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,14]
q_dash = [q,q,q,q,q,q,q,q,q,q,q,q,q,q,q]                                        // Plotting average weekly discharge
plot(w,q_dash,'r--')
a.x_label.text = 'Time(week)'                                        // labels x-axis
a.y_label.text = 'Q(m^3/sec)'                                        // labels y-axis
xtitle("Fig E7.4 . Plot of Hydrograph")
xset('thickness',2)
    // sets thickness of axes
xstring(13,560,'Q_av')
xstring(12.02,110,'Q_min')
xstring(2.02,1110,'Q_max')

// Results
disp("PART I - EXAMPLE : 7.25 : SOLUTION :-")
printf("\nThe hydrograph is shown in the Figure E7.4")
printf("\nAverage discharge available for the whole
```

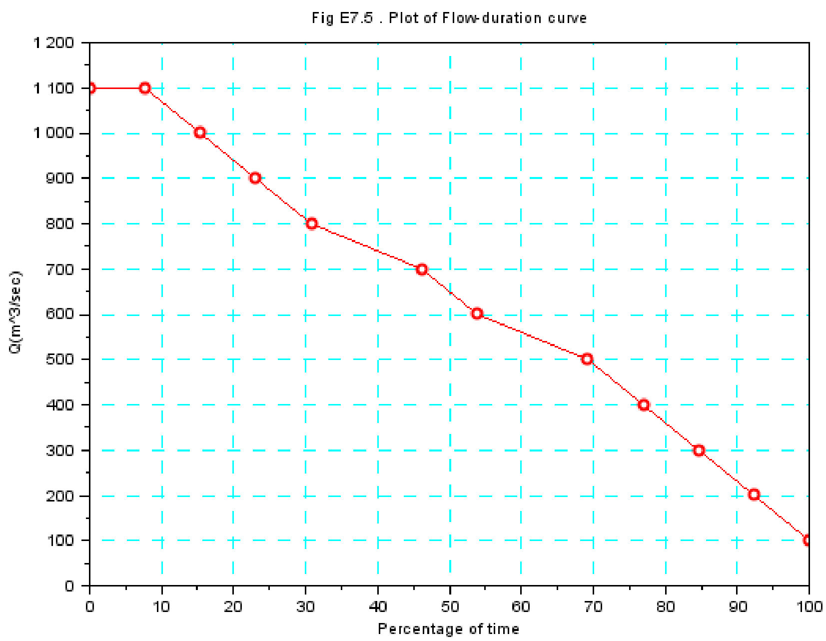

Figure 7.3: Plot of flow duration curve Maximum power Average power developed and Capacity of proposed station

```
period = %.f m^3/sec", Q_average)
```

Scilab code Exa 7.26 `Plot of flow duration curve Maximum power Average power devel`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART I : GENERATION
```

```
// CHAPTER 7: TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION

// EXAMPLE : 7.26 :
// Page number 85-86
clear ; clc ; close ; // Clear the work space and console

// Given data
Q1 = 1100.0             // Discharge in descending order(m^3/sec)
Q2 = 1000.0             // Discharge(m^3/sec)
Q3 = 900.0              // Discharge(m^3/sec)
Q4 = 800.0              // Discharge(m^3/sec)
Q5 = 700.0              // Discharge(m^3/sec)
Q6 = 600.0              // Discharge(m^3/sec)
Q7 = 500.0              // Discharge(m^3/sec)
Q8 = 400.0              // Discharge(m^3/sec)
Q9 = 300.0              // Discharge(m^3/sec)
Q10 = 200.0             // Discharge(m^3/sec)
Q11 = 100.0             // Discharge(m^3/sec)
no_week = 13.0          // Total weeks of discharge
h = 200.0               // Head of installation(m)
n_overall = 0.88        // Overall efficiency of turbine and generator
w = 1000.0              // Density of water(kg/m^3)

// Calculations
n1 = 1.0                             // Number of weeks for 1100 discharge(m^3/sec)
n2 = 2.0                             // Number of weeks for 1000 and above discharge(m^3/sec)
n3 = 3.0                             // Number of weeks for 900 and above discharge(m^3/sec)
n4 = 4.0                             // Number of weeks for 800 and above discharge(m^3/sec)
n5 = 6.0                             // Number of weeks for 700 and above discharge(m^3/sec)
```

```
n6 = 7.0                        // Number of weeks for 600 and above discharge(m^3/sec)
n7 = 9.0                        // Number of weeks for 500 and above discharge(m^3/sec)
n8 = 10.0                       // Number of weeks for 400 and above discharge(m^3/sec)
n9 = 11.0                       // Number of weeks for 300 and above discharge(m^3/sec)
n10 = 12.0                      // Number of weeks for 200 and above discharge(m^3/sec)
n11 = 13.0                      // Number of weeks for 100 and above discharge(m^3/sec)
P1 = n1/no_week*100             // Percentage of total period for n1
P2 = n2/no_week*100             // Percentage of total period for n2
P3 = n3/no_week*100             // Percentage of total period for n3
P4 = n4/no_week*100             // Percentage of total period for n4
P5 = n5/no_week*100             // Percentage of total period for n5
P6 = n6/no_week*100             // Percentage of total period for n6
P7 = n7/no_week*100             // Percentage of total period for n7
P8 = n8/no_week*100             // Percentage of total period for n8
P9 = n9/no_week*100             // Percentage of total period for n9
P10 = n10/no_week*100           // Percentage of total period for n10
P11 = n11/no_week*100           // Percentage of total period for n11
P = [0,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11]
Q = [Q1,Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,Q10,Q11]
                                // Plotting flow duration curve
```

```
a = gca() ;
a.thickness = 2                                              // sets thickness of plot
plot(P,Q,'ro-')
a.x_label.text = 'Percentage of time'                        // labels x-axis
a.y_label.text = 'Q(m^3/sec)'                                // labels y-axis
xtitle("Fig E7.5 . Plot of Flow-duration curve")
xset('thickness',2)                                          // sets thickness of axes
xgrid(4)
Q_1 = 1.0                                   // Discharge (m^3/sec)
P_1 = 0.736/75*w*Q_1*h*n_overall            // Power developed for Q_1(kW)
Q_av = 600.0                                // Average discharge(m^3/sec). Obtained from Example 1.7.25
P_av = P_1*Q_av/1000.0                      // Average power developed(MW)
Q_max = Q1                                  // Maximum discharge(m^3/sec)
P_max = P_1*Q_max/1000.0                    // Maximum power developed(MW)
Q_10 = 1070.0                               // Discharge for 10% of time(m^3/sec). Value is obtained from graph
P_10 = P_1*Q_10/1000.0                      // Installed capacity(MW)

// Results
disp("PART I - EXAMPLE : 7.26 : SOLUTION :-")
printf("\nFlow-duration curve is shown in the Figure E7.5")
printf("\nMaximum power developed = %.f MW", P_max)
printf("\nAverage power developed = %.f MW", P_av)
```

```
printf("\nCapacity of proposed station = %.f MW \n",
    P_10)
printf("\nNOTE: Changes in the obtained answer from
   that of textbook is due to more precision here &
   approximation in textbook solution")
```

Chapter 9

CONSTANTS OF OVERHEAD TRANSMISSION LINES

Scilab code Exa 9.1 Loop inductance and Reactance of transmission line

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
   LINES

// EXAMPLE : 2.1 :
// Page number 100
clear ; clc ; close ; // Clear the work space and
   console

// Given data
D = 100.0            // Distance between conductors(
   cm)
```

```
d = 1.25                // Diameter of conductor(cm)
f = 50.0                // Frequency(Hz)

// Calculations
r_GMR = 0.7788*d/2.0                    // GMR of
    conductor(cm)
L = 4.0*10**-4*log(D/r_GMR)             // Loop
    inductance(H/km)
X_L = 2*%pi*f*L                         // Reactance of
     transmission line(ohm)

// Results
disp("PART II - EXAMPLE : 2.1 : SOLUTION :-")
printf("\nLoop inductance of transmission line, L =
    %.2e H/km", L)
printf("\nReactance of transmission line, X_L = %.2f
     ohm", X_L)
```

Scilab code Exa 9.2 Inductance per phase of the system

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
    LINES

// EXAMPLE : 2.2 :
// Page number 101
clear ; clc ; close ; // Clear the work space and
    console

// Given data
```

```
l = 100.0              // Length of 3-phase transmission line(km)
D = 120.0              // Distance between conductors(cm)
d = 0.5                // Diameter of conductor(cm)

// Calculations
r_GMR = 0.7788*d/2.0                     // GMR of conductor(cm)
L = 2.0*10**-4*log(D/r_GMR)              // Inductance per phase(H/km)
L_l = L*l                                // Inductance per phase for 100km length(H)

// Results
disp("PART II - EXAMPLE : 2.2 : SOLUTION :-")
printf("\nInductance per phase of the system, L = %.4f H \n", L_l)
printf("\nNOTE: ERROR: In textbook to calculate L, log10 is used instead of ln i.e natural logarithm. So, there is change in answer")
```

Scilab code Exa 9.3 `Loop inductance of line per km`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.3 :
// Page number 101
```

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

// Given data
D = 135.0              // Spacing between conductors(cm
   )
r = 0.8                // Radius of conductor(cm)

// Calculations
L = (1+4*log(D/r))*10**-7*1000.0          // Loop
   inductance per km(H)
L_mH = L*1000.0                           // Loop
   inductance per km(mH)

// Results
disp("PART II - EXAMPLE : 2.3 : SOLUTION :-")
printf("\nLoop inductance of line per km, L = %.2f
  mH", L_mH)
```

Scilab code Exa 9.4 Inductance per phase of the system

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
   LINES

// EXAMPLE : 2.4 :
// Page number 101
clear ; clc ; close ; // Clear the work space and
   console

```

```
// Given data
l = 80.0              // Length of 3-phase transmission line(km)
D = 100.0             // Distance between conductors(cm)
d = 1.0               // Diameter of conductor(cm)

// Calculations
r_GMR = 0.7788*d/2.0                  // GMR of conductor(cm)
L = 2.0*10**-7*log(D/r_GMR)           // Inductance per phase(H/m)
L_l = L*l*1000.0                      // Inductance per phase for 80km(H)

// Results
disp("PART II - EXAMPLE : 2.4 : SOLUTION :-")
printf("\nInductance per phase of the system, L = %.4f H \n", L_l)
printf("\nNOTE: ERROR: Calculation mistake in textbook to find Inductance per phase of the system")
```

Scilab code Exa 9.5 Total inductance of the line

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.5 :
```

```
// Page number 103-104
clear ; clc ; close ; // Clear the work space and console

// Given data
D_a_b = 120.0              // Distance between conductors a & b(cm)
D_a_bb = 140.0             // Distance between conductors a & b'(cm)
D_aa_b = 100.0             // Distance between conductors a' & b(cm)
D_aa_bb = 120.0            // Distance between conductors a' & b'(cm)
D_a_aa = 20.0              // Distance between conductors a & a'(cm)
d = 2.0                    // Diameter of conductor(cm)

// Calculations
D_m = (D_a_b*D_a_bb*D_aa_b*D_aa_bb)**(1.0/4)          // Mutual GMD(cm)
D_a_a = 0.7788*d/2.0                                   // Self GMD of conductor a(cm)
D_aa_aa = D_a_a                                        // Self GMD of conductor a'(cm)
D_aa_a = D_a_aa                                        // Distance between conductors a' & a(cm)
D_s = (D_a_a*D_a_aa*D_aa_aa*D_aa_a)**(1.0/4)          // Self GMD(cm)
L = 4*10**-4*log(D_m/D_s)                              // Total inductance of the line(H/km)
L_mH = L*1000.0                                        // Total inductance of the line(mH/km)
```

```
// Results
disp("PART II - EXAMPLE : 2.5 : SOLUTION :-")
printf("\nTotal inductance of the line , L = %.2f mH/
    km", L_mH)
```

Scilab code Exa 9.6 Inductance of the line

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
    LINES

// EXAMPLE : 2.6 :
// Page number 104
clear ; clc ; close ; // Clear the work space and
    console

// Given data
D_a_b = 175.0              // Distance between
    conductors a & b(cm)
D_a_aa = 90.0              // Distance between
    conductors a & a'(cm)
d = 2.5                    // Diameter of conductor(cm
    )

// Calculations
GMR = 0.7788*d/2.0
                                        // GMR(cm)
D_a_a = GMR
                                             // Self
```

```
    GMD of conductor a(cm)
D_aa_aa = D_a_a                                          // Self GMD of conductor a'(cm)
D_aa_a = 90.0                                            // Distance between conductors a' & a(cm)
D_s = (D_a_a*D_a_aa*D_aa_aa*D_aa_a)**(1.0/4)             // Self GMD of conductor A = Self GMD of conductor B(cm)
D_a_bb = (D_a_aa**2+D_a_b**2)**(1.0/2)                   // Distance between conductors a & b'(cm)
D_m = ((D_a_b*D_a_bb)**2)**(1.0/4)                       // Mutual GMD(cm)
L = 4*10**-4*log(D_m/D_s)                                // Inductance of the line(H/km)

// Results
disp("PART II - EXAMPLE : 2.6 : SOLUTION :-")
printf("\nInductance of the line, L = %.1e H/km", L)
```

Scilab code Exa 9.7 Inductance per km of the double circuit line

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
    LINES

// EXAMPLE : 2.7 :
```

```
// Page number 104
clear ; clc ; close ; // Clear the work space and console

// Given data
D_a_a = 100.0                // Distance between conductors a & a(cm)
D_a_b = 25.0                 // Distance between conductors a & b(cm)
d = 2.0                      // Diameter of conductor(cm)

// Calculations
r = d/2.0                                                        // Conductor radius(cm)
GMR = 0.7788*r                                                   // GMR(cm)
D_a_aa = GMR                                                     // GMR of conductors a & a'(cm)
D_aa_a = D_a_aa                                                  // GMR of conductors a' & a(cm)
D_aa_aa = D_a_a                                                  // GMR of conductors a' & a'(cm)
D_s = (D_a_a*D_a_aa*D_aa_aa*D_aa_a)**(1.0/4)             // Self GMD of conductor A = Self GMD of conductor B(cm)
D_a_bb = (D_a_a**2+D_a_b**2)**(1.0/2)                    // Distance between conductors a & b'(cm)
D_aa_b = D_a_bb                                                  // Distance between conductors a' & b(cm)
D_aa_bb = D_a_b                                                  // Distance
```

```
     between conductors a' & b'(cm)
D_m = (D_a_b*D_a_bb*D_aa_b*D_aa_bb)**(1.0/4)
             // Mutual GMD(cm)
L = 2*10**-7*log(D_m/D_s)
                                // Inductance/
    conductor/mt(H)
L_mH = 2.0*L*1000.0*1000.0
                               // Loop inductance per
     km(mH)

// Results
disp("PART II - EXAMPLE : 2.7 : SOLUTION :-")
printf("\nInductance per km of the double circuit
    line, L = %.1f mH", L_mH)
```

Scilab code Exa 9.8 Geometric mean radius of the conductor and Ratio of GMR to ove

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
    LINES

// EXAMPLE : 2.8 :
// Page number 104-105
clear ; clc ; close ; // Clear the work space and
    console

// Given data
n = 7.0          // Number of strands
r = 1.0          // Radius of each conductor. Assume
     it 1 for calculation purpose
```

```

// Calculations
D_1_2 = 2.0*r                                       // Distance between conductor 1 & 2
D_1_6 = 2.0*r                                       // Distance between conductor 1 & 6
D_1_7 = 2.0*r                                       // Distance between conductor 1 & 7
D_3_4 = 2.0*r                                       // Distance between conductor 3 & 4
D_1_4 = 4.0*r                                       // Distance between conductor 1 & 4
D_1_3 = (D_1_4**2-D_3_4**2)**(1.0/2)                // Distance between conductor 1 & 3
D_1_5 = D_1_3                                       // Distance between conductor 1 & 5
GMR = 0.7788*r                                      // GMR
n_o = n-1                                           // Number of outside strands
D_s = (GMR**n*(D_1_2**2*D_1_3**2*D_1_4*D_1_7)**6*(2*r)**n_o)**(1.0/49)    // GMR
overall_radius = 3*r                                // Overall conductor radius
ratio = D_s/overall_radius                          // Ratio of GMR to overall conductor radius

// Results
disp("PART II - EXAMPLE : 2.8 : SOLUTION :-")
printf("\nGeometric mean radius of the conductor, D_s = %.3f*r", D_s)
printf("\nRatio of GMR to overall conductor radius = %.4f ", ratio)
```

Scilab code Exa 9.9 Inductance of the line per phase

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.9 :
// Page number 108-109
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 1.8                    // Diameter of conductor(cm)
D_A_B = 4.0                // Distance between conductor A & B(cm)
D_B_C = 9.0                // Distance between conductor B & C(cm)
D_A_C = 6.0                // Distance between conductor A & C(cm)

// Calculations
D_eq = (D_A_B*D_B_C*D_A_C)**(1.0/3)       // Equivalent distance(cm)
r_GMR = 0.7788*d/2.0                      // GMR(cm)
L = 2*10**-4*log(D_eq/r_GMR)              // Inductance per phase(H/km)
L_mH = L*1000.0                           // Inductance per phase(mH/km)

// Results
disp("PART II - EXAMPLE : 2.9 : SOLUTION :-")
printf("\nInductance of the line per phase, L = %.3f mH/km \n", L_mH)
printf("\nNOTE: ERROR: Calculation mistake in the
```

```
textbook")
```

Scilab code Exa 9.10 `Inductance per km of 3 phase transmission line`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
   LINES

// EXAMPLE : 2.10 :
// Page number 109
clear ; clc ; close ; // Clear the work space and
   console

// Given data
d = 5.0                  // Diameter of conductor(cm
   )
d_1 = 400.0              // Distance between
   conductor 1 & 2(cm)
d_2 = 500.0              // Distance between
   conductor 2 & 3(cm)
d_3 = 600.0              // Distance between
   conductor 1 & 3(cm)

// Calculations
D_eq = (d_1*d_2*d_3)**(1.0/3)                    //
    Equivalent distance(cm)
r_GMR = 0.7788*d/2.0                             //
    GMR(cm)
L = 0.2*log(D_eq/r_GMR)                          //
    Inductance per phase per km(mH)
```

```

// Results
disp("PART II - EXAMPLE : 2.10 : SOLUTION :-")
printf("\nInductance per km of 3 phase transmission line, L = %.3f mH \n", L)
printf("\nNOTE: ERROR: Calculation mistake in the textbook")
```

Scilab code Exa 9.11 `Inductance of each conductor per phase per km`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.11 :
// Page number 109
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 3.0                    // Diameter of conductor(cm)
D_12 = 200.0               // Distance between conductor 1 & 2(cm)
D_23 = 200.0               // Distance between conductor 2 & 3(cm)
D_31 = 400.0               // Distance between conductor 1 & 3(cm)

// Calculations
```

```
D_eq = (D_12*D_23*D_31)**(1.0/3)                    // Equivalent distance(cm)
r = d/2.0                                           // Radius of conductor(cm)
L = (0.5+2*log(D_eq/r))*10**-7                      // Inductance/phase/m(H)
L_mH = L*1000.0*1000.0                              // Inductance per phase per km(mH)

// Results
disp("PART II - EXAMPLE : 2.11 : SOLUTION :-")
printf("\nInductance of each conductor per phase per km, L = %.3f mH \n", L_mH)
printf("\nNOTE: ERROR: Calculation mistake in the textbook")
```

Scilab code Exa 9.12 Inductance of each conductor and Average inductance of each p

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.12 :
// Page number 109-110
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 2.0                         // Diameter of conductor(cm)
```

```
D_ab = 400.0                  // Distance between conductor a & b(cm)
D_bc = 400.0                  // Distance between conductor b & c(cm)
D_ca = 800.0                  // Distance between conductor c & a(cm)

// Calculations
I_ab = 1.0*exp(%i*-240.0*%pi/180)        // I_a/I_b
I_cb = 1.0*exp(%i*-120.0*%pi/180)        // I_c/I_b
r_GMR = 0.7788*d/2.0                     // GMR(cm)
L_a = 2.0*10**-7*complex(log((D_ab*D_ca)**0.5/r_GMR),(3**0.5/2*log(D_ab/D_ca)))    // Inductance per phase of A(H/m)
L_amH = L_a*10.0**6                      // Inductance per phase of A(mH/km)
L_b = 2.0*10**-7*complex(log((D_bc*D_ab)**0.5/r_GMR),(3**0.5/2*log(D_bc/D_ab)))    // Inductance per phase of B(H/m)
L_bmH = L_b*10.0**6                      // Inductance per phase of B(mH/km)
L_c = 2.0*10**-7*complex(log((D_ca*D_bc)**0.5/r_GMR),(3**0.5/2*log(D_ca/D_bc)))    // Inductance per phase of C(H/m)
L_cmH = L_c*10.0**6                      // Inductance per phase of C(mH/km)
D_eq = (D_ab*D_bc*D_ca)**(1.0/3)         // Equivalent distance(cm)
```

```
L_avg = 0.2*log(D_eq/r_GMR)    // Average inductance per phase(mH/km)

// Results
disp("PART II - EXAMPLE : 2.12 : SOLUTION :-")
printf("\nInductance of conductor a, L_a = (%.4f%.2fj) mH/km", real(L_amH),imag(L_amH))
printf("\nInductance of conductor b, L_b = %.3f mH/km", abs(L_bmH))
printf("\nInductance of conductor c, L_c = (%.4f+%.2fj) mH/km", real(L_cmH),imag(L_cmH))
printf("\nAverage inductance of each phase, L_avg = %.3f mH/km", L_avg)
```

Scilab code Exa 9.13 Inductance per phase

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.13 :
// Page number 110
clear ; clc ; close ; // Clear the work space and console

// Given data
D_a_a = 0.9                    // Self GMD of conductor a(cm)
D_a_aa = 40.0                  // Distance between
```

```
conductor a & a'(cm)
D_a_b = 1000.0                    // Distance between conductor a & b(cm)
D_a_bb = 1040.0                   // Distance between conductor a & b'(cm)
D_aa_b = 960.0                    // Distance between conductor a' & b(cm)
D_c_a = 2000.0                    // Distance between conductor a & c(cm)
D_c_aa = 1960.0                   // Distance between conductor a' & c(cm)
D_cc_a = 2040.0                   // Distance between conductor a & c'(cm)

// Calculations
D_aa_aa = D_a_a                                          // Self GMD of conductor a'(cm)
D_aa_a = D_a_aa                                          // Distance between conductor a' & a(cm)
D_s1 = (D_a_a*D_a_aa*D_aa_aa*D_aa_a)**(1.0/4)            // Self GMD in position 1(cm)
D_s2 = D_s1                                              // Self GMD in position 2(cm)
D_s3 = D_s1                                              // Self GMD in position 3(cm)
D_s = (D_s1*D_s2*D_s3)**(1.0/3)                          // Equivalent self GMD(cm)
D_aa_bb = D_a_b                                          // Distance between conductor a' & b'(cm)
D_AB = (D_a_b*D_a_bb*D_aa_b*D_aa_bb)**(1.0/4)            // Mutual GMD(cm)
D_BC = D_AB                                              // Mutual GMD(cm)
D_cc_aa = D_c_a                                          // Distance between conductor a' & c'(cm)
D_CA = (D_c_a*D_c_aa*D_cc_a*D_cc_aa)**(1.0/4)            // Mutual GMD(cm)
D_m = (D_AB*D_BC*D_CA)**(1.0/3)                          //
```

```
    Equivalent Mutual GMD(cm)
L = 0.2*log(D_m/D_s)                        // Inductance per phase(mH/km)

// Results
disp("PART II - EXAMPLE : 2.13 : SOLUTION :-")
printf("\nInductance per phase, L = %.3f mH/km", L)
```

Scilab code Exa 9.14 Inductance per phase of double circuit

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.14 :
// Page number 110-111
clear ; clc ; close ; // Clear the work space and console

// Given data
r = 6.0/1000                 // Radius of conductor(m)
D_a_cc = 5.0                 // Distance between conductor a & c'(m)
D_b_bb = 6.0                 // Distance between conductor b & b'(m)
D_c_aa = 5.0                 // Distance between conductor c & a'(m)
D_acc_bbb = 3.0              // Distance between conductor ac' & bb'(m)
```

```
D_bbb_caa = 3.0                  // Distance between conductor bb' & ca'(m)
D_a_c = 6.0                      // Distance between conductor a & c(m)

// Calculations
r_GMR = 0.7788*r                                       // GMR of conductor(m)
D_a_b = (D_acc_bbb**2+((D_b_bb-D_a_cc)/2)**2)**(1.0/2)              // Distance between conductor a & b(m)
D_a_bb = (D_acc_bbb**2+(D_a_cc+(D_b_bb-D_a_cc)/2)**2)**(1.0/2)     // Distance between conductor a & b'(m)
D_a_aa = ((D_acc_bbb+D_bbb_caa)**2+D_c_aa**2)**(1.0/2)              // Distance between conductor a & a'(m)
D_a_a = r_GMR                                          // Self GMD of conductor a(m)
D_aa_aa = D_a_a                                        // Self GMD of conductor a'(m)
D_aa_a = D_a_aa                                        // Distance between conductor a' & a(m)
D_S1 = (D_a_a*D_a_aa*D_aa_aa*D_aa_a)**(1.0/4)                       // Self GMD in position 1(m)
D_bb_b = D_b_bb                                        // Distance between conductor b' & b(m)
D_S2 = (D_a_a*D_b_bb*D_aa_aa*D_bb_b)**(1.0/4)                       // Self GMD in position 2(m)
D_S3 = (D_a_a*D_a_aa*D_aa_aa*D_aa_a)**(1.0/4)                       // Self GMD in position 3(m)
D_S = (D_S1*D_S2*D_S3)**(1.0/3)                                     // Equivalent
```

```
    self GMD(m)
D_aa_bb = D_a_b                                    // Distance between conductor a' & b'(m)
D_aa_b = D_a_bb                                    // Distance between conductor a' & b(m)
D_AB = (D_a_b*D_a_bb*D_aa_b*D_aa_bb)**(1.0/4)      // Mutual GMD(m)
D_BC = D_AB                                        // Mutual GMD(m)
D_c_a = D_a_c                                      // Distance between conductor c & a(m)
D_cc_aa = D_c_a                                    // Distance between conductor a' & c'(m)
D_cc_a = D_a_cc                                    // Distance between conductor c' & a(m)
D_CA = (D_c_a*D_c_aa*D_cc_a*D_cc_aa)**(1.0/4)      // Mutual GMD(m)
D_m = (D_AB*D_BC*D_CA)**(1.0/3)                    // Equivalent Mutual GMD(m)
L = 0.2*log(D_m/D_S)                               // Inductance per phase(mH/km)

// Results
disp("PART II - EXAMPLE : 2.14 : SOLUTION :-")
printf("\nInductance per phase, L = %.2f mH/km", L)
```

Scilab code Exa 9.15 Spacing between adjacent conductor to keep same inductance

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
   LINES

// EXAMPLE : 2.15 :
// Page number 111
clear ; clc ; close ; // Clear the work space and
   console

// Given data
D_eq = 2.88                          // Equilateral
   spacing of line(m)

// Calculations
D = D_eq/2**(1.0/3)                  // Distance(m)
D_13 = 2.0*D                         // Distance between
   conductor 1 & 3(m)
D_12 = D                             // Distance between
   conductor 1 & 2(m)
D_23 = D                             // Distance between
   conductor 2 & 3(m)

// Results
disp("PART II - EXAMPLE : 2.15 : SOLUTION :-")
printf("\nSpacing between conductor 1 & 2 to keep
   inductance same, D_12 = %.1f m", D_12)
printf("\nSpacing between conductor 2 & 3 to keep
   inductance same, D_23 = %.1f m", D_23)
printf("\nSpacing between conductor 1 & 3 to keep
   inductance same, D_13 = %.1f m", D_13)
```

Scilab code Exa 9.16 `Capacitance of line neglecting and taking presence of ground`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
   LINES

// EXAMPLE : 2.16 :
// Page number 112
clear ; clc ; close ; // Clear the work space and
   console

// Given data
l = 40.0       // Length of line(km)
d = 5.0/1000   // Diameter of wire(m)
D = 1.5        // Spacing between conductor(m)
h = 7.0        // Height of conductors above ground(m
   )

// Calculations
r = d/2
                                                    // Radius of wire(m)
e = 1.0/(36*%pi)*10**-9                                          // Constant
    _0
// Neglecting presence of ground
C_ab_1 = %pi*e/(log(D/r))                                      //
   Capacitance(F/m)
```

```
C_ab_12 = C_ab_1*l*1000.0*10**6                                   // Capacitance( F )
// Taking presence of ground
C_ab_2 = %pi*e/log(D/(r*(1+(D/(2*h))**2)**(1.0/2)))                // Capacitance(F/m)
C_ab_22 = C_ab_2*l*1000.0*10**6                                   // Capacitance( F )

// Results
disp("PART II - EXAMPLE : 2.16 : SOLUTION :-")
printf("\nCapacitance of line neglecting presence of ground, C_ab = %.3f  F ", C_ab_12)
printf("\nCapacitance of line taking presence of ground, C_ab = %.3f  F ", C_ab_22)
```

Scilab code Exa 9.17 Capacitance of conductor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.17 :
// Page number 114-115
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 2.0/100      // Diameter of conductor(m)
D_AB = 4.0       // Spacing between conductor A & B(m)
D_BC = 4.0       // Spacing between conductor B & C(m)
```

```
D_CA = 8.0      // Spacing between conductor C & A(m)

// Calculations
r = d/2                                    // Radius of conductor(m)
D = 4.0                                    // Assuming coomon distance(m)
e = 1.0/(36*%pi)*10**-9                    // Constant _0
C_A = 2*%pi*e/(log(D/r)-complex(-0.5,0.866)*log(2))*1000.0      // Capacitance of conductor A(F/km)
C_Au = C_A*10.0**6                         // Capacitance of conductor A( F /km)
C_B = 2*%pi*e/log(D/r)*1000.0              // Capacitance of conductor B(F/km)
C_Bu = C_B*10.0**6                         // Capacitance of conductor B( F /km)
C_C = 2*%pi*e/(log(D/r)-complex(-0.5,-0.866)*log(2))*1000.0     // Capacitance of conductor C(F/km)
C_Cu = C_C*10.0**6                         // Capacitance of conductor C( F /km)

// Results
disp("PART II - EXAMPLE : 2.17 : SOLUTION :-")
printf("\nCapacitance of conductor A, C_A = (%.5f+%.6fj)  F /km", real(C_Au),imag(C_Au))
printf("\nCapacitance of conductor B, C_B = %.6f  F /km", C_Bu)
printf("\nCapacitance of conductor C, C_C = (%.5f%.6fj)  F /km", real(C_Cu),imag(C_Cu))
```

Scilab code Exa 9.18 New value of capacitance

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.18 :
// Page number 115
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 2.0/100      // Diameter of conductor(m)
D_AB = 4.0       // Spacing between conductor A & B(m)
D_BC = 4.0       // Spacing between conductor B & C(m)
D_CA = 8.0       // Spacing between conductor C & A(m)

// Calculations
r = d/2                                       // Radius of conductor(m)
e = 1.0/(36*%pi)*10**-9                       // Constant _0
D_eq = (D_AB*D_BC*D_CA)**(1.0/3)              // Equivalent distance(m)
C_n = 2*%pi*e/log(D_eq/r)*1000.0              // Capacitance to neutral(F/km)
C_nu = C_n*10.0**6                            // Capacitance to neutral( F /km)

```

```
// Results
disp("PART II - EXAMPLE : 2.18 : SOLUTION :-")
printf("\nNew value of capacitance , C_n = %.5f   F /
   km \n", C_nu)
printf("\nNOTE: Changes in the obtained answer from
   that of textbook is due to more approximation in
   the textbook")
```

Scilab code Exa 9.19 Capacitance per phase to neutral of a line

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
   LINES

// EXAMPLE : 2.19 :
// Page number 115
clear ; clc ; close ; // Clear the work space and
   console

// Given data
d = 2.6          // Outside diameter of conductor(cm)
D_RY = 8.0       // Spacing between conductor R & Y(m)
D_YB = 8.0       // Spacing between conductor Y & B(m)
D_RB = 16.0      // Spacing between conductor R & B(m)
h = 13.0         // Height of conductor from ground(m)

// Calculations
r = d/2

   // Radius of conductor(m)
```

```
e = 1.0/(36*%pi)*10**-9                                   // Constant _0
h_12 = (D_RY**2+(2*h)**2)**(1.0/2)                        // Height of conductor 1 & 2(m)
h_23 = h_12                                               // Height of conductor 2 & 3(m)
h_31 = (D_RB**2+(2*h)**2)**(1.0/2)                        // Height of conductor 3 & 1(m)
h_1 = 2*h                                                 // Height of transposed conductor 1(m)
h_2 = 2*h                                                 // Height of transposed conductor 2(m)
h_3 = 2*h                                                 // Height of transposed conductor 3(m)
D_eq = (D_RY*D_YB*D_RB)**(1.0/3)                          // Equivalent distance(m)
h_123 = (h_12*h_23*h_31)**(1.0/3)                         // Height(m)
h_1_2_3 = (h_1*h_2*h_3)**(1.0/3)                          // Height(m)
C_n = 2*%pi*e/(log(D_eq*100/r)-log(h_123/h_1_2_3))*1000.0     // Capacitance of conductor A(F/km)

// Results
disp("PART II - EXAMPLE : 2.19 : SOLUTION :-")
printf("\nCapacitance per phase to neutral of a line, C_n = %.1e F/km", C_n)
```

Scilab code Exa 9.20 `Phase to neutral capacitance`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.20 :
// Page number 117-118
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 2.5           // Diameter of conductor(cm)
D = 200.0         // Distance of separation(cm)
l = 100.0         // Length of line(km)

// Calculations
r = d/2                                          // Radius of conductor(cm)
e = 1.0/(36*%pi)*10**-9                          // Constant _0
D_m = (D*(3**0.5)*D*(3**0.5)*D*D)**(1.0/4)       // Mutual GMD(cm)
D_s = (2*D*r)**(1.0/2)                           // Self GMD(cm)
C_n = 2*%pi*e/log(D_m/D_s)*1000.0                // Phase-to-neutral capacitance(F/km)
C_nu = C_n*l*10.0**6                             // Phase-to-neutral capacitance( F )
```

```

// Results
disp("PART II - EXAMPLE : 2.20 : SOLUTION :-")
printf("\nPhase-to-neutral capacitance, C_n = %.2f F ", C_nu)
```

Scilab code Exa 9.21 Capacitance per phase to neutral

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.21 :
// Page number 118
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 2.5/100          // Diameter of conductor(m)
D = 5.0              // Distance of separation(m)
h = 2.0              // Height of separation(m)

// Calculations
r = d/2                                              // Radius of conductor(m)
e = 1.0/(36*%pi)*10**-9                              // Constant _0
m = (D**2+h**2)**(1.0/2)                             // (m)
n = (D**2+(h*2)**2)**(1.0/2)                         //
```

```
    (m)
D_ab = (D*m)**(1.0/2)                          //
    Distance between conductor a & b(m)
D_bc = (D*m)**(1.0/2)                          //
    Distance between conductor b & c(m)
D_ca = (2*D*h)**(1.0/2)                        //
    Distance between conductor c & a(m)
D_eq = (D_ab*D_bc*D_ca)**(1.0/3)               //
    Equivalent GMD(m)
D_s1 = (r*n)**(1.0/2)                          //
    Self GMD in position 1(m)
D_s2 = (r*h)**(1.0/2)                          //
    Self GMD in position 2(m)
D_s3 = (r*n)**(1.0/2)                          //
    Self GMD in position 3(m)
D_s = (D_s1*D_s2*D_s3)**(1.0/3)                //
    Self GMD(m)
C_n = 2*%pi*e/log(D_eq/D_s)*1000.0             //
    Capacitance per phase to neutral(F/km)
C_nu = C_n*10.0**6                             //
    Capacitance per phase to neutral( F /km)

// Results
disp("PART II - EXAMPLE : 2.21 : SOLUTION :-")
printf("\nCapacitance per phase to neutral, C_n = %
    .2f F /km", C_nu)
```

Scilab code Exa 9.22 Capacitive reactance to neutral and Charging current per phas

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
```

```
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.22 :
// Page number 119
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 2.5/100          // Diameter of conductor(m)
V = 132.0*10**3      // Line voltage(V)
f = 50.0             // Frequency(Hz)
h = 4.0              // Height(m)
H = 8.0              // Height of separation(m)
D_1_33 = 7.0         // Distance between conductors 1 & 3'(m)
D_1_22 = 9.0         // Distance between conductors 1 & 2'(m)
D_1_11 = 8.0         // Distance between conductors 1 & 1'(m)
D_1 = 1.0            // Distance(m)

// Calculations
r = d/2                                            // Radius of conductor(m)
e = 1.0/(36*%pi)*10**-9                            // Constant _0
D_12 = (h**2+D_1**2)**(1.0/2)                      // Distance between conductors 1 & 2(m)
D_122 = (h**2+D_1_11**2)**(1.0/2)                  // Distance between conductors 1 & 2'(m)
D_111 = (D_1_11**2+D_1_33**2)**(1.0/2)             // Distance between conductors 1 & 1'(m)
D_1_2 = (D_12*D_122)**(1.0/2)                      // Mutual GMD(m)
D_2_3 = (D_12*D_122)**(1.0/2)                      // Mutual GMD(m)
D_3_1 = (D_1_33*D_1_11)**(1.0/2)                   //
```

```
Mutual GMD(m)
D_eq = (D_1_2*D_2_3*D_3_1)**(1.0/3)                        //
    Equivalent GMD(m)
D_s1 = (r*D_111)**(1.0/2)                                  //
    Self GMD in position 1(m)
D_s2 = (r*D_1_22)**(1.0/2)                                 //
    Self GMD in position 2(m)
D_s3 = (r*D_111)**(1.0/2)                                  //
    Self GMD in position 3(m)
D_s = (D_s1*D_s2*D_s3)**(1.0/3)                            //
    Self GMD(m)
C_n = 2*%pi*e/log(D_eq/D_s)                                //
    Capacitance per phase to neutral(F/m)
X_cn = 1/(2.0*%pi*f*C_n)                                   //
    Capacitive reactance to neutral(ohms/m)
V_ph = V/(3**0.5)                                          //
    Phase voltage(V)
I_charg = V_ph/X_cn*1000.0                                 //
    Charging current per phase(A/km)

// Results
disp("PART II - EXAMPLE : 2.22 : SOLUTION :-")
printf("\nCapacitive reactance to neutral, X_cn = %
    .2e ohms/m", X_cn)
printf("\nCharging current per phase, I_charg = %.3f
     A/km", I_charg)
```

Scilab code Exa 9.23 Inductive reactance Capacitance and Capacitive reactance of t

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
```

```
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.23 :
// Page number 119
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 0.8/100          // Diameter of conductor(m)
f = 50.0             // Frequency(Hz)
D_a_b = 5.0          // Distance between conductors a & b(m)
D_b_c = 5.0          // Distance between conductors b & c(m)
D_c_a = 8.0          // Distance between conductors c & a(m)
l = 25.0             // Length of line(km)

// Calculations
r = d/2                                                    // Radius of conductor(m)
e = 8.854*10**-12                                          // Constant  _0
D_e = (D_a_b*D_b_c*D_c_a)**(1.0/3)                         // Equivalent GMD(m)
L = 2*((1.0/4)+log(D_e/r))*10**-4                          // Inductance(H/km)
X_L = 2*%pi*f*L                                            // Inductive reactance per km(ohms)
C = %pi*e/log(D_e/r)                                       // Capacitance(F/m)
C_l = C*1000.0*l                                           // Capacitance for entire length(F)
C_lu = C_l*10.0**6                                         // Capacitance for entire length( F )
X_c = 1/(2.0*%pi*f*C_l)                                    // Capacitive reactance to neutral(ohm)
```

```
X_ck = X_c/1000.0                                       //
    Capacitive reactance to neutral(kilo-ohm)

// Results
disp("PART II - EXAMPLE : 2.23 : SOLUTION :-")
printf("\nInductive reactance of the line per
    kilometer per phase, X_L = %.3f ohm", X_L)
printf("\nCapacitance of the line, C = %.3f  F ",
    C_lu)
printf("\nCapacitive reactance of the transmission
    line, X_c = %.1f kilo-ohm\n", X_ck)
printf("\nNOTE: ERROR: Change in obtained answer
    from that of textbook due to wrong substitution
    in finding Capacitance")
```

Scilab code Exa 9.24 Capacitance of the line and Charging current

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
    LINES

// EXAMPLE : 2.24 :
// Page number 119-120
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 250.0             // Line voltage(V)
f = 50.0              // Frequency(Hz)
D = 1.5               // Distance of separation(m)
```

```
d = 1.5/100          // Diameter of conductor(m)
l = 50.0             // Length of line(km)

// Calculations
// Case(i)
r = d/2                                                   // Radius of conductor(m)
e = 8.854*10**-12                                         // Constant _0
C = %pi*e/log(D/r)                                        // Capacitance(F/m)
C_l = C*1000.0*l                                          // Capacitance for entire length(F)
C_lu = C_l*10.0**6                                        // Capacitance for entire length( F )
// Case(ii)
I_charg = 2.0*%pi*f*C_l*V*1000.0                          // Charging current(mA)

// Results
disp("PART II - EXAMPLE : 2.24 : SOLUTION :-")
printf("\nCase(i) : Capacitance of the line, C = %.3f  F ", C_lu)
printf("\nCase(ii): Charging current, I_charg = %.2f mA", I_charg)
```

Scilab code Exa 9.25 `Capacitance of the line`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION
```

```
LINES

// EXAMPLE : 2.25 :
// Page number 120
clear ; clc ; close ; // Clear the work space and console

// Given data
d_1 = 6.0                   // Distance between conductor 1 & 2(m)
d_2 = 6.0                   // Distance between conductor 2 & 3(m)
d_3 = 12.0                  // Distance between conductor 3 & 1(m)
dia = 1.24/100              // Diameter of conductor(m)
l = 100.0                   // Length of line(km)

// Calculations
r = dia/2                                         // Radius of conductor(m)
e = 8.854*10**-12                                 // Constant _0
d = (d_1*d_2*d_3)**(1.0/3)                        // Distance(m)
C = 2*%pi*e/log(d/r)                              // Capacitance(F/m)
C_l = C*1000.0*l                                  // Capacitance for entire length(F)
C_lu = C_l*10.0**6                                // Capacitance for entire length( F )

// Results
disp("PART II - EXAMPLE : 2.25 : SOLUTION :-")
printf("\nCapacitance of the line, C = %.3f  F ", C_lu)
```

Scilab code Exa 9.26 `Capacitance of each line conductor`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 2: CONSTANTS OF OVERHEAD TRANSMISSION LINES

// EXAMPLE : 2.26 :
// Page number 120
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 2.0                  // Spacing between conductors (m)
dia = 1.25/100           // Diameter of conductor(m)

// Calculations
r = dia/2                                  // Radius of conductor(m)
e = 8.854*10**-12                          // Constant  _0
C = 2*%pi*e/log(d/r)                       // Capacitance(F/m)
C_u = C*1000*10.0**6                       // Capacitance for entire length( F /km)

// Results
disp("PART II - EXAMPLE : 2.26 : SOLUTION :-")
printf("\nCapacitance of each line conductor, C = %.4f  F /km", C_u)
```

Chapter 10

STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

Scilab code Exa 10.1 Voltage regulation Sending end power factor and Transmission

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND
   PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.1 :
// Page number 127-128
clear ; clc ; close ; // Clear the work space and
   console

// Given data
```

```
P = 2.0*10**6               // Power delivered(W)
V_r = 33.0*10**3            // Receiving end voltage(V)
PF_r = 0.8                  // Receiving end lagging power factor
R = 10.0                    // Total resistance of the line(ohm)
X = 18.0                    // Total inductive resistance of the line(ohm)

// Calculations
// Case(i)
I = P/(V_r*PF_r)                        // Line current(A)
sin_phi_r = (1-PF_r**2)**0.5            // Sin _R
V_s = V_r+I*R*PF_r+I*X*sin_phi_r        // Sending end voltage(V)
reg = (V_s-V_r)/V_r*100                 // Voltage regulation(%)
// Case(ii)
PF_s  = (V_r*PF_r+I*R)/V_s              // Sending end lagging power factor
// Case(iii)
loss = I**2*R                           // Losses(W)
P_s = P+loss                            // Sending end power(W)
n = P/P_s*100                           // Transmission efficiency(%)

// Results
disp("PART II - EXAMPLE : 3.1 : SOLUTION :-")
printf("\nCase(i)  : Percentage voltage regulation = %.3f percent", reg)
printf("\nCase(ii) : Sending end power factor = %.2f (lag)", PF_s)
printf("\nCase(iii): Transmission efficiency ,   = %.2f percent \n", n)
printf("\nNOTE: ERROR: pf is 0.8 and not 0.9 as mentioned in the textbook problem statement")
```

Scilab code Exa 10.2 Line current Receiving end voltage and Efficiency of transmis

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.2 :
// Page number 128-129
clear ; clc ; close ; // Clear the work space and console

// Given data
l = 10.0                    // Length(km)
V_s = 11.0*10**3            // Sending end voltage(V)
P = 1000.0*10**3            // Load delivered at receiving end(W)
PF_r = 0.8                  // Receiving end lagging power factor
r = 0.5                     // Resistance of each conductor(ohm/km)
x = 0.56                    // Reactance of each conductor(ohm/km)

// Calculations
// Case(a)
R = r*l                                  // Resistance per phase(ohm)
X = x*l                                  // Reactance per phase(ohm)
```

```
E_s = V_s/3**0.5                        // Phase voltage (V)
I = P/(3**0.5*V_s*PF_r)                 // Line current (A)
// Case(b)
sin_phi_r = (1-PF_r**2)**0.5            // Sin _R
E_r = E_s-I*R*PF_r-I*X*sin_phi_r        // Receiving end voltage(V)
E_r_ll = 3**0.5*E_r/1000                // Receiving end line to line voltage(kV)
// Case(c)
loss = 3*I**2*R                         // Loss in the transmission line(W)
P_s = P+loss                            // Sending end power(W)
n = P/P_s*100                           // Transmission efficiency(%)
// Alternate method
Z = R**2+X**2
P_A = 1.0/3*P                           // Load delivered(W/phase)
Q = 1.0*P*sin_phi_r/(3*PF_r)            // Reactive load delivered(VAR/phase)
A = (V_s**2/3.0)-2*(P_A*R+Q*X)          // Constant
B = (1/9.0)*P**2*Z/PF_r**2              // Constant
const = (A**2-4*B)**0.5                 // sqrt(A^2-4B)
E_r_A = ((A+const)/2)**0.5/1000.0       // Receiving end voltage(kV/phase)
E_r_A_ll = 3**0.5*E_r_A                 // Receiving end line-line voltage(kV)
I_A = P/(3**0.5*E_r_A_ll*1000*PF_r)     // Line current(A)
loss_A = 3*I_A**2*R                     // Loss in the transmission line(W)
P_s_A = P+loss_A                        // Sending end power(W)
n_A = P/P_s_A*100                       // Transmission efficiency(%)
```

```

// Results
disp("PART II - EXAMPLE : 3.2 : SOLUTION :-")
printf("\nCase(a): Line current, |I| = %.1f A", I)
printf("\nCase(b): Receiving end voltage, E_r = %.f V (line-to-neutral) = %.2f kV (line-to-line)", E_r,E_r_ll)
printf("\nCase(c): Efficiency of transmission = %.2f percent \n", n)
printf("\nAlternative solution by mixed condition:")
printf("\nCase(a): Line current, |I| = %.1f A", I_A)
printf("\nCase(b): Receiving end voltage, E_r = %.3f kV/phase = %.2f kV (line-line)", E_r_A,E_r_A_ll)
printf("\nCase(c): Efficiency of transmission = %.2f percent", n_A)
```

Scilab code Exa 10.3 `Sending end voltage`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.3 :
// Page number 129
clear ; clc ; close ; // Clear the work space and console

// Given data
I = 200.0                // Line current(A)
PF_r = 0.8               // Receiving end lagging
```

```
    power factor
R = 0.6                    // Total resistance of the
    line(ohm)
X = 1.0                    // Total inductive
    resistance of the line(ohm)
n  = 0.93                  // Efficiency(%)

// Calculations
V_r = 3*I**2*R/((3*I*PF_r/n)-3*I*PF_r)    //
    Receiving end phase voltage(V)
sin_phi_r = (1-PF_r**2)**0.5              // Sin _R
V_s = V_r+I*R*PF_r+I*X*sin_phi_r          // Sending
    end voltage(V)
V_s_ll = 3**0.5*V_s                       // Sending
    end line voltage(V)

// Results
disp("PART II - EXAMPLE : 3.3 : SOLUTION :-")
printf("\nSending end voltage, V_s(line-line)  = %.2
    f V", V_s_ll)
```

Scilab code Exa 10.4 Distance over which load is delivered

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND
    PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.4 :
// Page number 129
clear ; clc ; close ; // Clear the work space and
```

```
console

// Given data
P = 15.0*10**6                  // Load delivered at receiving end(W)
PF_r = 0.85                     // Receiving end lagging power factor
r = 0.905                       // Resistance of each conductor(ohm/km)
V_r = 132.0*10**3               // Receiving end voltage(V)
loss_per = 7.5/100              // Loss

// Calculations
loss = loss_per*P                    // Losses in line(W)
I = P/(3**0.5*V_r*PF_r)              // Line current(A)
l = loss/(3*I**2*r)                  // Length of line(km)

// Results
disp("PART II - EXAMPLE : 3.4 : SOLUTION :-")
printf("\nDistance over which load is delivered , l = %.2f km", l)
```

Scilab code Exa 10.5 Sending end voltage Voltage regulation Value of capacitors an

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.5 :
```

```
// Page number 130
clear ; clc ; close ; // Clear the work space and console

// Given data
f = 50.0                    // Frequency(Hz)
l = 20.0                    // Length(km)
P = 5.0*10**6               // Load delivered at receiving end(W)
PF_r = 0.8                  // Receiving end lagging power factor
r = 0.02                    // Resistance of each conductor(ohm/km)
L = 0.65*10**-3             // Inductance of each conductor(H/km)
E_r = 10.0*10**3            // Receiving end voltage(V)

// Calculations
R = r*l                                           // Resistance per phase(ohm)
X = 2*%pi*f*L*l                                 // Reactance per phase(ohm)
// Case(a)
I = P/(E_r*PF_r)                                  // Line current(A)
sin_phi_r = (1-PF_r**2)**0.5                      // Sin _R
E_s = E_r+I*R*PF_r+I*X*sin_phi_r                  // Sending end voltage(V)
E_s_kV = E_s/1000.0                               // Sending end voltage(kV)
reg = (E_s-E_r)/E_r*100                           // Voltage regulation(%)
// Case(b)
reg_new = reg/2                                   // New regulation(%)
E_s_new = (reg_new/100)*E_r+E_r                   // New value of sending end voltage(V)
```

```
tan_phi_r1 = ((E_s_new-E_r)*(E_r/P)-R)/X     // tan _r1
phi_r1 = atan(tan_phi_r1)                     //  _r1 (radians)
phi_r1d = phi_r1*180/%pi                      //  _r1 (degree)
PF_r1 = cos(phi_r1)                           // Lagging power factor of receiving end
sin_phi_r1 = (1-PF_r1**2)**0.5                // Sin _r1
I_R_new = P/(E_r*PF_r1)                       // New line current(A)
I_R = I_R_new*complex(PF_r1,-sin_phi_r1)
I_c = I_R-I*complex(PF_r,-sin_phi_r)          // Capacitive current(A)
I_C = imag(I_c)                               // Imaginary part of Capacitive current(A)
c = I_C/(2*%pi*f*E_r)*10.0**6                 // Capacitance( F )
// Case(c)
loss_1 = I**2*R                               // Loss(W)
n_1 = P/(P+loss_1)*100                        // Transmission efficiency(%)
loss_2 = I_R_new**2*R                         // Loss(W)
n_2 = P/(P+loss_2)*100                        // Transmission efficiency(%)

// Results
disp("PART II - EXAMPLE : 3.5 : SOLUTION :-")
printf("\nCase(a): Sending end voltage, E_s = %.2f kV", E_s_kV)
printf("\n         Voltage regulation of the line = %.1f percent", reg)
printf("\nCase(b): Value of capacitors to be placed in parallel with load, c = %.2f  F ", c)
printf("\nCase(c): Transmission efficiency in part(a
```

```
    ), _1 = %.2f percent", n_1)
printf("\n          Transmission efficiency in part(b
    ), _2 = %.1f percent", n_2)
```

Scilab code Exa 10.6 Voltage regulation Sending end voltage Line loss and Sending

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND
   PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.6 :
// Page number 130-131
clear ; clc ; close ; // Clear the work space and
    console

// Given data
f = 50.0                                                    //
    Frequency(Hz)
l = 10.0                                                    //
    Line length(km)
Z_l = 0.5*exp(%i*60.0*%pi/180)
                          // Load impedance(ohm/km)
P = 316.8*10**3
                                          // Load side
     power(W)
PF_r = 0.8
                                                  // Load
     side power factor
```

```
E_r = 3.3*10**3                                         // Load bus voltage(V)

// Calculations
Z_line = Z_l*l                                          // Load impedance(ohm)
I_r = P/(E_r*PF_r)*exp(%i*-acos(PF_r))                  // Line current(A)
sin_phi_r = (1-PF_r**2)**0.5                            // Sin _R
E_s = E_r+I_r*Z_line                                    // Sending end voltage(V)
reg = (abs(E_s)-abs(E_r))/abs(E_r)*100                  // Voltage regulation(%)
R = real(Z_line)                                        // Resistance of the load line(ohm)
loss = abs(I_r)**2*R                                    // Loss in the transmission line(W)
loss_kW = loss/1000.0                                   // Loss in the transmission line(kW)
P_s = P+loss                                            // Sending end power(W)
angle_Er_Es = phasemag(E_s)                             // Angle between V_r and V_s( )
angle_Er_Ir = acosd(PF_r)                               // Angle between V_r and I_r( )
angle_Es_Is = angle_Er_Es+angle_Er_Ir                   // Angle between V_s and I_s( )
PF_s = cosd(angle_Es_Is)
```

```
                                          // Sending end power factor

// Results
disp("PART II - EXAMPLE : 3.6 : SOLUTION :-")
printf("\nVoltage regulation = %.2f percent", reg)
printf("\nSending end voltage, E_s = %.f % .1f  V", abs(E_s),phasemag(E_s))
printf("\nLine loss = %.f kW", loss_kW)
printf("\nSending end power factor = %.2f ", PF_s)
```

Scilab code Exa 10.7 Nominal pi equivalent circuit parameters and Receiving end vo

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.7 :
// Page number 132-133
clear ; clc ; close ; // Clear the work space and console

// Given data
V_s = 66.0                    // Voltage(kV)
f = 50.0                      // Frequency(Hz)
l = 150.0                     // Line length(km)
r = 0.25                      // Resistance of each conductor(ohm/km)
x = 0.5                       // Inductive reactance of each conductor(ohm/km)
```

```
y = 0.04*10**-4                    // Capacitive
    admittance(s/km)

// Calculations
// Case(a)
R = r*l                                              //
    Total resistance(ohm)
X = x*l                                              //
    Inductive reactance(ohm)
Y = y*l                                              //
    Capacitive resistance(s)
Y_2 = Y/2                                            //
    1/2 of Capacitive resistance(s)
// Case(b)
Z = complex(R,X)                                     //
    Total impedance(ohm)
A = 1+(Y*exp(%i*90.0*%pi/180)*Z/2)                   //
    Line constant
V_R_noload = V_s/abs(A)                              //
    Receiving end voltage at no-load(kV)

// Results
disp("PART II - EXAMPLE : 3.7 : SOLUTION :-")
printf("\nCase(a): Total resistance, R = %.1f ohm",
    R)
printf("\n         Inductive reactance, X = %.1f ohm
    ", X)
printf("\n         Capacitive resistance, Y = %.1e s
    ", Y)
printf("\n         Capacitive resistance, Y/2 = %.1e
     s", Y_2)
printf("\nCase(b): Receiving end voltage at no-load,
     V_R = %.2f kV", V_R_noload)
```

Scilab code Exa 10.8 Voltage Current and Power factor at sending end

```
// A Textbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND
   PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.8 :
// Page number 133-134
clear ; clc ; close ; // Clear the work space and
   console

// Given data
f = 50.0                          // Frequency(Hz)
V_r = 132.0*10**3                 // Line voltage at
   receiving end(V)
L = 100.0                         // Line length(km)
r = 0.17                          // Resistance(ohm/km/
   phase)
l = 1.1*10**-3                    // Inductance(H/km/
   phase)
c = 0.0082*10**-6                 // Capacitance(F/km/
   phase)
P_L = 70.0*10**6                  // Load at receiving
   end(W)
PF_r = 0.8                        // Lagging load power
   factor

// Calculations
E_r = V_r/3**0.5
                                                  //
   Receiving end phase voltage(V)
I_r = P_L/(3**0.5*V_r*PF_r)*exp(%i*-acos(PF_r))
          // Receiving end current(A)
R = r*L
                                                          //
```

```
Total resistance(ohm/phase)
X = 2*%pi*f*l*L                                          // Inductive reactance(ohm/phase)
Z = complex(R,X)                                         // Total impedance(ohm/phase)
Y = 2*%pi*f*c*exp(%i*90.0*%pi/180)/L                     // Shunt admittance of line(mho/phase)
E = E_r+I_r*(Z/2)                                        // Voltage across shunt admittance(V/phase)
I_s = I_r+E*Y                                            // Sending end current(A)
E_s = E+I_s*(Z/2)                                        // Sending end voltage(V/phase)
E_s_ll = 3**0.5*abs(E_s)/1000                            // Sending end line to line voltage(kV)
angle_Er_Es = phasemag(E_s)                              // Angle between E_r and V_s( )
angle_Er_Is = phasemag(I_s)                              // Angle between E_r and I_s( )
angle_Es_Is = angle_Er_Es-angle_Er_Is                    // Angle between E_s and I_s( )
PF_s = cosd(angle_Es_Is)                                 // Sending end power factor

// Results
disp("PART II - EXAMPLE : 3.8 : SOLUTION :-")
printf("\nVoltage at sending end, E_s = %.2f % .2f V/phase = %.f kV (line-to-line)", abs(E_s),
```

```
phasemag(E_s),E_s_ll)
printf("\nCurrent at sending end, I_s = %.1f%.1f A", abs(I_s),phasemag(I_s))
printf("\nSending end power factor = %.3f (lagging)", PF_s)
```

Scilab code Exa 10.9 Sending end voltage Current and Transmission efficiency

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.9 :
// Page number 134
clear ; clc ; close ; // Clear the work space and console

// Given data
f = 50.0                         // Frequency(Hz)
E_r = 66.0*10**3                 // Line voltage at receiving end(V)
l = 120.0                        // Line length(km)
r = 0.1                          // Resistance(ohm/km/phase)
x = 0.3                          // Inductive reactance (ohm/km/phase)
y = 0.04*10**-4                  // Capacitive susceptance(S/km/phase)
P_L = 10.0*10**6                 // Load at receiving end(W)
```

```
PF_r = 0.8                              // Lagging load power factor

// Calculations
R = r*l                                 // Total resistance(ohm/phase)
X = x*l                                 // Inductive reactance(ohm/phase)
Y = y*l                                 // Susceptance(mho)
Z = complex(R,X)                        // Total impedance(ohm/phase)
V_r = E_r/3**0.5                        // Receiving end phase voltage(V)
I_r = P_L/(3**0.5*E_r*PF_r)*exp(%i*-acos(PF_r))     // Load current(A)
V_1 = V_r+I_r*(Z/2)                     // Voltage across capacitor(V)
I_c = %i*Y*V_1                          // Charging current(A)
I_s = I_r+I_c                           // Sending end current(A)
V_s = V_1+I_s*(Z/2)                     // Sending end voltage(V/phase)
V_s_ll = 3**0.5*abs(V_s)/1000.0         // Sending end line to line voltage(kV)
angle_Vr_Vs = phasemag(V_s)             // Angle
```

```
    between V_r and V_s(  )
angle_Vr_Is = phasemag(I_s)
                                                    // Angle
    between V_r and I_s(  )
angle_Vs_Is = angle_Vr_Vs-angle_Vr_Is
                                          // Angle between V_s
     and I_s(  )
PF_s = cosd(angle_Vs_Is)
                                                       //
    Sending end power factor
P_s = 3*abs(V_s*I_s)*PF_s
                                                      //
    Sending end power(W)
n = P_L/P_s*100

    // Transmission efficiency(%)

// Results
disp("PART II - EXAMPLE : 3.9 : SOLUTION :-")
printf("\nSending end voltage, |V_s| = %.f V/phase =
     %.3f V (line-to-line)", abs(V_s),V_s_ll)
printf("\nSending end current, |I_s| = %.2f A", abs(
    I_s))
printf("\nTransmission efficiency = %.2f percent \n"
    , n)
printf("\nNOTE: ERROR: Calculation mistake in
    finding sending end power factor")
printf("\n      Changes in the obtained answer from
    that of textbook is due to more precision")
```

Scilab code Exa 10.10 Line to line voltage and Power factor at sending end

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
```

```
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.10 :
// Page number 135
clear ; clc ; close ; // Clear the work space and console

// Given data
f = 50.0                          // Frequency(Hz)
l = 125.0                         // Line length(km)
P_r = 40.0*10**6                  // Load at receiving end(VA)
V_r = 110.0*10**3                 // Line voltage at receiving end(V)
PF_r = 0.8                        // Lagging load power factor
R = 11.0                          // Resistance(ohm/phase)
X = 38.0                          // Inductive reactance(ohm/phase)
Y = 3.0*10**-4                    // Capacitive susceptance(S)

// Calculations
// Case(i)
E_r = V_r/3**0.5                                    // Receiving end phase voltage(V)
Z = complex(R,X)                                    // Total impedance(ohm/phase)
I_c1 = E_r*(Y/2)*exp(%i*90.0*%pi/180)              // Current through shunt admittance at receiving end(A)
```

```
I_r = P_r/(3**0.5*V_r)*exp(%i*-acos(PF_r))                    // Load current(A)
I = I_r+I_c1                                                  // Current through series impedance(A)
E_s = I*Z+E_r                                                 // Voltage across shunt admittance at sending end(V)
E_s_ll = 3**0.5*E_s/1000.0                                    // Line to line voltage at sending end(kV)
I_c2 = E_s*(Y/2)*exp(%i*90.0*%pi/180)                         // Current through shunt admittance at sending end(A)
// Case(ii)
I_s = I_c2+I_r                                                // Sending end current(A)
angle_Er_Es = phasemag(E_s)                                   // Angle between E_r and E_s( )
angle_Er_Is = phasemag(I_s)                                   // Angle between E_r and I_s( )
angle_Es_Is = angle_Er_Es-angle_Er_Is                         // Angle between E_s and I_s( )
PF_s = cosd(angle_Es_Is)                                      // Sending end power factor

// Results
disp("PART II - EXAMPLE : 3.10 : SOLUTION :-")
printf("\nCase(i) : Line to line voltage at sending end, E_s = %.f kV", abs(E_s_ll))
printf("\nCase(ii): Sending end power factor = %.3f \n", PF_s)
printf("\nNOTE: Answers in the textbook are
```

```
incomplete")
```

Scilab code Exa 10.11 Voltage Current Power factor at sending end Regulation and T

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND
   PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.11 :
// Page number 135-137
clear ; clc ; close ; // Clear the work space and
    console

// Given data
f = 50.0                       // Frequency(Hz)
R = 28.0                       // Resistance(ohm/
    phasemag)
X = 63.0                       // Inductive reactance
    (ohm/phasemag)
Y = 4.0*10**-4                 // Capacitive
    susceptance(mho)
P_r = 75.0*10**6               // Load at receiving
    end(VA)
PF_r = 0.8                     // Lagging load power
    factor
V_r = 132.0*10**3              // Line voltage at
    receiving end(V)

// Calculations
// Case(i) Nominal T method
```

```
Z = complex(R,X)                                   // Total impedance(ohm/phasemag)
E_r = V_r/3**0.5                                   // Receiving end phasemag voltage(V)
I_r = P_r/(3**0.5*V_r)*exp(%i*-acos(PF_r))         // Line current at receiving end(A)
E = E_r+I_r*(Z/2)
I_c = %i*Y*E                                       // Capacitive current(A)
I_s = I_r+I_c                                      // Sending end current(A)
v_drop = I_s*(Z/2)                                 // Voltage drop(V)
E_s = E+I_s*(Z/2)                                  // Sending end voltage(V)
E_s_kV = E_s/1000.0                                // Sending end voltage(kV)
E_s_ll= 3**0.5*abs(E_s)                            // Sending end line voltage(V)
E_s_llkV = E_s_ll/1000.0                           // Sending end line voltage(kV)
angle_Er_Es = phasemag(E_s)                        // Angle between E_r and E_s( )
angle_Er_Is = phasemag(I_s)                        // Angle between E_r and I_s( )
angle_Es_Is = angle_Er_Es-angle_Er_Is
```

```
                                        // Angle between E_s
    and I_s(  )
PF_s = cosd(angle_Es_Is)                                     //
   Sending end power factor
P_s = 3**0.5*E_s_ll*abs(I_s)*PF_s                   // Power at
   sending end(W)
reg = (abs(E_s_ll)-V_r)/V_r*100                  // Regulation(
   %)
n = (P_r*PF_r)/P_s*100                                        //
   Transmission efficiency(%)
// Case(ii) Nominal    method
I_c2 = E_r*(%i*Y/2)                                             //
    Current through shunt admittance at receiving
   end(A)
I = I_r+I_c2
   // Line current(A)
E_s_p = E_r+I*Z
   // Sending end voltage(V)
E_s_pkV = E_s_p/1000.0                                        //
   Sending end voltage(kV)
E_s_pll = 3**0.5*abs(E_s_p)                        // Sending
    end line voltage(V)
E_s_pllkV = E_s_pll/1000.0                                //
   Sending end line voltage(kV)
I_c1 = E_s_p*(%i*Y/2)                                          //
   Current through shunt admittance at sending end(A
   )
```

```
I_s_p = I+I_c1                                             // Sending end current(A)
angle_Er_Esp = phasemag(E_s)                               // Angle between E_r and E_s(  )
angle_Er_Isp = phasemag(I_s)                               // Angle between E_r and I_s(  )
angle_Es_Isp = angle_Er_Esp-angle_Er_Isp                   // Angle between E_s and I_s(  )
PF_s_p = cosd(angle_Es_Isp)                                // Sending end power factor
P_s_p = 3**0.5*E_s_pll*abs(I_s_p)*PF_s_p                   // Power at sending end (W)
reg_p = (abs(E_s_pll)-V_r)/V_r*100                         // Regulation(%)
n_p = (P_r*PF_r)/P_s_p*100                                 // Transmission efficiency(%)

// Results
disp("PART II - EXAMPLE : 3.11 : SOLUTION :-")
printf("\n(i) Nominal T method")
printf("\nCase(a): Voltage at sending end, E_s = %.2f % .2f  kV = %.1f kV (line-to-line)", abs(E_s_kV),phasemag(E_s_kV),E_s_llkV)
printf("\nCase(b): Sending end current, I_s = %.1f % .2f  A", abs(I_s),phasemag(I_s))
printf("\nCase(c): Power factor at sending end = %.4f (lagging)", PF_s)
printf("\nCase(d): Regulation = %.2f percent", reg)
printf("\nCase(e): Efficiency of transmission = %.2f percent \n", n)
printf("\n(ii) Nominal   method")
```

```
printf("\nCase(a): Voltage at sending end, E_s = %.2f∠%.2f kV = %.1f kV (line-to-line)", abs(E_s_pkV),phasemag(E_s_pkV),E_s_pllkV)
printf("\nCase(b): Sending end current, I_s = %.1f∠%.2f A", abs(I_s_p),phasemag(I_s_p))
printf("\nCase(c): Power factor at sending end = %.4f (lagging)", PF_s_p)
printf("\nCase(d): Regulation = %.2f percent", reg_p)
printf("\nCase(e): Efficiency of transmission = %.2f percent \n", n_p)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here and more approximation in textbook")
```

Scilab code Exa 10.12 `Receiving end Voltage Load and Nature of compensation requir`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.12 :
// Page number 143
clear ; clc ; close ; // Clear the work space and console

// Given data
E_s = 275.0                    // Sending end voltage(kV)
f = 50.0                       // Frequency(Hz)
```

```
l = 400.0                    // Line length(km)
x = 0.05                     // Inductive reactance(ohm/km)
y = 3.0*10**-6               // Line charging susceptance(S/km)
r = 0.0                      // Lossless line

// Calculations
// Case(a)
R = r*l                      // Total resistance(ohm/phase)
X = x*l                      // Inductive reactance(ohm/phase)
Y = y*l                      // Susceptance(mho)
Z = complex(R,X)             // Total impedance(ohm/phase)
A = 1+(Y*Z/2)*%i             // Line constant
E_r = E_s/abs(A)             // Receiving end voltage at no load(kV)
// case(b)
Z_0 = (X/Y)**0.5             // Load at receiving end (ohm)
// Case(c)
Z_0_new = 1.2*Z_0            // New load at receiving station(ohm)

// Results
disp("PART II - EXAMPLE : 3.12 : SOLUTION :-")
printf("\nCase(a): Receiving end voltage on open circuit = %.1f kV", E_r)
printf("\nCase(b): Load at receiving end for flat voltage profile on line, Z_0 = %.1f  ", Z_0)
printf("\nCase(c): Distributed inductive reactance of the line is to be increased as, Loading for new voltage profile = %.2f  ", Z_0_new)
```

Scilab code Exa 10.13 Sending end voltage and Current

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.13 :
// Page number 143-144
clear ; clc ; close ; // Clear the work space and console

// Given data
V_r = 220.0*10**3             // Receiving end voltage(V)
Z = complex(20,100)           // Impedance(ohm/phase)
Y = %i*0.0010                 // Admittance(mho)
I_r = 300.0                   // Receiving end current(A)
PF_r = 0.9                    // Lagging power factor

// Calculations
V_2 = V_r/3**0.5                              // Receiving end phase voltage(V)
I_2 = I_r*exp(%i*-acos(PF_r))                 // Receiving end current(A)
I_C2 = (Y/2)*V_2                              // Capacitive current at receiving end(A)
I = I_2+I_C2
V_1 = V_2+I*Z                                 //
```

```
Voltage across shunt admittance at sending end(V)
V_1kV = V_1/1000.0                              // Voltage across shunt admittance at sending end(kV)
I_C1 = (Y/2)*V_1                                // Capacitive current at sending end(A)
I_1 = I_C1+I_2                                  // Sending end current(A)

// Results
disp("PART II - EXAMPLE : 3.13 : SOLUTION :-")
printf("\nSending end voltage, V_1 = %.2f %.2f kV", abs(V_1kV),phasemag(V_1kV))
printf("\nSending end current, I_1 = %.3f %.4f  A", abs(I_1),phasemag(I_1))
```

Scilab code Exa 10.14 `Incident voltage and Reflected voltage at receiving end and`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.14 :
// Page number 144
clear ; clc ; close ; // Clear the work space and console
funcprot(0)

// Given data
f = 50.0                        // Frequency(Hz)
```

```
r = 0.1                              // Resistance(ohm/km)
l = 1.4*10**-3                       // Inductance(H/km)
c = 8.0*10**-9                       // Capacitance(F/km)
g = 4.0*10**-8                       // conductance(mho/km)
V_r = 400.0                          // Receiving end voltage(kV)
x = 200.0                            // Length of line(km)

// Calculations
V_2 = V_r/3**0.5                                // Receiving end phase voltage(kV)
z = r+%i*2*%pi*f*l                              // Total impedance(ohm/km)
y = g+%i*2*%pi*f*c                              // Total susceptance(mho/km)
Z_c = (z/y)**0.5                                // Surge impedance(ohm)
gamma = (z*y)**0.5                              //
// Case(i)
V_0_plus = V_2/2                                // Incident voltage to neutral at receiving end(kV)
// Case(ii)
V_0_minus = V_2/2                               // Reflected voltage to neutral at receiving end(kV)
// Case(iii)
gamma_l = gamma*x                               //  l
V_1_plus = (V_2/2)*exp(gamma_l)                 // Incident voltage to neutral at 200 km from receiving end(kV)
V_1_minus = (V_2/2)*exp(-gamma_l)               // Reflected voltage to neutral at 200 km from receiving end(kV)
// Case(iv)
V_1 = V_1_plus+V_1_minus                        // Resultant voltage to neutral(kV)
V_L = abs(V_1)                                  // Resultant voltage to neutral(kV)
V_L_ll = 3**0.5*V_L                             // Line to
```

```
line voltage at 200 km from receiving end(kV)

// Results
disp("PART II - EXAMPLE : 3.14 : SOLUTION :-")
printf("\nCase(i)  : Incident voltage to neutral at receiving end, V_0_plus = %.1f %.f  kV", abs(V_0_plus),phasemag(V_0_plus))
printf("\nCase(ii) : Reflected voltage to neutral at receiving end, V_0_minus = %.1f %.f  kV", abs(V_0_minus),phasemag(V_0_minus))
printf("\nCase(iii): Incident voltage to neutral at 200 km from receiving end, V_1_plus = (%.3f+%.2fj) kV", real(V_1_plus),imag(V_1_plus))
printf("\nCase(iv) : Resultant voltage to neutral at 200 km from receiving end, V_L = %.2f kV", V_L)
printf("\n           Line to line voltage at 200 km from receiving end = %.2f kV", V_L_ll)
```

Scilab code Exa 10.15 A B C D constants

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.15 :
// Page number 145
clear ; clc ; close ; // Clear the work space and console

// Given data
```

```
f = 50.0                          // Frequency(Hz)
L = 200.0                         // Line length(km)
l = 1.20*10**-3                   // Inductance(H/km)
c = 8.0*10**-9                    // Capacitance(F/km)
r = 0.15                          // Resistance(ohm/km)
g = 0.0                           // Conductance(mho/km)

// Calculations
z = r+%i*2*%pi*f*l                // Total impedance(ohm/km)
Z = z*L                           // Total impedance(ohm)
y = g+%i*2*%pi*f*c                // Total susceptance(mho/km)
Y = y*L                           // Total susceptance(mho/km)
gamma_l = (Z*Y)**0.5              // l
alpha_l = real(gamma_l)           // l
beta_l = imag(gamma_l)            // l
Z_c = (Z/Y)**0.5                  // Surge impedance(ohm)
A = cosh(gamma_l)                 // Constant
B = Z_c*sinh(gamma_l)             // Constant(ohm)
C = (1/Z_c)*sinh(gamma_l)         // Constant(S)
D = A                             // Constant

// Results
disp("PART II - EXAMPLE : 3.15 : SOLUTION :-")
printf("\nA = D = %.3 f % .2 f   ", abs(A),phasemag(A))
printf("\nB = %.2 f % .3 f      ", abs(B),phasemag(B))
printf("\nC = %.2 e % .3 f   S", abs(C),phasemag(C))
```

Scilab code Exa 10.16 `Sending end voltage Current Power factor and Efficiency`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.16 :
// Page number 145-146
clear ; clc ; close ; // Clear the work space and console
funcprot(0)

// Given data
V_r = 132.0*10**3             // Receiving end voltage(V)
f = 50.0                      // Frequency(Hz)
L = 200.0                     // Line length(km)
l = 1.3*10**-3                // Inductance(H/km)
c = 9.0*10**-9                // Capacitance(F/km)
r = 0.2                       // Resistance(ohm/km)
g = 0.0                       // Conductance(mho/km)
P_r = 50.0*10**6              // Power received(VA)
PF_r = 0.8                    // Lagging power factor at receiving end

// Calculations
z = r+%i*2*%pi*f*l                                  // Total impedance(ohm/km)
y = g+%i*2*%pi*f*c                                  // Total susceptance(mho/km)
Z_c = (z/y)**0.5                                    // Surge impedance(ohm)
gamma = (z*y)**0.5                                  //
```

```
gamma_l = gamma*L                                   // l
cosh_gl = cosh(gamma_l)                             // cosh  l
sinh_gl = sinh(gamma_l)                             // sinh  l
V_2 = V_r/(3**0.5)                                  // Receiving end phase voltage(V)
I_2 = P_r/(3*V_2)*exp(%i*-acos(PF_r))               // Line current(A)
V_1 = V_2*cosh_gl+I_2*Z_c*sinh_gl                   // Sending end voltage(V)
V_1kV = V_1/1000.0                                  // Sending end voltage(kV)
I_1 = (V_2/Z_c)*sinh_gl+I_2*cosh_gl                 // Sending end current(A)
angle_V2_V1 = phasemag(V_1)                         // Angle between V_2 and V_1(  )
angle_V2_I1 = phasemag(I_1)                         // Angle between V_2 and I_1(  )
angle_V1_I1 = angle_V2_V1-angle_V2_I1               // Angle between V_1 and I_1(  )
PF_s = cosd(angle_V1_I1)                            // Sending end power factor
P_1 = 3*abs(V_1*I_1)*PF_s                           // Sending end power(W)
P_2 = P_r*PF_r                                      // Receiving end power(W)
n = P_2/P_1*100                                     // Efficiency

// Results
disp("PART II - EXAMPLE : 3.16 : SOLUTION :-")
printf("\nSending end voltage, V_1 = %.3f%.4f kV per phase", abs(V_1kV),phasemag(V_1kV))
printf("\nSending end current, I_1 = %.3f%.2f  A", abs(I_1),phasemag(I_1))
```

```
printf("\nPower factor = %.3f ", PF_s)
printf("\nEfficiency ,     = %.2f percent", n)
```

Scilab code Exa 10.17 Values of auxiliary constants A B C D

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.17 :
// Page number 147-148
clear ; clc ; close ; // Clear the work space and console

// Given data
f = 50.0                          // Frequency(Hz)
L = 160.0                         // Line length(km)
r = 0.15                          // Resistance(ohm/km/phasemag)
l = 1.2*10**-3                    // Inductance(H/km/phasemag)
c = 0.008*10**-6                  // Capacitance(F/km/phasemag)
g = 0.0                           // Conductance(mho/km/phasemag)

// Calculations
// Case(i) Using convergent series(Complex angles) method
z = r+%i*2*%pi*f*l                                  //
```

```
    Impedance(ohm/km)
Z = z*L                                          // Total series impedance(ohm)
y = g+%i*2*%pi*f*c                               // Shunt admittance(S/km)
Y = y*L                                          // Total shunt admittance(S)
A = 1+(Y*Z/2)+((Y*Z)**2/24)                      // Constant
B = Z*(1+(Y*Z/6)+((Y*Z)**2/120))                 // Constant(ohm)
C = Y*(1+(Y*Z/6)+((Y*Z)**2/120))                 // Constant(mho)
D = A                                            // Constant
// Case(ii) Using convergent series(Real angles) method
gamma_l = (Z*Y)**0.5                             //  l
alpha_l = real(gamma_l)                          //  l
beta_l = imag(gamma_l)                           //  l
Z_c = (Z/Y)**0.5                                 // Surge impedance(ohm)
A_2 = cosh(gamma_l)                              // Constant
B_2 = Z_c*sinh(gamma_l)                          // Constant(ohm)
C_2 = (1/Z_c)*sinh(gamma_l)                      // Constant(mho)
D_2 = A_2                                        // Constant

// Results
disp("PART II - EXAMPLE : 3.17 : SOLUTION :-")
printf("\nCase(i): Using convergent series(Complex Angles) method")
printf("\nA = D = %.3f % .1f  ", abs(A),phasemag(A))
printf("\nB = %. f % .1f  ohm", abs(B),phasemag(B))
```

```
printf("\nC = %.4 f % .1 f   mho \n", abs(C),phasemag(C))
printf("\nCase(ii): Using convergent series(Real Angles) method")
printf("\nA = D = %.3 f % .1 f   ", abs(A_2),phasemag(A_2))
printf("\nB = %.1 f % .1 f   ohm", abs(B_2),phasemag(B_2))
printf("\nC = %.4 f % .1 f   S \n", abs(C_2),phasemag(C_2))
printf("\nNOTE: Slight change in obtained answer from that of textbook is due to more precision")
```

Scilab code Exa 10.18 Sending end voltage and Current using convergent series meth

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.18 :
// Page number 148
clear ; clc ; close ; // Clear the work space and console

// Given data
V_r = 220.0*10**3                    // Line voltage at receiving end(V)
Z = complex(40,200)                  // Impedance per phasemag(ohm)
Y = %i*0.0015                        // Admittance(
```

```
mho)
I_r = 200.0                              // Receiving end current(A)
PF_r = 0.95                              // Lagging power factor

// Calculations
// Case(a)
A = 1+(Y*Z/2)+((Y*Z)**2/24)                              // Constant
B = Z*(1+(Y*Z/6)+((Y*Z)**2/120)+((Y*Z)**3/5040))          // Constant(ohm)
C = Y*(1+(Y*Z/6)+((Y*Z)**2/120)+((Y*Z)**3/5040))          // Constant(mho)
D = A                                                    // Constant
E_r = V_r/3**0.5                                         // Receiving end phasemag voltage(V)
I_r1 = I_r*exp(%i*-acos(PF_r))                           // Line current(A)
E_s = A*E_r+B*I_r1                                       // Sending end voltage(V)
E_s_ll = 3**0.5*E_s/1000.0                               // Sending end line voltage(kV)
// Case(b)
I_s = C*E_r+D*I_r1                                       // Sending end current(A)

// Results
disp("PART II - EXAMPLE : 3.18 : SOLUTION :-")
printf("\nCase(a): Sending end voltage, E_s = %.1f %.2f  kV (line-to-line)", abs(E_s_ll), phasemag(E_s_ll))
```

```
printf("\nCase(b): Sending end current, I_s = %.1f % .2f  A\n", abs(I_s),phasemag(I_s))
printf("\nNOTE: ERROR: Z = (40+j200)  , not Z=(60+j200)  as given in problem statement")
printf("\n      Changes in obtained answer from that of textbook is due to more precision")
```

Scilab code Exa 10.19 Sending end voltage and Current using nominal pi and nominal

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.19 :
// Page number 148-149
clear ; clc ; close ; // Clear the work space and console

// Given data
V_r = 220.0*10**3                    // Line voltage at receiving end(V)
Z = complex(40,200)                  // Impedance per phasemag(ohm)
Y = %i*0.0015                        // Admittance(S)
I_R = 200.0                          // Receiving end current(A)
PF_r = 0.95                          // Lagging power factor

```

```
// Calculations
// Case(i) Nominal   method
// Case(a)
E_r = V_r/3**0.5                                 // Receiving end phasemag voltage(V)
I_r = I_R*exp(%i*-acos(PF_r))      // Line current(A)
Y_2 = Y/2.0                                      // Admittance(S)
I_c2 = Y_2*E_r                                   // Current through shunt admittance at receiving end(A)
I = I_r+I_c2                                     // Current through impedance(A)
IZ_drop = I*Z                                    // Voltage drop(V)
E_s = E_r+IZ_drop                                // Sending end voltage(V)
E_s_kV = E_s/1000.0                              // Sending end voltage(kV)
// Case(b)
I_c1 = E_s*Y_2                                   // Current through shunt admittance at sending end(A)
I_s = I+I_c1                                     // Sending end current(A)
// Case(ii) Nominal T method
// Case(a)
I_r_Z2 = I_r*Z/2                                 // Voltage drop at receiving end(V)
E = E_r+I_r_Z2                                   // Voltage(V)
I_c = Y*E                                        // Current through shunt admittance(A)
I_s_2 = I_c+I_r                                  // Sending end current(A)
I_s_Z2 = I_s_2*(Z/2)                             // Voltage drop at sending end(V)
E_s_2 = I_s_Z2+E                                 //
```

```
Sending end voltage(V)
E_s_2kV = E_s_2/1000.0                                  // Sending end voltage(kV)

// Results
disp("PART II - EXAMPLE : 3.19 : SOLUTION :-")
printf("\nCase(i): Nominal    method")
printf("\n            Case(a): Sending end voltage, E_s = %.1f %.2f  kV", abs(E_s_kV),phasemag(E_s_kV))
printf("\n            Case(b): Sending end current, I_s = %.1f %.2f  A", abs(I_s),phasemag(I_s))
printf("\nCase(ii): Nominal T method")
printf("\n            Case(a): Sending end voltage, E_s = %.1f %.2f  kV", abs(E_s_2kV),phasemag(E_s_2kV))
printf("\n            Case(b): Sending end current, I_s = %.1f %.2f  A \n", abs(I_s_2),phasemag(I_s_2))
printf("\nThe results are tabulated below")
printf("\n______________________________________________________")
printf("\nMETHOD              E_s(kV)              I_s(A)")
printf("\n______________________________________________________")
printf("\nRigorous           3 *132.6  16 .46           209.8  39 .42  ")
printf("\nNominal             3 *%.1f % .2f          %.1f % .2f  ", abs(E_s_kV),phasemag(E_s_kV),abs(I_s),phasemag(I_s))
printf("\nNominal T          3 *%.1f % .2f   %.1f % .2f  ", abs(E_s_2kV),phasemag(E_s_2kV),abs(I_s_2),phasemag(I_s_2))
printf("\n______________________________________________________
```

```
")
```

Scilab code Exa 10.20 Sending end voltage Voltage regulation Transmission efficien

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND
   PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.20 :
// Page number 149-153
clear ; clc ; close ; // Clear the work space and
   console

// Given data
f = 50.0                    // Frequency(Hz)
L = 280.0                   // Line length(km)
Z = complex(35,140)         // Series impedance(ohm)
Y = %i*930.0*10**-6         // Shunt admittance(S)
P_r = 40.0*10**6            // Power delivered(W)
V_r = 220.0*10**3           // Voltage at receiving
   end(V)
PF_r = 0.9                  // Lagging power factor

// Calculations
R = real(Z)

   // Resistance of the line(ohm)
// Case(a)
I_r_a = P_r/(3**0.5*V_r*PF_r)*exp(%i*-acos(PF_r))
   // Receiving end current(A)
```

```
I_s_a = I_r_a                                    // Sending end current(A)
V_r_a = V_r/3**0.5                               // phasemag voltage at receiving end(V)
V_s_a = V_r_a+I_r_a*Z                            // Sending end voltage(V)
V_s_a_ll = 3**0.5*V_s_a                          // Sending end line voltage(V)
V_s_a_llkv = V_s_a_ll/1000.0                     // Sending end line voltage(kV)
reg_a = (abs(V_s_a_ll)-V_r)/V_r*100              // Voltage regulation(%)
loss_a = 3*abs(I_r_a)**2*R                       // Line loss(W)
input_a = P_r+loss_a                             // Input to line(W)
n_a = P_r/input_a*100                            // Efficiency of transmission(%)
A_a = 1.0                                        // Constant
B_a = Z                                          // Constant(ohm)
C_a = 0                                          // Constant(mho)
D_a = A_a
```

```
    // Constant
// Case(b)
V_b = V_r_a+I_r_a*Z/2                                    // Voltage drop across shunt admittance(V)
I_c_b = Y*V_b                                            // Current through shunt admittance(A)
I_s_b = I_r_a+I_c_b                                      // Sending end current(A)
V_s_b = V_b+I_s_b*Z/2                                    // Sending end voltage(V)
V_s_b_ll = 3**0.5*V_s_b                                  // Sending end line voltage(V)
V_s_b_llkv = V_s_b_ll/1000.0                             // Sending end line voltage(kV)
angle_V_Is_b = phasemag(I_s_b)                           // Angle between V_r and I_s_b( )
angle_V_Vs_b = phasemag(V_s_b)                           // Angle between V_r and V_s_b( )
angle_Is_Vs_b = angle_V_Is_b-angle_V_Vs_b                // Angle between V_s_b and I_s_b( )
PF_s_b = cosd(angle_Is_Vs_b)                             // Sending end power factor
P_s_b = 3**0.5*abs(V_s_b_ll*I_s_b)*PF_s_b                // Sending end power(W)
n_b = P_r/P_s_b*100                                      // Efficiency of transmission(%)
reg_b = (abs(V_s_b_ll)-V_r)/V_r*100                      // Voltage regulation(%)
```

```
A_b = 1+(1.0/2)*Y*Z                                   // Constant
B_b = Z*(1+(1.0/4)*Y*Z)                               // Constant(ohm)
C_b =Y                                                // Constant(mho)
D_b = A_b                                             // Constant
// Alternative solution for case(b)
V_s_ba = A_b*V_r_a+B_b*I_r_a                          // Sending end voltage(V)
V_s_ba_ll = 3**0.5*V_s_ba                             // Sending end line voltage(V)
V_s_ba_llkv = V_s_ba_ll/1000.0                        // Sending end line voltage(kV)
I_s_ba = C_b*V_r_a+D_b*I_r_a                          // Sending end current(A)
angle_V_Is_ba = phasemag(I_s_ba)                      // Angle between V_r and I_s_b( )
angle_V_Vs_ba = phasemag(V_s_ba)                      // Angle between V_r and V_s_b( )
angle_Is_Vs_ba = angle_V_Is_ba-angle_V_Vs_ba          // Angle between V_s_b and I_s_b( )
PF_s_ba = cosd(angle_Is_Vs_ba)                        // Sending end power factor
P_s_ba = 3**0.5*abs(V_s_ba_ll*I_s_ba)*PF_s_ba         // Sending end power(W)
n_ba = P_r/P_s_ba*100                                 // Efficiency of
```

```
    transmission(%)
reg_ba = (abs(V_s_ba_ll)-V_r)/V_r*100                    // Voltage regulation(%)
// Case(c)
I_c2_c = Y/2.0*V_r_a                                     // Current through shunt admittance at receiving end(A)
I_c = I_r_a+I_c2_c                                       // Current through impedance(A)
V_s_c = V_r_a+I_c*Z                                      // Sending end voltage(V)
V_s_c_ll = 3**0.5*V_s_c                                  // Sending end line voltage(V)
V_s_c_llkv = V_s_c_ll/1000.0                             // Sending end line voltage(kV)
I_c1_c = V_s_c*Y/2.0                                     // Current through shunt admittance at sending end(A)
I_s_c = I_c+I_c1_c                                       // Sending end current(A)
angle_V_Is_c = phasemag(I_s_c)                           // Angle between V_r and I_s_c( )
angle_V_Vs_c = phasemag(V_s_c)                           // Angle between V_r and V_s_c( )
angle_Is_Vs_c = angle_V_Is_c-angle_V_Vs_c                // Angle between V_s_c and I_s_c( )
PF_s_c = cosd(angle_Is_Vs_c)                             // Sending end power factor
P_s_c = 3**0.5*abs(V_s_c_ll*I_s_c)*PF_s_c                // Sending end power(W)
n_c = P_r/P_s_c*100                                      // Efficiency of transmission(%)
reg_c = (abs(V_s_c_ll)-V_r)/V_r*100                      // Voltage regulation(%)
A_c = 1+(1.0/2)*Y*Z                                      // Constant
B_c = Z                                                  // Constant(ohm)
```

```
C_c =Y*(1+(1.0/4)*Y*Z)                    // Constant(mho)
D_c = A_c                                 // Constant
// Alternative solution for case(c)
V_s_ca = A_c*V_r_a+B_c*I_r_a              // Sending end voltage(V)
V_s_ca_ll = 3**0.5*V_s_ca                 // Sending end line voltage(V)
V_s_ca_llkv = V_s_ca_ll/1000.0            // Sending end line voltage(kV)
I_s_ca = C_c*V_r_a+D_c*I_r_a              // Sending end current(A)
angle_V_Is_ca = phasemag(I_s_ca)          // Angle between V_r and I_s_c(  )
angle_V_Vs_ca = phasemag(V_s_ca)          // Angle between V_r and V_s_c(  )
angle_Is_Vs_ca = angle_V_Is_ca-angle_V_Vs_ca    // Angle between V_s_b and I_s_c(  )
PF_s_ca = cosd(angle_Is_Vs_ca)            // Sending end power factor
P_s_ca = 3**0.5*abs(V_s_ca_ll*I_s_ca)*PF_s_ca   // Sending end power(W)
n_ca = P_r/P_s_ca*100                     // Efficiency of transmission(%)
reg_ca = (abs(V_s_ca_ll)-V_r)/V_r*100     // Voltage regulation(%)
// Case(d).(i)
gamma_l = (Y*Z)**0.5                      //  l
Z_c = (Z/Y)**0.5                          // Surge impedance(ohm)
V_s_d1 = V_r_a*cosh(gamma_l)+I_r_a*Z_c*sinh(gamma_l)    // Sending end voltage(V)
V_s_d1_ll = 3**0.5*V_s_d1                 //
```

```
Sending end line voltage(V)
V_s_d1_llkv = V_s_d1_ll/1000.0                // Sending end line voltage(kV)
I_s_d1 = V_r_a/Z_c*sinh(gamma_l)+I_r_a*cosh(gamma_l)    // Sending end current(A)
angle_V_Is_d1 = phasemag(I_s_d1)              // Angle between V_r and I_s_d( )
angle_V_Vs_d1 = phasemag(V_s_d1)              // Angle between V_r and V_s_d( )
angle_Is_Vs_d1 = angle_V_Is_d1-angle_V_Vs_d1  // Angle between V_s_d and I_s_d( )
PF_s_d1 = cosd(angle_Is_Vs_d1)                // Sending end power factor
P_s_d1 = 3**0.5*abs(V_s_d1_ll*I_s_d1)*PF_s_d1 // Sending end power(W)
n_d1 = P_r/P_s_d1*100                         // Efficiency of transmission(%)
reg_d1 = (abs(V_s_d1_ll)-V_r)/V_r*100         // Voltage regulation(%)
A_d1 = cosh(gamma_l)                          // Constant
B_d1 = Z_c*sinh(gamma_l)                      // Constant(ohm)
C_d1 = (1/Z_c)*sinh(gamma_l)                  // Constant(mho)
D_d1 = A_d1                                   // Constant
```

```
// Case(d).(ii)
A_d2 = (1+(Y*Z/2)+((Y*Z)**2/24.0))                            // Constant
B_d2 = Z*(1+(Y*Z/6)+((Y*Z)**2/120))                           // Constant(ohm)
C_d2 = Y*(1+(Y*Z/6)+((Y*Z)**2/120))                           // Constant(mho)
D_d2 = A_d2                                                    // Constant
V_s_d2 = A_d2*V_r_a+B_d2*I_r_a                                // Sending end voltage(V)
V_s_d2_ll = 3**0.5*V_s_d2                                      // Sending end line voltage(V)
V_s_d2_llkv = V_s_d2_ll/1000.0                                 // Sending end line voltage(kV)
I_s_d2 = C_d2*V_r_a+D_d2*I_r_a                                // Sending end current(A)
angle_V_Is_d2 = phasemag(I_s_d2)                               // Angle between V_r and I_s_d( )
angle_V_Vs_d2 = phasemag(V_s_d2)                               // Angle between V_r and V_s_d( )
angle_Is_Vs_d2 = angle_V_Is_d2-angle_V_Vs_d2                   // Angle between V_s_d and I_s_d( )
PF_s_d2 = cosd(angle_Is_Vs_d2)                                 // Sending end power factor
P_s_d2 = 3**0.5*abs(V_s_d2_ll*I_s_d2)*PF_s_d2                  // Sending end power(W)
n_d2 = P_r/P_s_d2*100                                          // Efficiency of transmission(%)
```

```
reg_d2 = (abs(V_s_d2_ll)-V_r)/V_r*100                    // Voltage regulation(%)

// Results
disp("PART II - EXAMPLE : 3.20 : SOLUTION :-")
printf("\nCase(a): Short line approximation")
printf("\nSending end voltage, V_s = %.1f % .1f kV (line-to-line)", abs(V_s_a_llkv),phasemag(V_s_a_llkv))
printf("\nVoltage regulation = %.1f percent", reg_a)
printf("\nTransmission efficiency,   = %.1f percent", n_a)
printf("\nA = D = %.f ", A_a)
printf("\nB = %.1f % .1f  ohm", abs(B_a),phasemag(B_a))
printf("\nC = %.f \n", C_a)
printf("\nCase(b): Nominal T method approximation")
printf("\nSending end voltage, V_s = %.1f % .1f kV (line-to-line)", abs(V_s_b_llkv),phasemag(V_s_b_llkv))
printf("\nVoltage regulation = %.2f percent", reg_b)
printf("\nTransmission efficiency,   = %.1f percent", n_b)
printf("\nA = D = %.3f % .2f  ", abs(A_b),phasemag(A_b))
printf("\nB = %.1f % .1f  ohm", abs(B_b),phasemag(B_b))
printf("\nC = %.2e % .f  S ", abs(C_b),phasemag(C_b))
printf("\n\tALTERNATIVE SOLUTION:")
printf("\n\tSending end voltage, V_s = %.1f % .1f kV (line-to-line)", abs(V_s_ba_llkv),phasemag(V_s_ba_llkv))
printf("\n\tVoltage regulation = %.2f percent", reg_ba)
printf("\n\tTransmission efficiency,   = %.1f percent", n_ba)
printf("\n\tA = D = %.3f % .2f  ", abs(A_b),
```

```
    phasemag(A_b))
printf("\n\tB = %.1f % .1f   ohm", abs(B_b), phasemag(B_b))
printf("\n\tC = %.2e % .f   S \n", abs(C_b), phasemag(C_b))
printf("\nCase(c): Nominal    method approximation")
printf("\nSending end voltage, V_s = %. f % .1f   kV (line-to-line)", abs(V_s_c_llkv),phasemag(V_s_c_llkv))
printf("\nVoltage regulation = %.2f percent", reg_c)
printf("\nTransmission efficiency,    = %.1f percent", n_c)
printf("\nA = D = %.3f % .2f   ", abs(A_c),phasemag(A_c))
printf("\nB = %.1f % .1f   ohm", abs(B_c),phasemag(B_c))
printf("\nC = %.2e % .1f   mho", abs(C_c),phasemag(C_c))
printf("\n\tALTERNATIVE SOLUTION:")
printf("\n\tSending end voltage, V_s = %.1f % .1f   kV (line-to-line)", abs(V_s_ca_llkv),phasemag(V_s_ca_llkv))
printf("\n\tVoltage regulation = %.2f percent", reg_ca)
printf("\n\tTransmission efficiency,    = %.1f percent", n_ca)
printf("\n\tA = D = %.3f % .2f   ", abs(A_c), phasemag(A_c))
printf("\n\tB = %.1f % .1f   ohm", abs(B_c), phasemag(B_c))
printf("\n\tC = %.2e % .f   S \n", abs(C_c), phasemag(C_c))
printf("\nCase(d): Long Line Rigorous Solution")
printf("\n Case(i): Using Convergent Series (Real Angles) Method")
printf("\n Sending end voltage, V_s = %. f % .1f   kV (line-to-line)", abs(V_s_d1_llkv),phasemag(V_s_d1_llkv))
```

```
printf("\n Voltage regulation = %.2f percent", reg_d1)
printf("\n Transmission efficiency ,    = %.1f percent", n_d1)
printf("\n A = D = %.3 f % .2 f   ", abs(A_d1), phasemag(A_d1))
printf("\n B = %. f % .1 f   ohm", abs(B_d1),phasemag(B_d1))
printf("\n C = %.2 e % .1 f   mho \n", abs(C_d1), phasemag(C_d1))
printf("\n Case(ii): Using Convergent Series (Complex Angles) Method")
printf("\n Sending end voltage , V_s = %. f % .1 f kV (line-to-line)", abs(V_s_d2_llkv),phasemag(V_s_d2_llkv))
printf("\n Voltage regulation = %.2f percent", reg_d2)
printf("\n Transmission efficiency ,    = %.1f percent", n_d2)
printf("\n A = D = %.3 f % .2 f   ", abs(A_d2), phasemag(A_d2))
printf("\n B = %.1 f % .1 f   ohm", abs(B_d2), phasemag(B_d2))
printf("\n C = %.2 e % .1 f   mho \n", abs(C_d2), phasemag(C_d2))
printf("\nNOTE: Changes in obtained answer from that of textbook is due to more precision")
```

Scilab code Exa 10.21 Sending end voltage Current Power factor and Efficiency of t

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.21 :
// Page number 153
clear ; clc ; close ; // Clear the work space and console

// Given data
V_r = 132.0*10**3                        // Line voltage at receiving end(V)
P_L = 45.0*10**6                         // Load delivered(VA)
PF_r = 0.8                               // Lagging power factor
A = 0.99*exp(%i*0.3*%pi/180)             // Constant
B = 70.0*exp(%i*69.0*%pi/180)            // Constant(ohms)
C = A                                    // Constant
D = 4.0*10**-4*exp(%i*90.0*%pi/180)      // Constant

// Calculations
E_r = V_r/3**0.5                         // Receiving end phasemag voltage(V)
I_r = P_L/(3**0.5*V_r)*exp(%i*-acos(PF_r))    // Line current(A)
E_s = A*E_r+B*I_r                        // Sending end voltage(V)
E_s_llkV = 3**0.5*E_s/1000.0             // Sending end line voltage(kV)
I_s = C*I_r+D*E_r
```

```
    // Sending end current(A)
angle_Er_Es = phasemag(E_s)    // Angle between E_r and E_s(  )
angle_Er_Is = phasemag(I_s)    // Angle between E_r and I_s(  )
angle_Es_Is = angle_Er_Es-angle_Er_Is    // Angle between E_s and I_s(  )
PF_s = cosd(angle_Es_Is)    // Sending end power factor
P_s = 3*abs(E_s*I_s)*PF_s    // Sending end power(W)
P_skW = P_s/1000.0    // Sending end power(kW)
P_r = P_L*PF_r    // Receiving end power(W)
n = P_r/P_s*100    // Transmission efficiency(%)

// Results
disp("PART II - EXAMPLE : 3.21 : SOLUTION :-")
printf("\nCase(i)  : Sending end voltage, E_s = %.1f % .f  kV (line-to-line)", abs(E_s_llkV), phasemag(E_s_llkV))
printf("\nCase(ii) : Sending end current, I_s = %.1f % .1f  A", abs(I_s),phasemag(I_s))
printf("\nCase(iii): Sending end power, P_s = %.f kW", P_skW)
printf("\nCase(iv) : Efficiency of transmission = %.2f percent \n", n)
```

```
printf("\nNOTE: Changes in obtained answer from that
    textbook is due to more precision")
```

Scilab code Exa 10.23 Overall constants A B C D

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND
    PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.23 :
// Page number 156
clear ; clc ; close ; // Clear the work space and
    console

// Given data
A_1 = 0.98*exp(%i*2.0*%pi/180)          // Constant
    of 1st line
B_1 = 28.0*exp(%i*69.0*%pi/180)         // Constant
    of 1st line(ohms)
C_1 = 0.0002*exp(%i*88.0*%pi/180)       // Constant
    of 1st line(mho)
D_1 = A_1                               // Constant
    of 1st line
A_2 = 0.95*exp(%i*3.0*%pi/180)          // Constant
    of 2nd line
B_2 = 40.0*exp(%i*85.0*%pi/180)         // Constant
    of 2nd line(ohms)
C_2 = 0.0004*exp(%i*90.0*%pi/180)       // Constant
    of 2nd line(mho)
D_2 = A_2                               // Constant
```

```
    of 2nd line

// Calculations
A = A_1*A_2+B_1*C_2          // Constant
B = A_1*B_2+B_1*D_2          // Constant(ohm)
C = C_1*A_2+D_1*C_2          // Constant(mho)
D = C_1*B_2+D_1*D_2          // Constant

// Results
disp("PART II - EXAMPLE : 3.23 : SOLUTION :-")
printf("\nA = %.3f % .1f   ", abs(A),phasemag(A))
printf("\nB = %.1f % .f   ohm", abs(B),phasemag(B))
printf("\nC = %.6f % .1f   mho", abs(C),phasemag(C))
printf("\nD = %.3f % .1f   ", abs(D),phasemag(D))
```

Scilab code Exa 10.24 Values of constants A0 B0 C0 D0

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.24 :
// Page number 156-157
clear ; clc ; close ; // Clear the work space and console

// Given data
A = 0.94*exp(%i*1.5*%pi/180)          // Constant
B = 150.0*exp(%i*67.2*%pi/180)        // Constant(ohm
```

```
)
D = A                                    // Constant
Y_t = 0.00025*exp(%i*-75.0*%pi/180)      // Shunt admittance(mho)
Z_t = 100.0*exp(%i*70.0*%pi/180)         // Series impedance(ohm)

// Calculations
C = (A*D-1)/B                  // Constant(mho)
A_0 = A*(1+Y_t*Z_t)+B*Y_t      // Constant
B_0 = A*Z_t+B                  // Constant(ohm)
C_0 = C*(1+Y_t*Z_t)+D*Y_t      // Constant(mho)
D_0 = C*Z_t+D                  // Constant

// Results
disp("PART II - EXAMPLE : 3.24 : SOLUTION :-")
printf("\nA_0 = %.3f % .f  ", abs(A_0),phasemag(A_0))
printf("\nB_0 = %.f % .1f  ohm", abs(B_0),phasemag(B_0))
printf("\nC_0 = %.6f % .1f  mho", abs(C_0),phasemag(C_0))
printf("\nD_0 = %.3f % .1f  \n", abs(D_0),phasemag(D_0))
printf("\nNOTE: Changes in obtained answer from that of textbook is due to more precision")
```

Scilab code Exa 10.25 Maximum power transmitted Receiving end power factor and Tot

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
```

```
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.25 :
// Page number 163
clear ; clc ; close ; // Clear the work space and console

// Given data
z = complex(0.2,0.6)      // Per phase impedance(ohm)
V_r = 6351.0              // Receiving end voltage per phase(V)
reg = 7.5/100.0           // Voltage regulation

// Calculations
V_s = (1+reg)*V_r          // Sending end voltage per phase(V)
R = real(z)          // Resistance of the line(ohm)
X = imag(z)          // Reactance of the line(ohm)
Z = (R**2+X**2)**0.5          // Impedance per phase(ohm)
P_m = (V_r**2/Z)*((Z*V_s/V_r)-R)          // Maximum power transmitted through line(W/phase)
P_m_MW = P_m/10**6          // Maximum power transmitted through line(MW/phase)
P_m_MWtotal = 3*P_m_MW          // Total maximum power(MW)
Q = -(V_r**2*X)/Z**2          // Reactive power per phase(Var)
Q_MW = Q/10**6          // Reactive power per phase(MVAR)
phi_r = atand(abs(Q_MW/P_m_MW))          //  _r (  )
PF_r = cosd(phi_r)
```

```
// Receiving end lagging PF
I = P_m/(V_r*PF_r)
    // Current delivered(A)
I_KA = I/1000.0
    // Current delivered(KA)
loss = 3*I**2*R
    // Total line loss(W)
loss_MW = loss/10**6
    // Total line loss(MW)

// Results
disp("PART II - EXAMPLE : 3.25 : SOLUTION :-")
printf("\nMaximum power transmitted through the line
    , P_m = %.1f MW", P_m_MWtotal)
printf("\nReceiving end power factor = %.2f (lagging
    )", PF_r)
printf("\nTotal line loss = %.2f MW", loss_MW)
```

Scilab code Exa 10.26 Maximum power that can be transferred to the load

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 3: STEADY STATE CHARACTERISTICS AND
    PERFORMANCE OF TRANSMISSION LINES

// EXAMPLE : 3.26 :
// Page number 163-164
clear ; clc ; close ; // Clear the work space and
    console

// Given data
```

```
L = 100.0                  // Length of line(km)
PF_r = 1.0                 // Receiving end Power factor
Z_c = 400.0                // Characteristic impedance(ohm)
beta = 1.2*10**-3          // Propagation constant(rad/km)
V_s = 230.0                // Sending end voltage(kV)

// Calculations
beta_L = beta*L                    // (rad)
beta_L_d = beta_L*180/%pi          // (  )
A = cosd(beta_L)                   // Constant
B = %i*Z_c*sin(beta_L)             // Constant
alpha_angle = phasemag(A)          //   (  )
beta_angle = phasemag(B)           //   (  )
V_r = V_s                          // Receiving end voltage due to lossless line(kV)
P_max = (V_s*V_r/abs(B))-(abs(A)*V_r**2/abs(B))*cosd(beta_angle-alpha_angle)    // Maximum power transferred(MW)

// Results
disp("PART II - EXAMPLE : 3.26 : SOLUTION :-")
printf("\nMaximum power that can be transferred to the load at receiving end, P_max = %.f MW \n", P_max)
printf("\nNOTE: Changes in obtained answer from that of textbook is due to more precision")
```

Chapter 11

OVERHEAD LINE INSULATORS

Scilab code Exa 11.1 Ratio of capacitance Line voltage and String efficiency

```
// A Textbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 4: OVERHEAD LINE INSULATORS

// EXAMPLE : 4.1 :
// Page number 183
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V_1 = 9.0        // Potential across top unit(kV)
V_2 = 11.0       // Potential across middle unit(kV)
n = 3.0          // Number of disc insulators

// Calculations
```

```
// Case(a)
K = (V_2-V_1)/V_1            // Ratio of capacitance b/w pin & earth to self capacitance
// Case(b)
V_3 = V_2+(V_1+V_2)*K        // Potential across bottom unit(kV)
V = V_1+V_2+V_3              // Voltage between line and earth(kV)
V_l = 3**0.5*V               // Line voltage(kV)
// Case(c)
eff = V/(n*V_3)*100          // String efficiency(%)

// Results
disp("PART II - EXAMPLE : 4.1 : SOLUTION :-")
printf("\nCase(a): Ratio of capacitance b/w pin & earth to self-capacitance of each unit, K = %.2f", K)
printf("\nCase(b): Line voltage = %.2f kV", V_l)
printf("\nCase(c): String efficiency = %.f percent", eff)
```

Scilab code Exa 11.2 Mutual capacitance of each unit in terms of C

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 4: OVERHEAD LINE INSULATORS

// EXAMPLE : 4.2 :
// Page number 183-184
clear ; clc ; close ; // Clear the work space and console
```

```

// Given data
m = 10.0                    // Mutual capacitance of top insulator in terms of C

// Calculations
X = 1+m                     // Mutual capacitance in terms of C
Y = (1.0+2)+m               // Mutual capacitance in terms of C
Z = (1.0+2+3)+m             // Mutual capacitance in terms of C
U = (1.0+2+3+4)+m           // Mutual capacitance in terms of C
V = (1.0+2+3+4+5)+m         // Mutual capacitance in terms of C

// Results
disp("PART II - EXAMPLE : 4.2 : SOLUTION :-")
printf("\nMutual capacitance of each unit:")
printf("\n X = %.f*C", X)
printf("\n Y = %.f*C", Y)
printf("\n Z = %.f*C", Z)
printf("\n U = %.f*C", U)
printf("\n V = %.f*C", V)
```

Scilab code Exa 11.3 Voltage distribution over a string of three suspension insula

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 4: OVERHEAD LINE INSULATORS
```

```

// EXAMPLE : 4.3 :
// Page number 184
clear ; clc ; close ; // Clear the work space and console

// Given data
n = 3.0                   // Number of insulators

// Calculations
V_1 = 155.0/475.0              // Potential across top unit
V_2 = 154.0/155.0*V_1          // Potential across middle unit
V_3 = 166.0/155.0*V_1          // Potential across bottom unit
eff = 100/(n*V_3)              // String efficiency(%)

// Results
disp("PART II - EXAMPLE : 4.3 : SOLUTION :-")
printf("\nVoltage across top unit, V_1 = %.3f*V", V_1)
printf("\nVoltage across middle unit, V_2 = %.3f*V", V_2)
printf("\nVoltage across bottom unit, V_3 = %.2f*V", V_3)
printf("\nString efficiency = %.2f percent", eff)
```

Scilab code Exa 11.4 `Line to neutral voltage and String efficiency`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 4: OVERHEAD LINE INSULATORS

// EXAMPLE : 4.4 :
// Page number 184-185
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V_3 = 17.5              // Voltage across line unit(kV)
c = 1.0/8               // Shunt capacitance = 1/8 of
    insulator capacitance
n = 3.0                 // Number of insulators

// Calculations
K = c                           // String constant
V_1 = V_3/(1+3*K+K**2)          // Voltage across top
    unit(kV)
V_2 = (1+K)*V_1                 // Voltage across middle
     unit(kV)
V = V_1+V_2+V_3                 // Voltage between line
    & earth(kV)
eff = V*100/(n*V_3)             // String efficiency(%)

// Results
disp("PART II - EXAMPLE : 4.4 : SOLUTION :-")
printf("\nLine to neutral voltage , V = %.2f kV", V)
printf("\nString efficiency = %.2f percent", eff)
```

Scilab code Exa 11.5 `Value of line to pin capacitance`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION
```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 4: OVERHEAD LINE INSULATORS

// EXAMPLE : 4.5 :
// Page number 185
clear ; clc ; close ; // Clear the work space and
    console

// Given data
n = 8.0                    // Number of insulators

// Calculations
A = 1.0/(n-1)              // Line to pin capacitance
B = 2.0/(n-2)              // Line to pin capacitance
C = 3.0/(n-3)              // Line to pin capacitance
D = 4.0/(n-4)              // Line to pin capacitance
E = 5.0/(n-5)              // Line to pin capacitance
F = 6.0/(n-6)              // Line to pin capacitance
G = 7.0/(n-7)              // Line to pin capacitance

// Results
disp("PART II - EXAMPLE : 4.5 : SOLUTION :-")
printf("\nLine-to-pin capacitance are:")
printf("\n A = %.3f*C", A)
printf("\n B = %.3f*C", B)
printf("\n C = %.3f*C", C)
printf("\n D = %.3f*C", D)
printf("\n E = %.3f*C", E)
printf("\n F = %.3f*C", F)
printf("\n G = %.3f*C", G)
```

Scilab code Exa 11.6 Voltage distribution as a percentage of voltage of conductor

```
// A Texbook on POWER SYSTEM ENGINEERING
```

```
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 4: OVERHEAD LINE INSULATORS

// EXAMPLE : 4.6 :
// Page number 186
clear ; clc ; close ; // Clear the work space and console

// Given data
m = 6.0                 // Mutual capacitance
n = 5.0                 // Number of insulators

// Calculations
E_4 = (1+(1/m))         // Voltage across 4th insulator as percent of E_5(%)
E_3 = (1+(3/m)+(1/m**2))        // Voltage across 3rd insulator as percent of E_5(%)
E_2 = (1+(6/m)+(5/m**2)+(1/m**3))       // Voltage across 2nd insulator as percent of E_5(%)
E_1 = (1+(10/m)+(15/m**2)+(7/m**3)+(1/m**4))    // Voltage across 1st insulator as percent of E_5(%)
E_5 = 100/(E_4+E_3+E_2+E_1+1)   // Voltage across 5th insulator as percent of E_5(%)
E4 = E_4*E_5            // Voltage across 4th insulator as percent of E_5(%)
E3 = E_3*E_5            // Voltage across 3rd insulator as percent of E_5(%)
```

```
E2 = E_2*E_5
    // Voltage across 2nd insulator as percent of E_5
    (%)
E1 = E_1*E_5
    // Voltage across 1st insulator as percent of E_5
    (%)
eff = 100/(n*E1/100)
    // String efficiency(%)

// Results
disp("PART II - EXAMPLE : 4.6 : SOLUTION :-")
printf("\nVoltage distribution as a percentage of
    voltage of conductor to earth are:")
printf("\n E_1 = %.2f percent", E1)
printf("\n E_2 = %.2f percent", E2)
printf("\n E_3 = %.1f percent", E3)
printf("\n E_4 = %.1f percent", E4)
printf("\n E_5 = %.2f percent", E_5)
printf("\nString efficiency = %.f percent \n", eff)
printf("\nNOTE: Changes in obtained answer from that
     of textbook is due to more precision")
```

Scilab code Exa 11.7 Voltage across each insulator as a percentage of line voltage

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 4: OVERHEAD LINE INSULATORS

// EXAMPLE : 4.7 :
// Page number 186-187
clear ; clc ; close ; // Clear the work space and
```

```
console

// Given data
n = 3.0                    // Number of insulators
C_1 = 0.2                  // Capacitance in terms of C
C_2 = 0.1                  // Capacitance in terms of C

// Calculations
// Without guard ring
e_2_a = 13.0/13.3                  // Potential across middle unit as top unit
e_1_a = 8.3/6.5*e_2_a              // Potential across bottom unit
E_a = 1+(1/(8.3/6.5))+(1/e_1_a)    // Voltage in terms of e_1
eff_a = E_a/n*100                  // String efficiency(%)
e1_a = 1/E_a                       // Voltage across bottom unit as a % of line voltage
e2_a = 1/(8.3/6.5)*e1_a            // Voltage across middle unit as a % of line voltage
e3_a = 1/e_1_a*e1_a                // Voltage across top unit as a % of line voltage
// With guard ring
e_2_b = 15.4/15.5                  // Potential across middle unit as top unit
e_1_b = 8.3/7.7*e_2_b              // Potential across bottom unit
E_b = 1+(1/(8.3/7.7))+(1/e_1_b)    // Voltage in terms of e_1
eff_b = E_b/n*100                  // String efficiency(%)
e1_b = 1/E_b                       // Voltage across bottom unit as a % of line voltage
e2_b = 1/(8.3/7.7)*e1_b            // Voltage across middle unit as a % of line voltage
e3_b = 1/e_1_b*e1_b                // Voltage across top unit as a % of line voltage
```

```
// Results
disp("PART II - EXAMPLE : 4.7 : SOLUTION :-")
printf("\nWithout guard ring:")
printf("\n Voltage across bottom unit, e_1 = %.2f*E"
    , e1_a)
printf("\n Voltage across bottom unit, e_2 = %.2f*E"
    , e2_a)
printf("\n Voltage across bottom unit, e_3 = %.2f*E"
    , e3_a)
printf("\n String efficiency = %.1f percent \n",
    eff_a)
printf("\nWith guard ring:")
printf("\n Voltage across bottom unit, e_1 = %.2f*E"
    , e1_b)
printf("\n Voltage across bottom unit, e_2 = %.2f*E"
    , e2_b)
printf("\n Voltage across bottom unit, e_3 = %.3f*E"
    , e3_b)
printf("\n String efficiency = %.2f percent", eff_b)
```

Scilab code Exa 11.8 Voltage across each insulator as a percentage of line voltage

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 4: OVERHEAD LINE INSULATORS

// EXAMPLE : 4.8 :
// Page number 187-188
clear ; clc ; close ; // Clear the work space and
    console
```

```

// Given data
n = 3.0                  // Number of insulators

// Calculations
V_1 = 0.988                        // Voltage across top unit as middle unit
V_3 = 1.362                        // Voltage across bottom unit as middle unit
V_2 = 1/(V_1+1+V_3)                // Voltage across middle unit as % of line voltage to earth
V1 = V_1*V_2*100                   // Voltage across top unit as % of line voltage to earth
V2 = V_2*100                       // Voltage across middle unit as % of line voltage to earth
V3 = V_3*V_2*100                   // Voltage across bottom unit as % of line voltage to earth
eff = 100/(n*V3/100)               // String efficiency(%)

// Results
disp("PART II - EXAMPLE : 4.8 : SOLUTION :-")
printf("\nCase(a): Voltage across top unit as a percentage of line voltage to earth, V_1 = %.2f percent", V1)
printf("\n         Voltage across middle unit as a percentage of line voltage to earth, V_2 = %.2f percent", V2)
printf("\n         Voltage across bottom unit as a percentage of line voltage to earth, V_3 = %.2f percent", V3)
printf("\nCase(b): String efficiency = %.2f percent", eff)
```

Scilab code Exa 11.9 Voltage on the line end unit and Value of capacitance require

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 4: OVERHEAD LINE INSULATORS

// EXAMPLE : 4.9 :
// Page number 188
clear ; clc ; close ; // Clear the work space and
    console

// Given data
n = 3.0                   // Number of insulators
V = 20.0                  // Voltage across each
    conductor(kV)
c = 1.0/5                 // Capacitance ratio

// Calculations
V_2 = 6.0/5.0                 // Voltage across middle
    unit as top unit
V_1 = V/(1+2*V_2)             // Voltage across top unit(
    kV)
V_3 = V_2*V_1                 // Voltage across bottom
    unit(kV)
C_x = c*(1+(1/V_2))           // Capacitance required

// Results
disp("PART II - EXAMPLE : 4.9 : SOLUTION :-")
printf("\nCase(a): Voltage on the line-end unit, V_3
     = %.2f kV", V_3)
printf("\nCase(b): Value of capacitance required, Cx
     = %.3f*C", C_x)
```

Chapter 12

MECHANICAL DESIGN OF OVERHEAD LINES

Scilab code Exa 12.1 `Weight of conductor`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

// EXAMPLE : 5.1 :
// Page number 198
clear ; clc ; close ; // Clear the work space and console

// Given data
u = 5758.0          // Ultimate strength(kg)
S = 2.0             // Sag(m)
s = 2.0             // Factor of safety
L = 250.0           // Span length(m)

```

```
// Calculations
T = u/s                                // Allowable
    max tension(kg)
w = S*8.0*T/L**2                       // weight(kg/
    m)
l = L/2                                // Half span
    length(m)
half_span = l+(w**2*l**3/(6*T**2))     // Half span
    length(m)
total_length = 2*half_span             // Total
    length(m)
weight = w*total_length                // Weight of
    conductor(kg)

// Results
disp("PART II - EXAMPLE : 5.1 : SOLUTION :-")
printf("\nWeight of conductor = %.2f kg", weight)
```

Scilab code Exa 12.2 Point of maximum sag at the lower support

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

// EXAMPLE : 5.2 :
// Page number 198
clear ; clc ; close ; // Clear the work space and
    console

// Given data
L = 250.0              // Span length(m)
```

```
h = 10.0                  // Difference in height(m)
r = 1.0                   // Radius of conductor(cm)
w = 2.5                   // Weight of conductor(kg/m)
wind = 1.2                // Wind load(kg/m)
s = 3.0                   // Factor of safety
tensile = 4300.0          // Maximum tensile strength(kg
    /sq.cm)

// Calculations
W = (w**2+wind**2)**0.5          // Total pressure on
    conductor(kg/m)
f = tensile/s                    // Permissible stress
    in conductor(kg/sq.cm)
a = %pi*r**2                     // Area of the
    conductor(sq.cm)
T = f*a                          // Allowable max
    tension(kg)
x = (L/2)-(T*h/(L*W))            // Point of maximum
    sag at the lower support(m)

// Results
disp("PART II - EXAMPLE : 5.2 : SOLUTION :-")
printf("\nPoint of maximum sag at the lower support,
     x = %.2f metres", x)
```

Scilab code Exa 12.3 `Vertical sag`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

```

```
// EXAMPLE : 5.3 :
// Page number 198-199
clear ; clc ; close ; // Clear the work space and console

// Given data
a = 2.5                    // Cross-sectional area(sq.cm)
L = 250.0                  // Span(m)
w_c = 1.8                  // Weight of conductor(kg/m)
u = 8000.0                 // Ultimate strength(kg/cm^2)
wind = 40.0                // Wind load(kg/cm^2)
s = 3.0                    // Factor of safety

// Calculations
d = (4.0*a/%pi)**0.5              // Diameter(cm)
T = u*a/s                         // Allowable max tension(kg)
w_w = wind*d/100.0                // Horizontal wind force(kg)
w_r = (w_c**2+w_w**2)**0.5        // Resultant force(kg/m)
S = w_r*L**2/(8*T)                // Slant sag(m)
vertical_sag = S*(w_c/w_r)        // Vertical sag(m)

// Results
disp("PART II - EXAMPLE : 5.3 : SOLUTION :-")
printf("\nVertical sag = %.3f metres", vertical_sag)
```

Scilab code Exa 12.4 Height above ground at which the conductors should be support

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

// EXAMPLE : 5.4 :
// Page number 199
clear ; clc ; close ; // Clear the work space and console

// Given data
a = 110.0                  // Cross-sectional area(sq.mm)
w_c = 844.0/1000           // Weight of conductor(kg/m)
U = 7950.0                 // Ultimate strength(kg)
L = 300.0                  // Span(m)
s = 2.0                    // Factor of safety
wind = 75.0                // Wind pressure(kg/m^2)
h = 7.0                    // Ground clearance(m)
d = 2.79                   // Diameter of copper(mm)
n = 7.0                    // Number of strands

// Calculations
dia = n*d                           // Diameter of conductor(mm)
w_w = wind*dia/1000.0               // Horizontal wind force(kg)
w = (w_c**2+w_w**2)**0.5            // Resultant force(kg)
T = U/2.0                           // Allowable tension(m)
l = L/2.0                           // Half-span(m)
D = w*l**2/(2*T)                    // Distance(m)
height = h+D                        // Height above ground at which the conductors should be supported(m)

// Results
disp("PART II - EXAMPLE : 5.4 : SOLUTION :-")
printf("\nHeight above ground at which the
```

```
conductors should be supported = %.2f metres",
height)
```

Scilab code Exa 12.5 Permissible span between two supports

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

// EXAMPLE : 5.5 :
// Page number 199
clear ; clc ; close ; // Clear the work space and
    console

// Given data
w_w = 1.781            // Wind pressure on conductor(
    kg/m)
w_i = 1.08             // Weight of ice on conductor(
    kg/m)
D = 6.0                // Maximum permissible sag(m)
s = 2.0                // Factor of safety
w_c = 0.844            // Weight of conductor(kg/m)
u = 7950.0             // Ultimate strength(kg)

// Calculations
w = ((w_c+w_i)**2+w_w**2)**0.5          // Total force
    on conductor(kg/m)
T = u/s                                 // Allowable
    maximum tension(kg)
l = ((D*2*T)/w)**0.5                    // Half span(m)
L = 2.0*l                               // Permissible
```

```
    span between two supports(m)

// Results
disp("PART II - EXAMPLE : 5.5 : SOLUTION :-")
printf("\nPermissible span between two supports = %.
    f metres \n", L)
printf("\nNOTE: ERROR: Horizontal wind load, w_w =
    1.781 kg/m, not 1.78 kg/m as mentioned in problem
     statement")
```

Scilab code Exa 12.6 Maximum sag of line due to weight of conductor Additional wei

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

// EXAMPLE : 5.6 :
// Page number 199-200
clear ; clc ; close ; // Clear the work space and
    console

// Given data
a = 0.484               // Area of conductor(sq.cm)
d = 0.889               // Overall diameter(cm)
w_c = 428/1000.0        // Weight(kg/m)
u = 1973.0              // Breaking strength(kg)
s = 2.0                 // Factor of safety
L = 200.0               // Span(m)
t = 1.0                 // Ice thickness(cm)
wind = 39.0             // Wind pressure(kg/m^2)

```

```
// Calculations
// Case(i)
l = L/2.0                                    // Half span(m)
T = u/s                                      // Allowable maximum tension(kg)
D_1 = w_c*l**2/(2*T)                         // Maximum sag due to weight of conductor(m)
// Case(ii)
w_i = 913.5*%pi*t*(d+t)*10**-4               // Weight of ice on conductor(kg/m)
w = w_c+w_i                                  // Total weight of conductor & ice(kg/m)
D_2 = w*l**2/(2*T)                           // Maximum sag due to additional weight of ice(m)
// Case(iii)
D = d+2.0*t                                  // Diameter due to ice(cm)
w_w = wind*D*10**-2                          // Wind pressure on conductor(kg/m)
w_3 = ((w_c+w_i)**2+w_w**2)**0.5             // Total force on conductor(kg/m)
D_3 = w_3*l**2/(2*T)                         // Maximum sag due to (i), (ii) & wind(m)
theta = atand(w_w/(w_c+w_i))                 // ( )
vertical_sag = D_3*cosd(theta)               // Vertical sag(m)

// Results
disp("PART II - EXAMPLE : 5.6 : SOLUTION :-")
printf("\nCase(i)  : Maximum sag of line due to weight of conductor, D = %.2f metres", D_1)
printf("\nCase(ii) : Maximum sag of line due to additional weight of ice, D = %.2f metres", D_2)
printf("\nCase(iii): Maximum sag of line due to (i),(ii) plus wind, D = %.2f metres", D_3)
printf("\n           Vertical sag = %.2f metres",
```

```
vertical_sag)
```

Scilab code Exa 12.7 Point of minimum sag

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

// EXAMPLE : 5.7 :
// Page number 200
clear ; clc ; close ; // Clear the work space and console

// Given data
W = 428/1000.0        // Weight(kg/m)
u = 1973.0            // Breaking strength(kg)
s = 2.0               // Factor of safety
l = 200.0             // Span(m)
h = 3.0               // Difference in tower height(m)

// Calculations
T = u/s                                  // Allowable maximum tension(kg)
x_2 = (l/2.0)+(T*h/(W*l))                // Point of minimum sag from tower at higher level(m)
x_1 = l-x_2                              // Point of minimum sag from tower at lower level(m)

// Results
disp("PART II - EXAMPLE : 5.7 : SOLUTION :-")
printf("\nPoint of minimum sag, x_1 = %.1f metres",
```

```
x_1)
printf("\nPoint of minimum sag, x_2 = %.1f metres",x_2)
```

Scilab code Exa 12.8 Clearance between conductor and water at a point midway betwe

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

// EXAMPLE : 5.8 :
// Page number 200-201
clear ; clc ; close ; // Clear the work space and console

// Given data
h_1 = 50.0          // Height of tower P1(m)
h_2 = 80.0          // Height of tower P2(m)
L = 300.0           // Horizontal distance b/w towers(m)
T = 2000.0          // Tension in conductor(kg)
w = 0.844           // Weight of conductor(kg/m)

// Calculations
h = h_2-h_1                           // Difference in height of tower(m)
x_2 = (L/2.0)+(T*h/(w*L))             // Point of minimum sag from tower P2(m)
x_1 = (L/2.0)-(T*h/(w*L))             // Point of minimum sag from tower at lower level(m)
P = (L/2.0)-x_1                       // Distance of
```

```
point P(m)
D = w*P**2/(2*T)                          // Height of P above O(m)
D_2 = w*x_2**2/(2*T)                      // Height of P2 above O(m)
mid_point_P2 = D_2-D                      // Mid-point below P2(m)
clearance = h_2-mid_point_P2              // Clearance b/w conductor & water(m)
D_1 = w*x_1**2/(2*T)                      // Height of P1 above O(m)
mid_point_P1 = D-D_1                      // Mid-point above P1(m)
clearance_alt = h_1+mid_point_P1          // Clearance b/w conductor & water(m)

// Results
disp("PART II - EXAMPLE : 5.8 : SOLUTION :-")
printf("\nClearance between conductor & water at a point midway b/w towers = %.2f m above water\n", clearance)
printf("\nALTERNATIVE METHOD:")
printf("\nClearance between conductor & water at a point midway b/w towers = %.2f m above water", clearance_alt)
```

Scilab code Exa 12.9 Sag at erection and Tension of the line

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES
```

```

// EXAMPLE : 5.9 :
// Page number 201
clear ; clc ; close ; // Clear the work space and console

// Given data
L = 300.0                // Span(m)
T_still = 45.0           // Temperature in still air( C )
a = 226.0                // Area(mm^2)
d = 19.53/10             // Overall diameter(cm)
w_2 = 0.844              // Weight of conductor(kg/m)
u = 7950.0               // Ultimate strength(kg)
alpha = 18.44*10**-6 // Co-efficient of linear expression(/ C )
E = 9.32*10**3           // Modulus of elasticity(kg/mm^2)
t = 0.95                 // Ice thickness(cm)
wind = 39.0              // Wind pressure(kg/m^2)
T_worst = -5.0           // Temperature in worst condition( C )

// Calculations
w_i = 915.0*%pi*t*(d+t)*10**-4          // Weight of ice on conductor(kg/m)
w_w = wind*(d+2*t)*10**-2               // Wind load of conductor(kg/m)
w_1  = ((w_2+w_i)**2+w_w**2)**0.5       // Total force on conductor(kg/m)
t = T_still-T_worst                     // Temperature( C )
l = L/2.0                               // Half span(m)
T = u/2.0                               // Allowable tension(kg)
A = 1.0                                 // Co-efficient of x^3
```

```
B = a*E*(alpha*t+((w_1*l/T)**2/6))-T    // Co-
    efficient of x^2
C = 0                                    // Co-
    efficient of x
D = -(w_2**2*l**2*a*E/6)                 // Co-
    efficient of constant
T_2_sol = roots([A,B,C,D])               // Roots of
    tension of a line
T_2_s = T_2_sol(3)                       // Feasible
    solution of tension of
T_2 = 1710.0                             // Tension
    in conductor(kg). Obtianed directly from textbook
sag = w_2*l**2/(2*T_2)                   // Sag at
    erection(m)

// Results
disp("PART II - EXAMPLE : 5.9 : SOLUTION :-")
printf("\nSag at erection = %.2f metres", sag)
printf("\nTension of the line, T_2 = %.f kg (An app.
     solution as per calculation) = %.f kg (More
    correctly as standard value)", T_2_s,T_2)
```

Scilab code Exa 12.10 Sag in inclined direction and Vertical direction

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

// EXAMPLE : 5.10 :
// Page number 201-202
clear ; clc ; close ; // Clear the work space and
```

```
console

// Given data
L = 250.0            // Span(m)
d = 1.42             // Diameter(cm)
w = 1.09             // Dead weight(kg/m)
wind = 37.8          // Wind pressure(kg/m^2)
r = 1.25             // Ice thickness(cm)
f_m = 1050.0         // Maximum working stress(kg/sq.cm)

// Calculations
w_i = 913.5*%pi*r*(d+r)*10**-4            // Weight of ice on conductor(kg/m)
w_w = wind*(d+2*r)*10**-2                 // Wind load of conductor(kg/m)
w_r  = ((w+w_i)**2+w_w**2)**0.5           // Resultant pressure(kg/m)
a = %pi*d**2/4.0                          // Area(cm^2)
T_0 = f_m*a                               // Tension(kg)
S = w_r*L**2/(8*T_0)                      // Total sag(m)
vertical_sag = S*(w+w_i)/w_r              // Vertical component of sag(m)

// Results
disp("PART II - EXAMPLE : 5.10 : SOLUTION :-")
printf("\nCase(i) : Sag in inclined direction = %.f m", S)
printf("\nCase(ii): Sag in vertical direction = %.2f m", vertical_sag)
```

Scilab code Exa 12.11 Sag in still air Wind pressure Ice coating and Vertical sag

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 5: MECHANICAL DESIGN OF OVERHEAD LINES

// EXAMPLE : 5.11 :
// Page number 202-203
clear ; clc ; close ; // Clear the work space and console

// Given data
a = 120.0                    // Area(mm^2)
ds = 2.11                    // Diameter of each strand(mm)
W = 1118.0/1000              // Weight of conductor(kg/m)
L = 200.0                    // Span(m)
stress = 42.2                // Ultimate tensile stress(kg/mm^2)
wind = 60.0                  // Wind pressure(kg/m^2)
t = 10.0                     // Ice thickness(mm)

// Calculations
n = 3.0                                             // Number of layers
d = (2*n+1)*ds                                      // Overall diameter of conductor(mm)
u = stress*a                                        // Ultimate strength(kg)
T = u/4.0                                           // Working stregth(kg)
// Case(a)
S_a = W*L**2/(8*T)                                  // Sag in still air(m)
// Case(b)
```

```
area = d*100*10.0*10**-6                      //Projected area to wind pressure(m^2)
w_w = wind*area                               //Wind load/m(kg)
w_r = (W**2+w_w**2)**0.5                      //Resultant weight/m(kg)
S_b = w_r*L**2/(8*T)                          //Total sag with wind pressure(m)
w_i = 0.915*%pi/4*((d+2*t)**2-(d**2))/1000.0  //Weight of ice on conductor(kg/m)
area_i = (d+2*t)*1000.0*10**-6                //Projected area to wind pressure(m^2)
w_n = wind*area_i                             //Wind load/m(kg)
w_r_c = ((W+w_i)**2+w_n**2)**0.5              //Resultant weight/m(kg)
S_c = w_r_c*L**2/(8*T)                        //Total sag with wind pressure and ice coating(m)
S_v = S_c*(W+w_i)/w_r_c                       //Vertical component of sag(m)

// Results
disp("PART II - EXAMPLE : 5.11 : SOLUTION :-")
printf("\nCase(a) : Sag in still air, S = %.2f m", S_a)
printf("\nCase(b) : Sag with wind pressure, S = %.2f m", S_b)
printf("\n          Sag with wind pressure and ice coating, S = %.2f m", S_c)
printf("\n          Vertical sag, S_v = %.2f m \n", S_v)
printf("\nNOTE: ERROR: calculation mistake in the textbook")
```

Chapter 13

INTERFERENCE OF POWER LINES WITH NEIGHBOURING COMMUNICATION CIRCUITS

Scilab code Exa 13.1 Mutual inductance between the circuits and Voltage induced in

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 6: INTERFERENCE OF POWER LINES WITH
   NEIGHBOURING COMMUNICATION CIRCUITS

// EXAMPLE : 6.1 :
// Page number 206
clear ; clc ; close ; // Clear the work space and
   console
```

```
// Given data
f = 50.0                 // Frequency(Hz)
d = 4.0                  // Spacing b/w conductors(m)
D = 2.0                  // Distance of telephone line below conductor(m)
s = 60.0/100             // Spacing b/w telephone line(m)
r = 2.0                  // Radius of power line(mm)
I = 150.0                // Current in power line(A)

// Calculations
D_ac = (D**2+((d-s)/2)**2)**0.5          // Distance b/w a & c(m)
D_ad = (D**2+(((d-s)/2)+s)**2)**0.5      // Distance b/w a & d(m)
M = 4.0*10**-7*log(D_ad/D_ac)*1000       // Mutual inductance b/w circuits(H/km)
V_CD = 2.0*%pi*f*M*I                     // Voltage induced in the telephone line(V/km)

// Results
disp("PART II - EXAMPLE : 6.1 : SOLUTION :-")
printf("\nMutual inductance between the circuits, M = %.e H/km", M)
printf("\nVoltage induced in the telephone line, V_CD = %.2f V/km", V_CD)
```

Scilab code Exa 13.2 Induced voltage at fundamental frequency and Potential of tel

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 6: INTERFERENCE OF POWER LINES WITH NEIGHBOURING COMMUNICATION CIRCUITS

// EXAMPLE : 6.2 :
// Page number 206-207
clear ; clc ; close ; // Clear the work space and console

// Given data
f = 50.0                  // Frequency(Hz)
l = 160.0                 // Length of line(km)
V = 132.0*10**3           // Line voltage(V)
P = 25.0*10**6            // Load delivered(W)
PF = 0.8                  // Lagging power factor
r = 5.0/1000              // Radius of power line conductor(m)
d = 4.0                   // Spacing b/w conductors(m)
OS = 6.0                  // Distance(m)
OT = 6.5                  // Distance(m)
CT = 18.0                 // Distance(m)

// Calculations
AO = 3**0.5*d/2.0                                                      // Distance A to O(m). From figure E6.2
AS = OS+AO                                                             // Distance A to S(m)
AT = AO+OT                                                             // Distance A to T(m)
OB = d/2.0                                                             // Distance O to B(m)
BS = (OB**2+OS**2)**0.5                                                // Distance B to S(m)
```

```
BT = (OB**2+OT**2)**0.5        // Distance B to T(m)
M_A = 0.2*log(AT/AS)           // Mutual inductance at A(mH/km)
M_B = 0.2*log(BT/BS)           // Mutual inductance at B(mH/km)
M = M_B-M_A                    // Mutual inductance at C(mH/km)
I = P/(3**0.5*V*PF)            // Current(A)
E_m = 2.0*%pi*f*M*I*10**-3*l   // Induced voltage(V)
V_A = V/3**0.5                 // Phase voltage(V)
h = AO+CT                      // Height(m)
V_SA = V_A*log10(((2*h)-AS)/AS)/log10(((2*h)-r)/r)   // Potential(V)
H = CT                         // Height(m)
V_B = V_A                      // Phase voltage(V)
V_SB = V_B*log10(((2*H)-BS)/BS)/log10(((2*H)-r)/r)   // Potential(V)
V_S = V_SB-V_SA                // Total potential of S w.r.t earth(V)

```

```
// Results
disp("PART II - EXAMPLE : 6.2 : SOLUTION :-")
printf("\nInduced voltage at fundamental frequency ,
   E_m = %.1f V", E_m)
printf("\nPotential of telephone conductor S above
   earth , V_S = %.f V \n", V_S)
printf("\nNOTE: ERROR: Changes in obtained answer is
    due to precision and calculation mistakes in
   textbook")
```

Chapter 14

UNDERGROUND CABLES

Scilab code Exa 14.1 Insulation resistance per km

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.1 :
// Page number 211
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 2.5                  // Core diameter(cm)
t = 1.25                 // Insulation thickness(cm)
rho = 4.5*10**14         // Resistivity of insulation(ohm-cm)
l = 10.0**5              // Length(cm)

// Calculations
```

```
D = d+2*t                           // Overall diameter
    (cm)
R_i = rho/(2*%pi*l)*log(D/d)        // Insulation
    resistance(ohm)

// Results
disp("PART II - EXAMPLE : 7.1 : SOLUTION :-")
printf("\nInsulation resistance per km, R_i = %.2e
    ohm\n", R_i)
printf("\nNOTE: ERROR: Mistake in final answer in
    textbook")
```

Scilab code Exa 14.2 Insulation thickness

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.2 :
// Page number 211
clear ; clc ; close ; // Clear the work space and
    console

// Given data
R = 495.0*10**6          // Insulation resistance(ohm/km
    )
d = 3.0                  // Core diameter(cm)
rho = 4.5*10**14         // Resistivity of insulation(
    ohm-cm)

// Calculations
```

```
l = 1000.0                                  // Length
    of cable(m)
r_2 = d/2.0                                 // Core
    radius(cm)
Rho = rho/100.0                             //
    Resistivity of insulation(ohm-m)
r1_r2 = exp((2*%pi*l*R)/Rho)                // r1/r2
r_1 = 2*r_2                                 // Cable
    radius(cm)
thick = r_1-r_2                             //
    Insulation thickness(cm)

// Results
disp("PART II - EXAMPLE : 7.2 : SOLUTION :-")
printf("\nInsulation thickness = %.1f cm", thick)
```

Scilab code Exa 14.3 Capacitance and Charging current of single core cable

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.3 :
// Page number 212
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 66.0*10**3          // Line Voltage(V)
l = 1.0                 // Length of cable(km)
d = 15.0                // Core diameter(cm)
```

```
D = 60.0                  // Sheath diameter(cm)
e_r = 3.6                 // Relative permittivity
f = 50.0                  // Frequency(Hz)

// Calculations
C = e_r/(18.0*log(D/d))*l          // Capacitance(F )
I_ch = V/3**0.5*2*%pi*f*C*10**-6    // Charging current(A)

// Results
disp("PART II - EXAMPLE : 7.3 : SOLUTION :-")
printf("\nCapacitance of single-core cable, C = %.3f F ", C)
printf("\nCharging current of single-core cable = %.2f A", I_ch)
```

Scilab code Exa 14.4 Most economical diameter of a single core cable and Overall d

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.4 :
// Page number 212
clear ; clc ; close ; // Clear the work space and console

// Given data
V_l = 132.0          // Line Voltage(kV)
g_max = 60.0         // Maximum Line Voltage(kV)
```

```

// Calculations
V = V_l/3**0.5*2**0.5      // Phase Voltage(kV)
d = 2*V/g_max              // Core diameter(cm)
D = 2.718*d                // Overall diameter(cm)

// Results
disp("PART II - EXAMPLE : 7.4 : SOLUTION :-")
printf("\nMost economical diameter of a single-core
   cable, d = %.1f cm", d)
printf("\nOverall diameter of the insulation, D = %
   .3f cm\n", D)
printf("\nNOTE: Slight change in obtained answer due
    to precision")
```

Scilab code Exa 14.6 Conductor radius and Electric field strength that must be wit

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.6 :
// Page number 212-213
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 11.0*10**3           // Line Voltage(V)
dia_out = 8.0            // Outside diameter(cm)

// Calculations
```

```
D = dia_out/2.0                                  // Overall diameter(cm)
d = (D)/2.718                                    // Conductor diameter(cm)
r = d/2                                          // Conductor radius(cm)
g_m = 2*V/(d*log(D/d)*10)                        // Maximum value of electric field strength(kV/m)

// Results
disp("PART II - EXAMPLE : 7.6 : SOLUTION :-")
printf("\nConductor radius , r = %.3f cm", r)
printf("\nElectric field strength that must be withstood , g_m = %.f kV/m", g_m)
```

Scilab code Exa 14.7 Location of intersheath and Ratio of maximum electric field s

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.7 :
// Page number 214
clear ; clc ; close ; // Clear the work space and console

// Given data
R_3 = 1.00              // Cable radius(cm)
R_1 = 2.5               // Cable radius(cm)

// Calculations
```

```
R_2 = (R_1*R_3)**0.5        // Location of intersheath
    (cm)
alpha = R_1/R_2             //
ratio = 2.0/(1+alpha)       // Ratio of maximum
    electric field strength with & without
    intersheath

// Results
disp("PART II - EXAMPLE : 7.7 : SOLUTION :-")
printf("\nLocation of intersheath , R_2 = %.2f cm",
    R_2)
printf("\nRatio of maximum electric field strength
    with & without intersheath = %.3f ", ratio)
```

Scilab code Exa 14.8 Maximum and Minimum stress in the insulation

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.8 :
// Page number 215
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 33.0                // Line Voltage(kV)
D_2 = 2.0               // Conductor diameter(cm)
D_1 = 3.0               // Sheath diameter(cm)

// Calculations
```

```
R_2 = D_2/2                        // Conductor radius(cm)
R_1 = D_1/2                        // Sheath radius (cm)
g_max = V/(R_2*log(R_1/R_2))       // RMS value of maximum stress in the insulation(kV/cm)
g_min = V/(R_1*log(R_1/R_2))       // RMS value of minimum stress in the insulation(kV/cm)

// Results
disp("PART II - EXAMPLE : 7.8 : SOLUTION :-")
printf("\nMaximum stress in the insulation , g_max = %.2f kV/cm (rms)", g_max)
printf("\nMinimum stress in the insulation , g_min = %.2f kV/cm (rms)", g_min)
```

Scilab code Exa 14.9 Maximum stress with and without intersheath Best position and

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.9 :
// Page number 215
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 2.5              // Conductor diameter(cm)
D = 6.0              // Sheath diameter(cm)
V_l = 66.0           // Line Voltage(kV)
```

```

// Calculations
alpha = (D/d)**(1.0/3)                    //
d_1 = d*alpha                             // Best position of first intersheath(cm)
d_2 = d_1*alpha                           // Best position of second intersheath(cm)
V = V_l/3**0.5*2**0.5                     // Peak voltage on core(kV)
V_2 = V/(1+(1/alpha)+(1/alpha**2))        // Peak voltage on second intersheath(kV)
V_1 = (1+(1/alpha))*V_2                   // Voltage on first intersheath(kV)
stress_max = 2*V/(d*log(D/d))             // Maximum stress without intersheath(kV/cm)
stress_min = stress_max*d/D               // Minimum stress without intersheath(kV/cm)
g_max = V*3/(1+alpha+alpha**2)            // Maximum stress with intersheath(kV/cm)

// Results
disp("PART II - EXAMPLE : 7.9 : SOLUTION :-")
printf("\nMaximum stress without intersheath = %.2f kV/cm", stress_max)
printf("\nBest position of first intersheath, d_1 = %.2f cm", d_1)
printf("\nBest position of second intersheath, d_2 = %.3f cm", d_2)
printf("\nMaximum stress with intersheath = %.2f kV/cm", g_max)
printf("\nVoltage on the first intersheath, V_1 = %.2f kV", V_1)
printf("\nVoltage on the second intersheath, V_2 = %.2f kV \n", V_2)
printf("\nNOTE: Changes in the obtained answer is due to more precision here")
```

Scilab code Exa 14.10 Maximum stress in the two dielectrics

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.10 :
// Page number 215-216
clear ; clc ; close ; // Clear the work space and
    console

// Given data
e_1 = 3.6          // Inner relative permittivity
e_2 = 2.5          // Outer relative permittivity
d = 1.0            // Conductor diameter(cm)
d_1 = 3.0          // Sheath diameter(cm)
D = 5.0            // Overall diameter(cm)
V_l = 66.0         // Line Voltage(kV)

// Calculations
V = V_l/3**0.5*2**0.5
                                        // Peak voltage on
     core(kV)
g1_max = 2*V/(d*(log(d_1/d)+e_1/e_2*log(D/d_1)))
       // Maximum stress in first dielectric(kV/km)
g_max = 2*V/(d_1*(e_2/e_1*log(d_1/d)+log(D/d_1)))
      // Maximum stress in second dielectric(kV/km)

// Results
disp("PART II - EXAMPLE : 7.10 : SOLUTION :-")
```

```
printf("\nMaximum stress in first dielectric ,
   g_1_max = %.2f kV/cm", g1_max)
printf("\nMaximum stress in second dielectric , g_max
    = %.2f kV/cm", g_max)
```

Scilab code Exa 14.11 Diameter and Voltage of intersheath Conductor and Outside di

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.11 :
// Page number 216-217
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 85.0          // Line Voltage(kV)
g_max = 55.0      // Maximum stress(kV/cm)

// Calculations
V_1 = 0.632*V         // Intersheath potential(kV)
d = 0.736*V/g_max     // Core diameter(cm)
d_1 = 2*V/g_max       // Intersheath diameter(cm)
D = 3.76*V/g_max      // Overall diameter(cm)
d_un = 2*V/g_max      // Core diameter of ungraded
   cable(cm)
D_un = 2.718*d_1      // Overall diameter of
   ungraded cable(cm)

// Results
```

```
disp("PART II - EXAMPLE : 7.11 : SOLUTION :-")
printf("\nDiameter of intersheath , d_1 = %.2f cm",
    d_1)
printf("\nVoltage of intersheath , V_1 = %.2f kV, to
    neutral", V_1)
printf("\nConductor diameter of graded cable , d = %
    .2f cm", d)
printf("\nOutside diameter of graded cable , D = %.2f
     cm", D)
printf("\nConductor diameter of ungraded cable , d =
    %.2f cm", d_un)
printf("\nOutside diameter of ungraded cable , D = %
    .2f cm", D_un)
```

Scilab code Exa 14.12 Equivalent star connected capacity and kVA required

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.12 :
// Page number 219
clear ; clc ; close ; // Clear the work space and
    console

// Given data
c = 0.3          // Capacitance b/w any 2 conductor &
     sheath earthed( F /km)
l = 10.0         // Length(km)
V = 33.0         // Line Voltage(kV)
f = 50.0         // Frequency(Hz)
```

```
// Calculations
C_eq = 1*c                                    // Capacitance
    b/w any 2 conductor & sheath earthed( F )
C_p = 2.0*C_eq                                // Capacitance
    per phase( F )
kVA = V**2*2*%pi*f*C_p/1000.0                 // Three-phase
    kVA required(kVA)

// Results
disp("PART II - EXAMPLE : 7.12 : SOLUTION :-")
printf("\nEquivalent star connected capacity , C_eq =
    %.f  F ", C_eq)
printf("\nkVA required = %.1f kVA", kVA)
```

Scilab code Exa 14.13 Charging current drawn by a cable with three cores

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.13 :
// Page number 219
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 11.0*10**3      // Line Voltage(V)
f = 50.0            // Frequency(Hz)
C_c = 3.7           // Measured capacitance( F )

```

```
// Calculations
C_0 = 2*C_c                                   // Capacitance( F )
I_ch = 2*%pi*f*C_0*V/3**0.5*10**-6            // Charging current per phase(A)

// Results
disp("PART II - EXAMPLE : 7.13 : SOLUTION :-")
printf("\nCharging current drawn by a cable = %.2f A", I_ch)
```

Scilab code Exa 14.14 Capacitance between any two conductors Two bounded conductor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.14 :
// Page number 219-220
clear ; clc ; close ; // Clear the work space and console

// Given data
c_s = 0.90          // Capacitance b/w all conductors( F )
C_0 = 0.4           // Capacitance b/w two conductor( F )
V = 11.0*10**3      // Line Voltage(V)
f = 50.0            // Frequency(Hz)

// Calculations
```

```
C_s = c_s/3.0                                    //
    Capacitance measured( F )
C_c = (C_0-C_s)/2.0                              //
    Capacitance( F )
C_a = 3.0/2*(C_c+(1/3.0)*C_s)                    //
    Capacitance b/w any two conductors( F )
C_b = 2.0*C_c+(2.0/3)*C_s                        //
    Capacitance b/w any two bounded conductors and
    the third conductor( F )
C_o = 3.0*C_c+C_s                                //
    Capacitance to neutral( F )
I_c = 2.0*%pi*f*C_o*V/3**0.5*10**-6              //
    Charging current(A)

// Results
disp("PART II - EXAMPLE : 7.14 : SOLUTION :-")
printf("\nCase(a): Capacitance between any two
    conductors = %.3f  F ", C_a)
printf("\nCase(b): Capacitance between any two
    bounded conductors and the third conductor = %.1f
      F ", C_b)
printf("\nCase(c): Capacitance to neutral, C_0 = %.2
    f  F ", C_o)
printf("\n          Charging current taken by cable,
    I_c = %.3f A \n", I_c)
printf("\nNOTE: ERROR: Calculation mistakes in
    textbook answer")
```

Scilab code Exa 14.15 Charging current drawn by cable

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES

// EXAMPLE : 7.15 :
// Page number 220-221
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 13.2*10**3      // Line Voltage(V)
f = 50.0            // Frequency(Hz)
C_BC = 4.2          // Capacitance b/w two cores( F )

// Calculations
C_n = 2.0*C_BC                                  // Capacitance to neutral( F )
V_ph = V/3**0.5                                 // Operating phase voltage(V)
I_c = 2.0*%pi*f*C_n*V/3**0.5*10**-6             // Charging current(A)

// Results
disp("PART II - EXAMPLE : 7.15 : SOLUTION :-")
printf("\nCharging current drawn by cable, I_c = %.2f A", I_c)
```

Scilab code Exa 14.16 Capacitance of the cable Charging current Total charging kVA

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 7: UNDERGROUND CABLES
```

```
// EXAMPLE : 7.16 :
// Page number 222-223
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 33.0*10**3     // Line Voltage(V)
f = 50.0           // Frequency(Hz)
l = 4.0            // Length(km)
d = 2.5            // Diameter of conductor(cm)
t = 0.5            // Radial thickness of insulation(cm)
e_r = 3.0          // Relative permittivity of the dielectric
PF = 0.02          // Power factor of unloaded cable

// Calculations
// Case(a)
r = d/2.0                                   // Radius of conductor(cm)
R = r+t                                     // External radius(cm)
e_0 = 8.85*10**-12                          // Permittivity
C = 2.0*%pi*e_0*e_r/log(R/r)*l*1000         // Capacitance of cable/phase(F)
// Case(b)
V_ph = V/3**0.5                             // Phase voltage(V)
I_c = V_ph*2.0*%pi*f*C                      // Charging current/phase(A)
// Case(c)
kVAR = 3.0*V_ph*I_c                         // Total charging kVAR
// Case(d)
phi = acosd(PF)                             // (  )
```

```
delta = 90.0-phi                                    // (  )
P_c = V_ph*I_c*sind(delta)/1000                     // Dielectric loss/phase(kW)
// Case(e)
E_max = V_ph/(r*log(R/r)*1000)                      // RMS value of Maximum stress in cable(kV/cm)

// Results
disp("PART II - EXAMPLE : 7.16 : SOLUTION :-")
printf("\nCase(a): Capacitance of the cable, C = %.3e F/phase", C)
printf("\nCase(b): Charging current = %.2f A/phase", I_c)
printf("\nCase(c): Total charging kVAR = %.4e kVAR", kVAR)
printf("\nCase(d): Dielectric loss/phase, P_c = %.2f kW", P_c)
printf("\nCase(e): Maximum stress in the cable, E_max = %.1f kV/cm (rms)", E_max)
```

Chapter 15

CORONA

Scilab code Exa 15.1 Minimum spacing between conductors

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 8: CORONA

// EXAMPLE : 8.1 :
// Page number 227
clear ; clc ; close ; // Clear the work space and
    console

// Given data
d = 30.0/10           // Diameter of conductor(cm)
delta = 0.95          // Air density factor
m = 0.95              // Irregularity factor
E = 230.0             // Line voltage(kV)
g_0 = 30.0/2**0.5     // Breakdown strength of air(kV
    /cm)

```

```
// Calculations
E_0 = E/3**0.5                              // Disruptive critical voltage(kV)
r = d/2.0                                   // Radius of conductor(cm)
D = exp(E_0/(m*delta*g_0*r))*r/100          // Minimum spacing between conductors(m)

// Results
disp("PART II - EXAMPLE : 8.1 : SOLUTION :-")
printf("\nMinimum spacing between conductors, D = %.3f m \n", abs(D))
printf("\nNOTE: Changes in obtained answer from that of textbook due to precision")
```

Scilab code Exa 15.2 `Critical disruptive voltage and Corona loss`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 8: CORONA

// EXAMPLE : 8.2 :
// Page number 227-228
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 220.0              // Operating line voltage(kV)
f = 50.0               // Frequency(Hz)
d = 1.5                // Diameter of conductor(cm)
D = 300.0              // Distance b/w conductor(cm)
```

```
delta = 1.05          // Air density factor
g_0 = 21.1            // Breakdown strength of air(kV/cm)
m = 1.0               // Irregularity factor

// Calculations
E = V/3**0.5                                        // Phase voltage(kV)
r = d/2.0                                           // Radius of conductor(cm)
E_0 = m*g_0*delta*r*log(D/r)                        // Disruptive critical voltage to neutral(kV/phase)
E_0_ll = 3**0.5*E_0                                 // Line-to-line Disruptive critical voltage(kV)
P = 244.0*10**-5*(f+25)/delta*(r/D)**0.5*(E-E_0)**2    // Corona loss(kW/km/phase)
P_total = P*3.0                                     // Corona loss(kW/km)

// Results
disp("PART II - EXAMPLE : 8.2 : SOLUTION :-")
printf("\nCritical disruptive voltage, E_0 = %.2f kV/phase = %.2f kV (line-to-line)", E_0,E_0_ll)
printf("\nCorona loss, P = %.2f kW/km \n", P_total)
printf("\nNOTE: ERROR: Calculation mistake in the final answer in textbook")
```

Scilab code Exa 15.3 Corona loss in fair weather and Foul weather

```
// A Texbook on POWER SYSTEM ENGINEERING
```

```
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 8: CORONA

// EXAMPLE : 8.3 :
// Page number 228
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 132.0              // Operating line voltage(kV)
f = 50.0               // Frequency(Hz)
d = 1.17               // Diameter of conductor(cm)
D = 300.0              // Distance b/w conductor(cm)
m = 0.96               // Irregularity factor
b = 72.0               // Barometric pressure(cm)
t = 20.0               // Temperature( C )

// Calculations
delta = 3.92*b/(273.0+t)
                                                  // Air
    density factor
r = d/2.0

    // Radius of conductor(cm)
E_0 = 21.1*m*delta*r*log(D/r)
                                             // Critical
    disruptive voltage for fair weather condition(kV/
    phase)
E_0_foul = 0.8*E_0
                                                        //
    Critical disruptive voltage for foul weather(kV/
    phase)
E = V/3**0.5
```

```
    // Phase voltage(kV)
P_fair = 244.0*10**-5*(f+25)/delta*(r/D)**0.5*(E-E_0
    )**2            // Corona loss for fair weather
    condition(kW/km/phase)
P_foul = 244.0*10**-5*(f+25)/delta*(r/D)**0.5*(E-
    E_0_foul)**2    // Corona loss for foul weather
    condition(kW/km/phase)

// Results
disp("PART II - EXAMPLE : 8.3 : SOLUTION :-")
printf("\nCorona loss in fair weather, P = %.3f kW/
    km/phase", P_fair)
printf("\nCorona loss in foul weather, P = %.3f kW/
    km/phase", P_foul)
```

Scilab code Exa 15.4 Corona characteristics

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 8: CORONA

// EXAMPLE : 8.4 :
// Page number 228-229
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 110.0                // Operating line voltage(kV)
f = 50.0                 // Frequency(Hz)
l = 175.0                // Line length(km)
d = 1.0                  // Diameter of conductor(cm)
```

```
D = 300.0               // Distance b/w conductor(cm)
t = 26.0                // Temperature( C )
b = 74.0                // Barometric pressure(cm)
m = 0.85                // Irregularity factor
m_v_local = 0.72        // Roughness factor for local corona
m_v_gen = 0.82          // Roughness factor for general corona

// Calculations
delta = 3.92*b/(273.0+t)                                     // Air density factor
r = d/2.0                                                    // Radius of conductor(cm)
E_0 = 21.1*m*delta*r*log(D/r)                                // Critical disruptive voltage(kV) rms
E_v_local = 21.1*m_v_local*delta*r*(1+(0.3/(delta*r)**0.5))*log(D/r)       // Critical disruptive voltage for local corona(kV) rms
E_v_gen = 21.1*m_v_gen*delta*r*(1+(0.3/(delta*r)**0.5))*log(D/r)           // Critical disruptive voltage for general corona(kV) rms
E = V/3**0.5                                                 // Phase voltage(kV)
// Case(i)
P_c_i = 244.0*10**-5*(f+25)/delta*(r/D)**0.5*(E-E_0)**2                    // Peek"s formula for fair weather condition(kW/km/phase)
P_c_i_total = P_c_i*l*3                                      // Total power loss(kW)
// Case(ii)
P_c_ii = 244.0*10**-5*(f+25)/delta*(r/D)**0.5*(E-0.8*E_0)**2               // Peek"s formula for
```

```
stormy condition(kW/km/phase)
P_c_ii_total = P_c_ii*l*3    // Total power loss(kW)
// Case(iii)
F_iii = 0.0713    // From text depending on E/E_0
P_c_iii = 21.0*10**-6*f*E**2*F_iii/(log10(D/r))**2    // Peterson"s formula for fair condition(kW/km/phase)
P_c_iii_total = P_c_iii*l*3    // Total power loss(kW)
// Case(iv)
F_iv = 0.3945    // From text depending on E/E_0
P_c_iv = 21.0*10**-6*f*E**2*F_iv/(log10(D/r))**2    // Peterson"s formula for stormy condition(kW/km/phase)
P_c_iv_total = P_c_iv*l*3    // Total power loss(kW)

// Results
disp("PART II - EXAMPLE : 8.4 : SOLUTION :-")
printf("\nCase(i)  : Power loss due to corona using Peek formula for fair weather condition, P_c  = %.3f kW/km/phase", P_c_i)
printf("\n            Total corona loss in fair weather condition using Peek formula = %.1f kW", P_c_i_total)
printf("\nCase(ii) : Power loss due to corona using Peek formula for stormy weather condition, P_c  = %.2f kW/km/phase", P_c_ii)
printf("\n            Total corona loss in stormy condition using Peek formula = %.f kW",
```

```
P_c_ii_total)
printf("\nCase(iii): Power loss due to corona using Peterson formula for fair weather condition, P_c = %.4f kW/km/phase", P_c_iii)
printf("\n           Total corona loss in fair condition using Peterson formula = %.2f kW", P_c_iii_total)
printf("\nCase(iii): Power loss due to corona using Peterson formula for fair weather condition, P_c = %.4f kW/km/phase", P_c_iv)
printf("\n           Total corona loss in stormy condition using Peterson formula = %.1f kW \n", P_c_iv_total)
printf("\nNOTE: ERROR: Calculation mistake in the final answer in textbook")
```

Scilab code Exa 15.5 Spacing between the conductors

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 8: CORONA

// EXAMPLE : 8.5 :
// Page number 229
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 132.0            // Operating line voltage(kV)
dia = 1.956          // Diameter of conductor(cm)
v_c = 210.0          // Disrputive voltage(kV)
```

```
g_0 = 30.0/2**0.5     // Breakdown strength of air(kV/cm)

// Calculations
r = dia/2.0                                   // Radius of conductor(cm)
V_c = v_c/3**0.5                              // Disrputive voltage/phase(kV)
m_0 = 1.0                                     // Irregularity factor
delta = 1.0                                   // Air density factor
d = exp(V_c/(m_0*delta*g_0*r))*r              // Spacing between conductors(cm)

// Results
disp("PART II - EXAMPLE : 8.5 : SOLUTION :-")
printf("\nSpacing between the conductors, d = %.f cm\n", abs(d))
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to precision")
```

Scilab code Exa 15.6 Disruptive critical voltage and Corona loss

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 8: CORONA

// EXAMPLE : 8.6 :
// Page number 229
clear ; clc ; close ; // Clear the work space and
```

```
    console

// Given data
P_c1 = 53.0             // Total corona loss(kW)
V_1 = 106.0             // Operating line voltage(kV)
P_c2 = 98.0             // Total corona loss(kW)
V_2 = 110.9             // Operating line voltage(kV)
V_3 = 113.0             // Operating line voltage(kV)

// Calculations
E_1 = V_1/3**0.5                              // Phase voltage(kV)
E_2 = V_2/3**0.5                              // Phase voltage(kV)
P_ratio = (P_c2/P_c1)**0.5
E_0 = (P_ratio*E_1-E_2)/(P_ratio-1)           // Disruptive critical voltage(kV)
E_3 = V_3/3**0.5                              // Phase voltage(kV)
W = ((E_3-E_0)/(E_1-E_0))**2*P_c1             // Corona loss at 113 kV(kW)

// Results
disp("PART II - EXAMPLE : 8.6 : SOLUTION :-")
printf("\nDisruptive critical voltage, E_0 = %.f kV", E_0)
printf("\nCorona loss at 113 kV, W = %.f kW\n", W)
printf("\nNOTE: Changes in obtained answer from textbook is due to more precision here")
```

Scilab code Exa 15.7 Corona will be present in the air space or not

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
```

```
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 8: CORONA

// EXAMPLE : 8.7 :
// Page number 229-230
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 3.0                  // Diameter of conductor(cm)
e_r = 4.0                // Relative permittivity
d_1 = 3.5                // Internal diameter of porcelain bushing(cm)
d_2 = 9.0                // External diameter of porcelain bushing(cm)
V = 25.0                 // Voltage b/w conductor and clamp(kV)

// Calculations
r = d/2.0                                          // Radius of conductor(cm)
r_1 = d_1/2.0                                      // Internal radius of porcelain bushing(cm)
r_2 = d_2/2.0                                      // External radius of porcelain bushing(cm)
g_2max = r/(e_r*r_1)                               // Maximum gradient of inner side of porcelain
g_1max = V/(r*log(r_1/r)+g_2max*r_1*log(r_2/r_1))            // Maximum gradient on surface of conductor(kV/cm)

// Results
```

```
disp("PART II - EXAMPLE : 8.7 : SOLUTION :-")
printf("\nMaximum gradient on surface of conductor,
    g_1max = %.2f kV/cm", g_1max)
printf("\nSince, gradient exceeds 21.1 kV/cm, corona
     will be present")
```

Scilab code Exa 15.8 Line voltage for commencing of corona

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 8: CORONA

// EXAMPLE : 8.8 :
// Page number 230
clear ; clc ; close ; // Clear the work space and
    console

// Given data
d = 2.0                // Diameter of conductor(cm)
D = 150.0              // Spacing b/w conductor(cm)
delta = 1.0            // Air density factor

// Calculations
r = d/2.0                                // Radius of
    conductor(cm)
V_d = 21.1*delta*r*log(D/r)              // Disruptive
    critical voltage(kV/phase)
V_d_ll = 3**0.5*V_d                      // Line voltage
    for commencing of corona(kV)

// Results
```

```
disp("PART II - EXAMPLE : 8.8 : SOLUTION :-")
printf("\nLine voltage for commencing of corona = %
    .2f kV \n", V_d_ll)
printf("\nNOTE: Solution is incomplete in textbook")
```

Chapter 16

LOAD FLOW STUDY USING COMPUTER TECHNIQUES

Scilab code Exa 16.1 Bus admittance matrix Ybus

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 9: LOAD FLOW STUDY USING COMPUTER
   TECHNIQUES

// EXAMPLE : 9.1 :
// Page number 235-236
clear ; clc ; close ; // Clear the work space and
   console

// Given data
Z_L1 = complex(14.3,97)        // Series impedance of
    line L1(ohm)
Z_PL1 = complex(0,-3274)       // Shunt impedance of
   line L1(ohm)
```

```
Z_L2 = complex(7.13,48.6)       // Series impedance of
     line L2(ohm)
Z_PL2 = complex(0,-6547)        // Shunt impedance of
    line L2(ohm)
Z_L3 = complex(9.38,64)         // Series impedance of
     line L3(ohm)
Z_PL3 = complex(0,-4976)        // Shunt impedance of
    line L3(ohm)

// Calculations
Y_S12 = 1.0/Z_L1                          // Series
    admittance(mho)
Y_P12 = 1.0/Z_PL1                         // Shunt
    admittance(mho)
Y_S23 = 1.0/Z_L3                          // Series
    admittance(mho)
Y_P23 = 1.0/Z_PL3                         // Shunt
    admittance(mho)
Y_S13 = 1.0/Z_L2                          // Series
    admittance(mho)
Y_P13 = 1.0/Z_PL2                         // Shunt
    admittance(mho)
Y_11 = Y_P12+Y_P13+Y_S12+Y_S13    // Admittance(mho)
Y_12 = -Y_S12                     // Admittance(mho)
Y_13 = -Y_S13                     // Admittance(mho)
Y_21 = Y_12                       // Admittance(mho)
Y_22 = Y_P12+Y_P23+Y_S12+Y_S23    // Admittance(mho)
Y_23 = -Y_S23                     // Admittance(mho)
Y_31 = Y_13                       // Admittance(mho)
Y_32 = Y_23                       // Admittance(mho)
Y_33 = Y_P13+Y_P23+Y_S23+Y_S13    // Admittance(mho)
Y_bus = [[Y_11, Y_12, Y_13],
         [Y_21, Y_22, Y_23],
         [Y_31, Y_32, Y_33]]

// Results
disp("PART II - EXAMPLE : 9.1 : SOLUTION :-")
printf("\n[Y_bus] = \n"); disp(Y_bus)
```

Scilab code Exa 16.3 Voltage values at different buses

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 9: LOAD FLOW STUDY USING COMPUTER TECHNIQUES

// EXAMPLE : 9.3 :
// Page number 236-237
clear ; clc ; close ; // Clear the work space and console

// Given data
V_a = 1.0                                   // Voltage(p.u)
V_b = 1.0*exp(%i*-36.87*%pi/180)            // Voltage(p.u)
V_c = 1.0                                   // Voltage(p.u)
Z_1 = complex(0,1)                          // Reactance(p.u)
Z_2 = complex(0,1)                          // Reactance(p.u)
Z_3 = complex(0,1)                          // Reactance(p.u)
Z_13 = complex(0,0.4)                       // Reactance(p.u)
Z_23 = complex(0,0.4)                       // Reactance(p.u)
Z_14 = complex(0,0.2)                       //
```

```
Reactance(p.u)
Z_24 = complex(0,0.2)                       // Reactance(p.u)
Z_34 = complex(0,0.2)                       // Reactance(p.u)
Z_12 = complex(0,0)                         // Reactance(p.u)

// Calculations
I_1 = V_a/Z_1           // Current injection vector(p.u)
I_2 = V_b/Z_2           // Current injection vector(p.u)
I_3 = V_c/Z_3           // Current injection vector(p.u)
I_4 = 0.0               // Current injection vector(p.u)
y1 = 1.0/Z_1            // Admittance(p.u)
y2 = 1.0/Z_2            // Admittance(p.u)
y3 = 1.0/Z_3            // Admittance(p.u)
y13 = 1.0/Z_13          // Admittance(p.u)
y23 = 1.0/Z_23          // Admittance(p.u)
y14 = 1.0/Z_14          // Admittance(p.u)
y24 = 1.0/Z_24          // Admittance(p.u)
y34 = 1.0/Z_34          // Admittance(p.u)
y12 = 0.0               // Admittance(p.u)
Y_11 = y1+y13+y14       // Equivalent admittance(p.u)
Y_12 = y12              // Equivalent admittance(p.u)
Y_13 = -y13             // Equivalent admittance(p.u)
Y_14 = -y14             // Equivalent admittance(p.u)
Y_21 = Y_12             // Equivalent admittance(p.u)
Y_22 = y2+y23+y24       // Equivalent admittance(p.u)
Y_23 = -y23             // Equivalent admittance(p.u)
Y_24 = -y24             // Equivalent admittance(p.u)
Y_31 = Y_13             // Equivalent admittance(p.u)
Y_32 = Y_23             // Equivalent admittance(p.u)
Y_33 = y3+y13+y23+y34   // Equivalent admittance(p.u)
Y_34 = -y34             // Equivalent admittance(p.u)
```

```
Y_41 = Y_14            // Equivalent admittance(p.u)
Y_42 = Y_24            // Equivalent admittance(p.u)
Y_43 = Y_34            // Equivalent admittance(p.u)
Y_44 = y14+y24+y34     // Equivalent admittance(p.u)
Y_bus = [[Y_11, Y_12, Y_13, Y_14],
         [Y_21, Y_22, Y_23, Y_24],
         [Y_31, Y_32, Y_33, Y_34],
         [Y_41, Y_42, Y_43, Y_44]]           // Bus
            admittance matrix
I_bus = [I_1,
         I_2,
         I_3,
         I_4]
V = inv(Y_bus)*I_bus                         // Bus
   voltage(p.u)

// Results
disp("PART II - EXAMPLE : 9.3 : SOLUTION :-")
printf("\nVoltage at bus 1, V_1 = %.4f%.4fj p.u",
   real(V(1,1:1)),imag(V(1,1:1)))
printf("\nVoltage at bus 2, V_2 = %.4f%.4fj p.u",
   real(V(2,1:1)),imag(V(2,1:1)))
printf("\nVoltage at bus 3, V_3 = %.4f%.4fj p.u",
   real(V(3,1:1)),imag(V(3,1:1)))
printf("\nVoltage at bus 4, V_4 = %.4f%.4fj p.u\n",
   real(V(4,1:1)),imag(V(4,1:1)))
printf("\nNOTE: Node equation matrix could not be
   represented in a single equation. Hence, it is
   not displayed")
```

Scilab code Exa 16.4 New bus admittance matrix Ybus

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
```

```
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 9: LOAD FLOW STUDY USING COMPUTER
   TECHNIQUES

// EXAMPLE : 9.4 :
// Page number 237-238
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V_a = 1.0                                     //
   Voltage(p.u)
V_b = 1.0*exp(%i*-36.87*%pi/180)              //
   Voltage(p.u)
V_c = 1.0                                     //
   Voltage(p.u)
Z_1 = complex(0,1)                            //
   Reactance(p.u)
Z_2 = complex(0,1)                            //
   Reactance(p.u)
Z_3 = complex(0,1)                            //
   Reactance(p.u)
Z_13 = complex(0,0.4)                         //
   Reactance(p.u)
Z_23 = complex(0,0.4)                         //
   Reactance(p.u)
Z_14 = complex(0,0.2)                         //
   Reactance(p.u)
Z_24 = complex(0,0.2)                         //
   Reactance(p.u)
Z_34 = complex(0,0.2)                         //
   Reactance(p.u)
Z_12 = complex(0,0)                           //
   Reactance(p.u)

// Calculations
```

```
I_1 = V_a/Z_1           // Current injection vector(p.u)
I_2 = V_b/Z_2           // Current injection vector(p.u)
I_3 = V_c/Z_3           // Current injection vector(p.u)
I_4 = 0.0               // Current injection vector(p.u)
y1 = 1.0/Z_1            // Admittance(p.u)
y2 = 1.0/Z_2            // Admittance(p.u)
y3 = 1.0/Z_3            // Admittance(p.u)
y13 = 1.0/Z_13          // Admittance(p.u)
y23 = 1.0/Z_23          // Admittance(p.u)
y14 = 1.0/Z_14          // Admittance(p.u)
y24 = 1.0/Z_24          // Admittance(p.u)
y34 = 1.0/Z_34          // Admittance(p.u)
y12 = 0.0               // Admittance(p.u)
Y_11 = y1+y13+y14       // Equivalent admittance(p.u)
Y_12 = y12              // Equivalent admittance(p.u)
Y_13 = -y13             // Equivalent admittance(p.u)
Y_14 = -y14             // Equivalent admittance(p.u)
Y_21 = Y_12             // Equivalent admittance(p.u)
Y_22 = y2+y23+y24       // Equivalent admittance(p.u)
Y_23 = -y23             // Equivalent admittance(p.u)
Y_24 = -y24             // Equivalent admittance(p.u)
Y_31 = Y_13             // Equivalent admittance(p.u)
Y_32 = Y_23             // Equivalent admittance(p.u)
Y_33 = y3+y13+y23+y34   // Equivalent admittance(p.u)
Y_34 = -y34             // Equivalent admittance(p.u)
Y_41 = Y_14             // Equivalent admittance(p.u)
Y_42 = Y_24             // Equivalent admittance(p.u)
Y_43 = Y_34             // Equivalent admittance(p.u)
Y_44 = y14+y24+y34      // Equivalent admittance(p.u)
Y_bus = [[Y_11, Y_12, Y_13, Y_14],
         [Y_21, Y_22, Y_23, Y_24],
         [Y_31, Y_32, Y_33, Y_34],
         [Y_41, Y_42, Y_43, Y_44]]               // Bus admittance matrix
```

```
K = Y_bus([1,2],1:2)
L = Y_bus([1,2],3:4)
M = Y_bus([3,4],3:4)
N = Y_bus([3,4],1:2)
inv_M = inv([M(1,1:2);M(2,1:2)])                    //
    Multiplication of marix [L][M^-1][N]
Y_bus_new = K-L*inv_M*N                             //
    New bus admittance matrix

// Results
disp("PART II - EXAMPLE : 9.4 : SOLUTION :-")
printf("\n[Y_bus]_new = \n"); disp(Y_bus_new)
printf("\nNOTE: ERROR: Mistake in representing the
    sign in final answer in textbook")
```

Scilab code Exa 16.5 Bus admittance matrix V1 and V2

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 9: LOAD FLOW STUDY USING COMPUTER
    TECHNIQUES

// EXAMPLE : 9.5 :
// Page number 238
clear ; clc ; close ; // Clear the work space and
    console

// Given data
I_1 = 2.0                                        //
    Voltage(p.u)
I_2 = 2.0*exp(%i*45.0*%pi/180)                   //
```

```
Voltage(p.u)
y1 = complex(0,-1.0)                         // Admittance(p.u)
y2 = complex(0,-2.0)                         // Admittance(p.u)
y12 = complex(0,-2.0)                        // Admittance(p.u)

// Calculations
E_1 = I_1*y1                                 // Voltage element(p.u)
E_2 = I_2*y2                                 // Voltage element(p.u)
Y_11 = y1+y12                                // Self Admittance(p.u)
Y_12 = -y12                                  // Mutual Admittance(p.u)
Y_21 = Y_12                                  // Mutual Admittance(p.u)
Y_22 = y2+y12                                // Self Admittance(p.u)
Y_bus = [[Y_11, Y_12],
         [Y_21, Y_22]]                       // Bus admittance matrix
I_bus = [I_1,
         I_2]
V = inv(Y_bus)*I_bus
V_1 = V(1,1:1)                               // Voltage(p.u)
V_2 = V(2,1:1)                               // Voltage(p.u)

// Results
disp("PART II - EXAMPLE : 9.5 : SOLUTION :-")
printf("\n[Y_bus] = \n"); disp(Y_bus)
printf("\nV_1 = %.3f %.1f   p.u", abs(V_1), phasemag(V_1))
printf("\nV_2 = %.3f %.1f   p.u\n", abs(V_2),
```

```
    phasemag(V_2))
printf("\nNOTE: ERROR: Calculation mistake in V_1 in
     textbook")
```

Scilab code Exa 16.6 Bus impedance matrix Zbus

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 9: LOAD FLOW STUDY USING COMPUTER
    TECHNIQUES

// EXAMPLE : 9.6 :
// Page number 238
clear ; clc ; close ; // Clear the work space and
    console

// Given data
Y_bus = [[-%i*10.5, 0, %i*5.0, %i*5.0],
         [0, -%i*8.0, %i*2.5, %i*5.0],
         [%i*5.0, %i*2.5, -%i*18.0, %i*10.0],
         [%i*5.0, %i*5.0, %i*10.0, -%i*20.0]]    //
            Bus admittance matrix

// Calculations
Z_bus = inv(Y_bus)                               //
    Bus impedance matrix

// Results
disp("PART II - EXAMPLE : 9.6 : SOLUTION :-")
printf("\n[Z_bus] = \n'); disp(Z_bus)
```

Scilab code Exa 16.7 Power flow expressions

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 9: LOAD FLOW STUDY USING COMPUTER TECHNIQUES

// EXAMPLE : 9.7 :
// Page number 239
clear ; clc ; close ; // Clear the work space and console

// Given data
Y_C = complex(0,0.1)            // Shunt admittance(mho)
Z_L = complex(0,0.2)            // Series impedance(mho)

// Calculations
Y_L = 1.0/Z_L                   // Series admittance(mho)
Y_11 = Y_C+Y_C+Y_L+Y_L          // Admittance(mho)
Y_12 = -Y_L                     // Admittance(mho)
Y_13 = -Y_L                     // Admittance(mho)
Y_21 = Y_12                     // Admittance(mho)
Y_22 = Y_L+Y_L+Y_C+Y_C          // Admittance(mho)
Y_23 = -Y_L                     // Admittance(mho)
Y_31 = Y_13                     // Admittance(mho)
Y_32 = Y_23                     // Admittance(mho)
Y_33 = Y_L+Y_L+Y_C+Y_C          // Admittance(mho)
```

```
Y_bus = [[Y_11, Y_12, Y_13],
         [Y_21, Y_22, Y_23],
         [Y_31, Y_32, Y_33]]     // Bus admittance matrix
S_11 = conj(Y_bus(1,1:1))
S_12 = conj(Y_bus(1,2:2))
S_13 = conj(Y_bus(1,3:3))
S_21 = S_12
S_22 = conj(Y_bus(2,2:2))
S_23 = conj(Y_bus(2,3:3))
S_31 = S_13
S_32 = S_23
S_33 = conj(Y_bus(3,3:3))

// Results
disp("PART II - EXAMPLE : 9.7 : SOLUTION :-")
printf("\nPower flow expressions are:")
printf("\nS_1 = %.1fj|V_1|^2 %.1fjV_1V_2* %.1fjV_3*", imag(S_11),imag(S_12),imag(S_13))
printf("\nS_2 = %.1fjV_2V_1* + %.1fj|V_2|^2 %.1fjV_2V_3*", imag(S_21),imag(S_22),imag(S_23))
printf("\nS_3 = %.1fjV_3V_1* %.1fjV_3V_2* + %.1fj|V_3|^2", imag(S_31),imag(S_32),imag(S_33))
```

Scilab code Exa 16.8 Voltage V2 by GS method

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 9: LOAD FLOW STUDY USING COMPUTER TECHNIQUES

```

```
// EXAMPLE : 9.8 :
// Page number 242
clear ; clc ; close ; // Clear the work space and console

// Given data
V_1 = 1.0                          // Voltage(p.u)
S_g2 = complex(0,1.0)              // Complex power generated(p.u)
S_D2 = complex(0.5,1.0)            // Complex power demand(p.u)
Z_L = complex(0,0.5)               // Impedance(p.u)

// Calculations
Y_L = 1.0/Z_L                                        // Admittance(p.u)
Y_22 = Y_L                                           // Admittance(mho)
Y_21 = -Y_L                                          // Admittance(mho)
S_2 = S_g2-S_D2
V_2_0 = 1.0                                          // Initial guess
V_2_1 = 1.0/Y_22*((conj(S_2/V_2_0))-Y_21*V_1)    // V_2(p.u). In 1st iteration
V_2_2 = 1.0/Y_22*((conj(S_2/V_2_1))-Y_21*V_1)    // V_2(p.u). In 2nd iteration
V_2_3 = 1.0/Y_22*((conj(S_2/V_2_2))-Y_21*V_1)    // V_2(p.u). In 3rd iteration
V_2_4 = 1.0/Y_22*((conj(S_2/V_2_3))-Y_21*V_1)    // V_2(p.u). In 4th iteration
V_2_5 = 1.0/Y_22*((conj(S_2/V_2_4))-Y_21*V_1)    // V_2(p.u). In 5th iteration
V_2_6 = 1.0/Y_22*((conj(S_2/V_2_5))-Y_21*V_1)    // V_2(p.u). In 6th iteration

// Results
disp("PART II - EXAMPLE : 9.8 : SOLUTION :-")
```

```
printf("\nBy G-S method, V_2 = %.6f % .5f    p.u\n",abs(V_2_6),phasemag(V_2_6))
```

Chapter 17

POWER SYSTEM STABILITY

Scilab code Exa 17.1 Operating power angle and Magnitude of P0

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.1 :
// Page number 270
clear ; clc ; close ; // Clear the work space and
    console

// Given data
Z = 0.1           // Impedance of transmission line(p.u
    )
M = 0.3           // Stability margin
X = 1.0           // Constant(p.u)

```

```
// Calculations
sin_delta_0 = 1-M                    // Sin( _0 )
delta_0 = asind(sin_delta_0)         //  _0 (  )
P_0 = X/Z*sin_delta_0                // Magnitude of P_0 (p.u)

// Results
disp("PART II - EXAMPLE : 10.1 : SOLUTION :-")
printf("\nOperating power angle,  _0  = %.2f  ", delta_0)
printf("\nP_0 = %.2f p.u", P_0)
```

Scilab code Exa 17.2 Minimum value of E and VL Maximum power limit and Steady stat

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.2 :
// Page number 270
clear ; clc ; close ; // Clear the work space and console

// Given data
x_s = 0.85        // Reactance(p.u)
x_T1 = 0.157      // Reactance(p.u)
x_T2 = 0.157      // Reactance(p.u)
x_l1 = 0.35       // Reactance(p.u)
x_l2 = 0.35       // Reactance(p.u)
E = 1.50          // Sending end voltage(p.u)
V_L = 1.0         // Load voltage(p.u)
```

```
P_0 = 1.0         // Stable power output(p.u)

// Calculations
x = x_s+x_T1+x_T2+(x_l1/2)          // Total reactance(p.u)
P_max = E*V_L/x                     // Maximum power limit(p.u)
M = (P_max-P_0)/P_max*100           // Steady state stability margin(%)
V_Lmin = P_0*x/E                    // Minimum value of V_L(p.u)
E_min = P_0*x/V_L                   // Minimum value of E(p.u)

// Results
disp("PART II - EXAMPLE : 10.2 : SOLUTION :-")
printf("\nMinimum value of |E|, |E_min| = %.3f p.u", E_min)
printf("\nMinimum value of |V_L|, |V_Lmin| = %.3f p.u", V_Lmin)
printf("\nMaximum power limit, P_0 = %.2f p.u", P_max)
printf("\nSteady state stability margin, M = %.1f percent", M)
```

Scilab code Exa 17.3 Maximum power transfer if shunt inductor and Shunt capacitor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

```

```
// EXAMPLE : 10.3 :
// Page number 270-271
clear ; clc ; close ; // Clear the work space and console

// Given data
E_1 = 1.25      // Sending end voltage(p.u)
x_d = 1.0       // Reactance(p.u)
x_T1 = 0.2      // Reactance(p.u)
x_l1 = 1.0      // Reactance(p.u)
x_l2 = 1.0      // Reactance(p.u)
x_T2 = 0.2      // Reactance(p.u)
E_2 = 1.0       // Receiving end voltage(p.u)
x_L = 1.0       // Shunt inductor reactance(p.u)
x_C = 1.0       // Shunt capacitor reactance(p.u)

// Calculations
// Case(a)
Z_1_a = x_d+x_T1+(x_l1/2.0)                 // Reactance(p.u)
Z_2_a = x_T2+x_d                            // Reactance(p.u)
Z_3_a = x_L                                 // Reactance(p.u)
Z_a = Z_1_a+Z_2_a+(Z_1_a*Z_2_a/Z_3_a)       // Transfer reactance(p.u)
P_max_1 = E_1*E_2/Z_a                       // Maximum power transfer if shunt inductor is connected at bus 2(p.u)
// Case(b)
Z_1_b = x_d+x_T1+(x_l1/2.0)                 // Reactance(p.u)
Z_2_b = x_T2+x_d                            // Reactance(p.u)
Z_3_b = -x_C                                // Reactance(p.u)
Z_b = Z_1_b+Z_2_b+(Z_1_b*Z_2_b/Z_3_b)       // Transfer reactance(p.u)
```

```
P_max_2 = E_1*E_2/Z_b                          // Maximum power transfer if shunt capacitor is connected at bus 2(p.u)

// Results
disp("PART II - EXAMPLE : 10.3 : SOLUTION :-")
printf("\nCase(a): Maximum power transfer if shunt inductor is connected at bus 2, P_max1 = %.3f p.u", P_max_1)
printf("\nCase(b): Maximum power transfer if shunt capacitor is connected at bus 2, P_max2 = %.2f p.u", P_max_2)
```

Scilab code Exa 17.4 Maximum power transfer and Stability margin

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.4 :
// Page number 271
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 400.0          // Voltage(kV)
L = 220.0          // Line length(km)
P = 0.58           // Initial real power transfer(p.u)
PF = 0.85          // Lagging power factor
V_L = 1.00         // Load bus voltage(p.u)
```

```
x_d = 0.460         // Reactance(p.u)
x_T1 = 0.200        // Reactance(p.u)
x_T2 = 0.15         // Reactance(p.u)
x_line = 0.7        // Reactance(p.u)

// Calculations
x = x_d+x_T1+x_T2+(x_line/2)                    // Net reactance(p.u)
phi = acosd(PF)                                 //   ( )
Q = P*tand(phi)                                 // Reactive power(p.u)
E = ((V_L+(Q*x/V_L))**2+(P*x/V_L)**2)**0.5      // Excitation voltage of generator(p.u)
P_max = E*V_L/x                                 // Maximum power transfer(p.u)
M = (P_max-P)/P_max*100                         // Steady state stability margin(%)

// Results
disp("PART II - EXAMPLE : 10.4 : SOLUTION :-")
printf("\nMaximum power transfer, P_max = %.2f p.u", P_max)
printf("\nStability margin, M = %.f percent", M)
```

Scilab code Exa 17.5 `QgB Phase angle of VB and What happens if QgB is made zero`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

```

```
// EXAMPLE : 10.5 :
// Page number 271-272
clear ; clc ; close ; // Clear the work space and console

// Given data
V_A = 1.0           // Voltage at bus A(p.u)
Z_AB = %i*0.5       // Impedance(p.u)
S_DA = 1.0          // p.u
S_DB = 1.0          // p.u
V_B = 1.0           // Voltage at bus B(p.u)

// Calculations
// Case(i) & (ii)
X = abs(Z_AB)                              // Reactance(p.u)
sin_delta = 1.0*X/(V_A*V_B)                // Sin
delta = asind(sin_delta)                   //   ( )
V_2 = V_B
V_1 = V_A
Q_gB = (V_2**2/X)-(V_2*V_1*cosd(delta)/X)
// Case(iii)
V_2_3 = 1/2.0**0.5                         // Solving quadratic equation from textbook
delta_3 = acosd(V_2_3)                     //   ( )

// Results
disp("PART II - EXAMPLE : 10.5 : SOLUTION :-")
printf("\nCase(i)  : Q_gB = %.3f", Q_gB)
printf("\nCase(ii) : Phase angle of V_B,    = %.f  ", delta)
printf("\nCase(iii): If Q_gB is equal to zero then amount of power transmitted is, V_2 = %.3f %  .f  ", V_2_3,delta_3)
```

Scilab code Exa 17.6 Steady state stability limit with two terminal voltages const

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.6 :
// Page number 272
clear ; clc ; close ; // Clear the work space and console
funcprot(0)

// Given data
A = 0.98*exp(%i*0.3*%pi/180)       // Constant
B = 82.5*exp(%i*76.0*%pi/180)      // Constant(ohm)
C = 0.0005*exp(%i*90.0*%pi/180)    // Constant(mho)
D = A                              // Constant
V_S = 110.0                        // Sending end voltage(kV)
V_R = 110.0                        // Receiving end voltage(kV)

// Calculations
alpha = phasemag(A)
    //   (  )
beta = phasemag(B)
    //   (  )
P_max = (V_S*V_R/abs(B))-(abs(A)*V_R**2/abs(B)*cosd
```

```
((beta-alpha)))   // Maximum power transfer(MW)
B_new = abs(B)*sind(beta)                                      // Constant(ohm)
beta_new = 90.0                                                // (  )
P_max_new = (V_S*V_R/B_new)-(V_R**2/B_new*cosd(beta_new))      // Maximum power transfer(MW)

// Results
disp("PART II - EXAMPLE : 10.6 : SOLUTION :-")
printf("\nSteady state stability limit, P_max = %.2f MW", P_max)
printf("\nSteady state stability limit if shunt admittance is zero & series resistance neglected, P_max = %.2f MW \n", P_max_new)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to precision")
```

Scilab code Exa 17.8 Power angle diagram Maximum power the line is capable of tran

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.8 :
// Page number 273-275
```

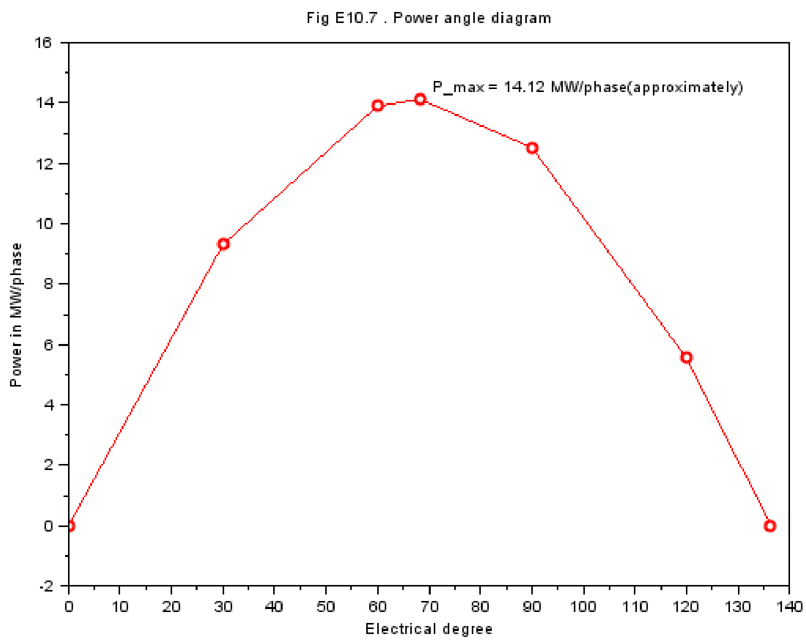

Figure 17.1: Power angle diagram Maximum power the line is capable of transmitting and Power transmitted with equal voltage at both ends

```
clear ; clc ; close ; // Clear the work space and console
funcprot(0)

// Given data
V = 33.0*10**3          // Line voltage(V)
R = 6.0                 // Resistance per phase(ohm)
X = 15.0                // Reactance per phase(ohm)

// Calculations
V_S = V/3**0.5                                                          // Sending end phase voltage(V)
V_R = V/3**0.5                                                          // Receiving end phase voltage(V)
beta = atand(X/R)                                                    //  ( )
Z = (R**2+X**2)**0.5                                              // Impedance(ohm)
delta_0 = 0.0                                                            // ( )
P_0 = (V_R/Z**2)*(V_S*Z*cosd((delta_0-beta))-V_R*R)/10**6    // Power received(MW/phase)
delta_1 = 30.0                                                          // ( )
P_1 = (V_R/Z**2)*(V_S*Z*cosd((delta_1-beta))-V_R*R)/10**6    // Power received(MW/phase)
delta_2 = 60.0                                                          // ( )
P_2 = (V_R/Z**2)*(V_S*Z*cosd((delta_2-beta))-V_R*R)/10**6    // Power received(MW/phase)
delta_3 = beta
```

```
// ( )
P_3 = (V_R/Z**2)*(V_S*Z*cosd((delta_3-beta))-V_R*R)/10**6    // Power received(MW/phase)
delta_4 = 90.0    // ( )
P_4 = (V_R/Z**2)*(V_S*Z*cosd((delta_4-beta))-V_R*R)/10**6    // Power received(MW/phase)
delta_5 = 120.0    // ( )
P_5 = (V_R/Z**2)*(V_S*Z*cosd((delta_5-beta))-V_R*R)/10**6    // Power received(MW/phase)
delta_6 = (acosd(R/Z))+beta    // ( )
P_6 = (V_R/Z**2)*(V_S*Z*cosd((delta_6-beta))-V_R*R)/10**6    // Power received(MW/phase)


delta = [delta_0,delta_1,delta_2,delta_3,delta_4,delta_5,delta_6]
P = [P_0,P_1,P_2,P_3,P_4,P_5,P_6]
a = gca() ;
a.thickness = 2    // sets thickness of plot
plot(delta,P,'ro-')
a.x_label.text = 'Electrical degree'    // labels x-axis
a.y_label.text = 'Power in MW/phase'    // labels y-axis
xtitle("Fig E10.7 . Power angle diagram")
xset('thickness',2)    // sets thickness of axes
xstring(70,14.12,'P_max = 14.12 MW/phase(approximately)')
P_max = V_R/Z**2*(V_S*Z-V_R*R)/10**6    // Maximum
```

```
power transmitted (MW/phase)
delta_equal = 0.0                                    //     With no phase shift (    )
P_no_shift = (V_R/Z**2)*(V_S*Z*cosd((delta_equal-beta))-V_R*R)/10**6    // Power transmitted with no phase shift (MW/phase)

// Results
disp("PART II - EXAMPLE : 10.8 : SOLUTION :-")
printf("\nPower angle diagram is plotted and is shown in the Figure 1")
printf("\nMaximum power the line is capable of transmitting , P_max = %.2f MW/phase", P_max)
printf("\nWith equal voltage at both ends power transmitted = %.f MW/phase", abs(P_no_shift))
```

Scilab code Exa 17.9 `Maximum steady state power that can be transmitted over the l`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.9 :
// Page number 275
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 132.0*10**3                    // Sending end voltage(V)
```

```
Z_line = complex(4,6)          // Line impedance per phase(ohm)

// Calculations
V_S = V/3**0.5                                             // Sending end phase voltage(V)
V_R = V/3**0.5                                             // Receiving end phase voltage(V)
Z = abs(Z_line)                                            // Impedance(ohm)
R = real(Z_line)                                           // Resistance per phase(ohm)
P_max_phase = ((V_S*V_R/Z)-(R*V_R**2/Z**2))/10**6     // Maximum steady state power that can be transmitted over the line(MW/phase)
P_max_total = 3.0*P_max_phase                          // Maximum steady state power that can be transmitted over the line(MW)

// Results
disp("PART II - EXAMPLE : 10.9 : SOLUTION :-")
printf("\nMaximum steady state power that can be transmitted over the line, P_max = %.f MW (total 3-phase)", P_max_total)
```

Scilab code Exa 17.10 Maximum steady state power Value of P and Q if static capaci

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION
```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.10 :
// Page number 275-276
clear ; clc ; close ; // Clear the work space and
    console

// Given data
E_1 = 1.1        // Sending end voltage(p.u)
x_d1 = 1.0       // Reactance(p.u)
x_T1 = 0.1       // Reactance(p.u)
x_l1 = 0.4       // Reactance(p.u)
x_l2 = 0.4       // Reactance(p.u)
x_T2 = 0.1       // Reactance(p.u)
E_2 = 1.0        // Receiving end voltage(p.u)
x_d2 = 1.0       // Reactance(p.u)
x_L = 1.0        // Shunt inductor reactance(p.u)
x_C = 1.0        // Static capacitor reactance(p.u)
delta = 30.0     //   (  )

// Calculations
// Case(a)
Z_1_a = x_d1+x_T1+(x_l1/2.0)
                              // Reactance(p.u)
X_1_a = %i*Z_1_a
Z_2_a = x_T2+x_d2
                                                //
    Reactance(p.u)
X_2_a = %i*Z_2_a
Z_3_a = -x_C
                                                     //
    Reactance(p.u)
X_3_a = %i*Z_3_a
X_a = X_1_a+X_2_a+(X_1_a*X_2_a/X_3_a)
                         // Transfer reactance(p.u)
P_max_a = E_1*E_2/abs(X_a)
```

```
                                        // Maximum steady state power if static capacitor is connected(p.u)
P_a = P_max_a*sind(delta)                                        // Value of P(p.u)
Q_a = (E_1*E_2/abs(X_a))*cosd(delta)-(E_2**2/abs(X_a))  // Value of Q(p.u)
// Case(b)
Z_1_b = x_d1+x_T1+(x_l1/2.0)                                    // Reactance(p.u)
X_1_b = %i*Z_1_b
Z_2_b = x_T2+x_d2                                                           // Reactance(p.u)
X_2_b = %i*Z_2_b
Z_3_b = x_L                                                                      // Reactance(p.u)
X_3_b = %i*Z_3_b
X_b = X_1_b+X_2_b+(X_1_b*X_2_b/X_3_b)                 // Transfer reactance(p.u)
P_max_b = E_1*E_2/abs(X_b)                                // Maximum steady state power if static capacitor is replaced by an inductive reactor(p.u)
P_b = P_max_b*sind(delta)                                        // Value of P(p.u)
Q_b = (E_1*E_2/abs(X_b))*cosd(delta)-(E_2**2/abs(X_b))  // Value of Q(p.u)

// Results
disp("PART II - EXAMPLE : 10.10 : SOLUTION :-")
printf("\nCase(a): Maximum steady state power if static capacitor is connected, P_max = %.3f p.u", P_max_a)
printf("\n         Value of P = %.3f p.u", P_a)
printf("\n         Value of Q = %.3f p.u", Q_a)
printf("\nCase(b): Maximum steady state power if static capacitor is replaced by an inductive
```

```
   reactor, P_max = %.3f p.u", P_max_b)
printf("\n          Value of P = %.3f p.u", P_b)
printf("\n          Value of Q = %.4f p.u", Q_b)
```

Scilab code Exa 17.11 Kinetic energy stored in the rotor at synchronous speed and

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.11 :
// Page number 303
clear ; clc ; close ; // Clear the work space and console

// Given data
f = 50.0          // Frequency(Hz)
G = 100.0         // Rating of generator(MVA)
H = 5.0           // Inertia constant(MJ/MVA)
P_a = 20.0        // Acceleration power(MVA)

// Calculations
GH = G*H                  // Energy stored in rotor at synchronous speed(MJ)
M = GH/(180*f)            // Angular momentum
acceleration = P_a/M      // Acceleration( /sec^2)

// Results
disp("PART II - EXAMPLE : 10.11 : SOLUTION :-")
printf("\nKinetic energy stored in the rotor at synchronous speed, GH = %.f MJ", GH)
```

```
printf("\nAcceleration = %.f /sec^2", acceleration)
```

Scilab code Exa 17.12 Kinetic energy stored in the rotor at synchronous speed and

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.12 :
// Page number 303-304
clear ; clc ; close ; // Clear the work space and
    console

// Given data
f = 50.0              // Frequency(Hz)
P = 4.0               // Number of poles
G = 20.0              // Rating of generator(MVA)
H = 9.0               // Inertia constant(kWsec/MVA)
P_m = 26800.0         // Rotational loss(hp)
P_e = 16000.0         // Electric power developed(kW)

// Calculations
GH = G*H                                              //
    Energy stored in rotor at synchronous speed(MJ)
P_m_kW = P_m*0.746                                    //
    Rotational loss(kW)
P_a = P_m_kW-P_e                                      //
    Acceleration power(kW)
P_a1 = P_a/1000.0                                     //
    Acceleration power(MW)
M = GH/(180*f)                                        //
```

```
    Angular momentum
acceleration = P_a1/M                         // Acceleration(  /sec^2)
acceleration_1 = acceleration*%pi/180.0       // Acceleration(rad/sec^2)

// Results
disp("PART II - EXAMPLE : 10.12 : SOLUTION :-")
printf("\nKinetic energy stored in the rotor at synchronous speed, GH = %.f MJ", GH)
printf("\nAcceleration = %.f  /sec^2 = %.2f rad/sec^2 \n", acceleration,acceleration_1)
printf("\nNOTE: ERROR: H = 9 kW-sec/MVA, not 9 kW-sec/kVA as mentioned in the textbook statement")
```

Scilab code Exa 17.13 Change in torque angle in that period and RPM at the end of

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.13 :
// Page number 304
clear ; clc ; close ; // Clear the work space and console

// Given data
f = 50.0            // Frequency(Hz)
P = 4.0             // Number of poles
alpha = 200.0       // Acceleration(  /sec^2)
alpha_rad = 3.49    // Acceleration(rad/sec^2)
```

```
n = 10.0             // Number of cycle

// Calculations
t = 1/f*n                                    // Time(sec)
delta_rel = ((alpha_rad*2)**0.5*0.5)**2   // Relation of change in rotor angle with time(rad)
delta = delta_rel*t**2                       // Change in torque angle(rad)
delta_deg = delta*180/%pi                    // Change in torque angle in that period(  )
rpm_rad = (alpha_rad*2*delta)**0.5           // r.p.m(rad/sec)
rpm = rpm_rad*60.0/(%pi*P)                   // r.p.m
speed_rotor = (120*f/P)+rpm                  // Rotor speed at the end of 10 cycles(r.p.m)

// Results
disp("PART II - EXAMPLE : 10.13 : SOLUTION :-")
printf("\nChange in torque angle in that period, = %.4f rad = %.f elect degree", delta,delta_deg)
printf("\nRotor speed at the end of 10 cycles = %.2f r.p.m", speed_rotor)
```

Scilab code Exa 17.14 Accelerating torque at the time the fault occurs

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.14 :
```

```
// Page number 304
clear ; clc ; close ; // Clear the work space and console

// Given data
Power = 20.0*10**3          // Rating of generator(kVA)
PF = 0.8                    // Lagging power factor
fault = 0.5                 // Reduction in output under fault
P = 4.0                     // Number of poles
f = 50.0                    // Frequency(Hz)

// Calculations
P_m = Power*PF              // Output power before fault(kW)
P_e = fault*P_m             // Output after fault(kW)
P_a = P_m-P_e               // Accelerating power(kW)
w_s = 4.0*%pi*f/P           // Speed
T_a = P_a*10**3/w_s         // Accelerating torque at the time the fault occurs(N-m)

// Results
disp("PART II - EXAMPLE : 10.14 : SOLUTION :-")
printf("\nAccelerating torque at the time the fault occurs, T_a = %.2f N-m", T_a)
```

Scilab code Exa 17.16 Value of H and in 100 MVA base

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY
```

```

// EXAMPLE : 10.16 :
// Page number 304-305
clear ; clc ; close ; // Clear the work space and console

// Given data
S = 1000.0            // Rating of generator(MVA)
N = 1500.0            // Speed of alternator(r.p.m)
WR_sq = 5.0*10**6     // WR^2(lb.ft^2)

// Calculations
H = 2.31*10**-10*WR_sq*N**2/S       // Inertia constant(MJ/MVA)
H_100 = H*1000.0/100                // Inertia constant on 100 MVA(MJ/MVA)

// Results
disp("PART II - EXAMPLE : 10.16 : SOLUTION :-")
printf("\nValue of inertia constant , H = %.1f MJ/MVA", H)
printf("\nValue of inertia constant in 100 MVA base , H = %.f MJ/MVA", H_100)
```

Scilab code Exa 17.17 Equivalent H for the two to common 100 MVA base

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.17 :
```

```
// Page number 305
clear ; clc ; close ; // Clear the work space and console

// Given data
MVA_1 = 500.0          // Rating of generator(MVA)
H_1 = 4.0              // Inertia constant(MJ/VA)
MVA_2 = 1000.0         // Rating of generator(MVA)
H_2 = 3.5              // Inertia constant(MJ/VA)
MVA = 100.0            // Base MVA

// Calculations
KE_T = H_1*MVA_1+H_2*MVA_2      // Total KE of the system(MJ)
H_total = KE_T/MVA              // Equivalent H for the two to common 100MVA base(MJ/MVA)

// Results
disp("PART II - EXAMPLE : 10.17 : SOLUTION :-")
printf("\nEquivalent H for the two to common 100 MVA base, H = %.f MJ/MVA", H_total)
```

Scilab code Exa 17.18 Energy stored in the rotor at the rated speed Value of H and

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.18 :
// Page number 305
clear ; clc ; close ; // Clear the work space and
```

```
   console

// Given data
MVA = 210.0              // Rating of generator(MVA)
P = 2.0                  // Number of poles
f = 50.0                 // Frequency(Hz)
MI = 60.0*10**3          // Moment of inertia(kg-mt^2)

// Calculations
N = 120.0*f/P                                  // Speed(r.p.m)
KE = 1.0/2*MI*(2*%pi*N/f)**2/10**6             // Energy stored in the rotor at rated speed(MJ)
H = KE/MVA                                     // Inertia constant(MJ/MVA)
G = MVA
M = G*H/(180*f)                                // Angular momentum(MJ-sec/elect.degree)

// Results
disp("PART II - EXAMPLE : 10.18 : SOLUTION :-")
printf("\nEnergy stored in the rotor at the rated speed, KE = %.2e MJ", KE)
printf("\nValue of inertia constant, H = %.2f MJ/MVA", H)
printf("\nAngular momentum, M = %.3f MJ-sec/elect.degree", M)
```

Scilab code Exa 17.19 `Acceleration of the rotor`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.19 :
// Page number 305
clear ; clc ; close ; // Clear the work space and
    console

// Given data
P_accl = 30.0          // Acceleration power(MVA)
M = 0.474              // Angular momentum(MJ-sec/
    elect.degree). From Example 10.18

// Calculations
acceleration = P_accl/M       // Acceleration of the
    rotor(elect.degree/sec^2)

// Results
disp("PART II - EXAMPLE : 10.19 : SOLUTION :-")
printf("\nAcceleration of the rotor = %.2f elect.
    degree/sec^2", acceleration)
```

Scilab code Exa 17.20 Accelerating power and New power angle after 10 cycles

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.20 :
// Page number 305
clear ; clc ; close ; // Clear the work space and
```

```
console

// Given data
MVA = 50.0              // Rating of alternator(MVA)
P = 4.0                 // Number of poles
f = 50.0                // Frequency(Hz)
KE = 150.0              // Kinetic energy stored in rotor(MJ)
P_m = 25.0              // Machine input(MW)
P_e = 22.5              // Developed power(MW)
n = 10.0                // Number of cycles

// Calculations
P_a = P_m-P_e                 // Accelerating power(MW)
H = KE/MVA                    // Inertia constant(MJ/MVA)
G = MVA
M_deg = G*H/(180*f)           // Angular momentum(MJ-sec/elect.degree)
M = G*H/(%pi*f)               // Angular momentum(MJ-sec/rad)
acceleration = P_a/M          // Accelerating power(rad/sec^2)
t = 1/f*n                     // Time(sec)
delta = 1.309*t**2            // Term in

// Results
disp("PART II - EXAMPLE : 10.20 : SOLUTION :-")
printf("\nAccelerating power = %.3f rad/sec^2", acceleration)
printf("\nNew power angle after 10 cycles,  = (%.3f +  _0 ) rad", delta)
```

Scilab code Exa 17.21 `Kinetic energy stored by rotor at synchronous speed and Acce`

```
// A Texbook on POWER SYSTEM ENGINEERING
```

```
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.21 :
// Page number 305-306
clear ; clc ; close ; // Clear the work space and
    console

// Given data
f = 50.0          // Frequency(Hz)
P = 4.0           // Number of poles
G = 20.0          // Rating of turbo-generator(MVA)
V = 13.2          // Voltage(kV)
H = 9.0           // Inertia constant(kW-sec/kVA)
P_s = 20.0        // Input power less rotational loss(
   MW)
P_e = 15.0        // Output power(MW)

// Calculations
KE = G*H                            // Kinetic energy
    stored(MJ)
M = G*H/(180*f)                     // Angular momentum
    (MJ-sec/elect.degree)
P_a = P_s-P_e                       // Accelerating
    power(MW)
alpha = P_a/M                       // Acceleration(
    elect.degree/sec^2)
alpha_deg = alpha/2.0               // Acceleration(
    degree/sec^2)
alpha_rpm = 60.0*alpha_deg/360      // Acceleration(rpm
    /sec)

// Results
disp("PART II - EXAMPLE : 10.21 : SOLUTION :-")
```

```
printf("\nCase(a): Kinetic energy stored by rotor at synchronous speed, GH = %.f MJ", KE)
printf("\nCase(b): Acceleration,     = %.f degree/sec^2", alpha_deg)
printf("\n             Acceleration,     = %.2f rpm/sec", alpha_rpm)
```

Scilab code Exa 17.22 Change in torque angle and Speed in rpm at the end of 10 cyc

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.22 :
// Page number 306
clear ; clc ; close ; // Clear the work space and console

// Given data
f = 50.0          // Frequency(Hz)
P = 4.0           // Number of poles
G = 20.0          // Rating of turbo-generator(MVA)
V = 13.2          // Voltage(kV)
H = 9.0           // Inertia constant(kW-sec/kVA)
P_s = 20.0        // Input power less rotational loss(MW)
P_e = 15.0        // Output power(MW)
n = 10.0          // Number of cycles

// Calculations
KE = G*H                                  // Kinetic energy
```

```
    stored(MJ)
M = G*H/(180*f)                   // Angular momentum (MJ-sec/elect.degree)
P_a = P_s-P_e                     // Accelerating power(MW)
alpha = P_a/M                     // Acceleration(elect.degree/sec^2)
alpha_deg = alpha/2.0             // Acceleration(degree/sec^2)
alpha_rpm = 60.0*alpha_deg/360    // Acceleration(rpm/sec)
t = 1.0/f*n                       // Time(sec)
delta = 1.0/2*alpha*t**2          // Change in torque angle(elect.degree)
N_s = 120*f/P                     // Synchronous speed(rpm)
speed = N_s+alpha_rpm*t           // Speed at the end of 10 cycles(rpm)

// Results
disp("PART II - EXAMPLE : 10.22 : SOLUTION :-")
printf("\nChange in torque angle in that period, = %.f elect degrees.", delta)
printf("\nSpeed in rpm at the end of 10 cycles = %.2f rpm", speed)
```

Scilab code Exa 17.23 Accelerating torque at the time of fault occurrence

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY
```

```

// EXAMPLE : 10.23 :
// Page number 306
clear ; clc ; close ; // Clear the work space and console

// Given data
G = 20.0          // Rating of turbo-generator(MVA)
PF = 0.75         // Lagging power factor
fault = 0.5       // Fault reduces output power
N_s = 1500.0      // Synchronous speed(rpm). From Example 10.22

// Calculations
P_prefault = PF*G           // Pre-fault output power(MW)
P_a = P_prefault*fault      // Post-fault output power(MW)
w = 2.0*%pi*N_s/60          //   (rad/sec)
T_a = P_a*10**6/w           // Accelerating torque at the time of fault occurrence(N-m)

// Results
disp("PART II - EXAMPLE : 10.23 : SOLUTION :-")
printf("\nAccelerating torque at the time of fault occurrence, T_a = %.f N-m", T_a)
```

Scilab code Exa 17.24 Swing equation

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
```

```
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.24 :
// Page number 306-307
clear ; clc ; close ; // Clear the work space and console

// Given data
x_d = %i*0.2          // Transient reactance of generator(p.u)
P_e = 0.8             // Power delivered(p.u)
V_t = 1.05            // Terminal voltage(p.u)
H = 4.0               // Inertia constant(kW-sec/kVA)
x_t = %i*0.1          // Transformer reactance(p.u)
x_l = %i*0.4          // Transmission line reactance(p.u)
V = 1.0               // Infinite bus voltage(p.u)
f = 50.0              // Frequency(Hz)

// Calculations
x_12 = x_d+x_t+(x_l/2)                                      // Reactance b/w bus 1 & 2(p.u)
y_12 = 1/x_12                                                // Admittance b/w bus 1 & 2(p.u)
y_21 = y_12                                                  // Admittance b/w bus 2 & 1(p.u)
y_10 = 0.0                                                   // Admittance b/w bus 1 & 0(p.u)
y_20 = 0.0                                                   // Admittance b/w bus 2 & 0(p.u)
Y_11 = y_12+y_10                                             // Admittance at bus 1(p.u)
```

```
Y_12 = -y_12                                                    // Admittance b/w bus 1 & 2(p.u)
Y_21 = -y_12                                                    // Admittance b/w bus 2 & 1(p.u)
Y_22 = y_21+y_20                                                // Admittance at bus 2(p.u)
x_32 = x_t+(x_l/2)                                              // Reactance b/w bus 3 & 1(p.u)
theta_t = asind(P_e*abs(x_32)/V_t)                              // Angle( )
V_t1 = V_t*exp(%i*theta_t*%pi/180)                              // Terminal voltage(p.u)
I = (V_t1-V)/x_32                                               // Current(p.u)
E = V_t1+I*x_d                                                  // Alternator voltage(p.u)
sine = poly(0,"sin")
P_e1 = 2.0*abs(E)                                               // Developed power(p.u) in terms of sin
P_m_P_e = P_e-P_e1*sine
M = 2*H/(2*%pi*f)                                               // Angular momentum
acc = (P_e-P_e1*sine)*2*%pi*f/(2*H)                             // Acceleration =   (rad/sec^2)

// Results
disp("PART II - EXAMPLE : 10.24 : SOLUTION :-")
printf("\nSwing equation is, %.4f*   = %.1f - %.3fsin   \n", M,P_e,P_e1)
```

```
printf("\nNOTE: Swing equation is simplified and
    represented here")
printf("\n       ERROR: x_d = 0.2 p.u, not 0.1 p.u as
     mentioned in textbook statement")
```

Scilab code Exa 17.26 Critical clearing angle

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.26 :
// Page number 308-309
clear ; clc ; close ; // Clear the work space and
    console

// Given data
X_d = 0.25           // Transient reactance of
    generator(p.u)
X_t1 = 0.15          // Reactance of transformer(p.u)
X_t2 = 0.15          // Reactance of transformer(p.u)
X_t3 = 0.15          // Reactance of transformer(p.u)
X_t4 = 0.15          // Reactance of transformer(p.u)
X_l1 = 0.20          // Reactance of line(p.u)
X_l2 = 0.20          // Reactance of line(p.u)
X_tr = 0.15          // Reactance of transformer(p.u)
P_m = 1.0            // Power delivered(p.u)
E = 1.20             // Voltage behind transient
    reactance(p.u)
V = 1.0              // Infinite bus voltage(p.u)

```

```
// Calculations
X_14 = X_d+((X_t1+X_t2+X_l1)/2)+X_tr                        // Reactance before fault(p.u)
x_1_b = X_t1+X_t2+X_l1                                       // Reactance(p.u). From figure (b)
x_2_b = X_l2+X_t4                                            // Reactance(p.u). From figure (b)
x_1 = x_1_b*X_t3/(x_1_b+x_2_b+X_t3)                          // Reactance(p.u). From figure (c)
x_2 = x_1_b*x_2_b/(x_1_b+x_2_b+X_t3)                         // Reactance(p.u). From figure (c)
x_3 = X_t3*x_2_b/(x_1_b+x_2_b+X_t3)                          // Reactance(p.u). From figure (c)
X_14_fault = x_1+X_d+x_2+X_tr+((x_1+X_d)*(x_2+X_tr)/x_3)    // Reactance under fault(p.u)
X_14_after_fault = X_d+X_t1+X_l1+X_t2+X_tr                   // Reactance after fault is cleared(p.u)
P_max = V*E/X_14                                             // Maximum power transfer(p.u)
gamma_1 = (V*E/X_14_fault)/P_max                             //  _1
gamma_2 = (V*E/X_14_after_fault)/P_max                       //  _2
delta_0 = asin(P_m/P_max)                                    //  _0 (radians)
delta_0_degree = delta_0*180/%pi                             //  _0 ( )
delta_m = %pi-asin(P_m/(gamma_2*P_max))                      //  _1 (radians)
delta_m_degree = delta_m*180/%pi
```

```
                                    //   _1  (   )
delta_c = acosd((P_m/P_max*(delta_m-delta_0)+gamma_2
    *cos(delta_m)-gamma_1*cos(delta_0))/(gamma_2-
    gamma_1)) // Clearing angle(   )

// Results
disp("PART II - EXAMPLE : 10.26 : SOLUTION :-")
printf("\nCritical clearing angle,   _c   = %.2f    ",
    delta_c)
```

Scilab code Exa 17.27 Critical angle using equal area criterion

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.27 :
// Page number 309-310
clear ; clc ; close ; // Clear the work space and
    console

// Given data
f = 50.0                  // Frequency(Hz)
P_m = 1.0                 // Power delivered(p.u)
P_max = 1.8               // Maximum power(p.u)
gamma_1_P_max = 0.4       // Reduced maximum power
    after fault(p.u)
gamma_2_P_max = 1.30      // Maximum power after fault
    clearance(p.u)

// Calculations
```

```
delta_0 = asin(P_m/P_max)                    //  _0 (radians)
delta_0_degree = delta_0*180/%pi             //  _0 (  )
delta_f = %pi-asin(P_m/(gamma_2_P_max))      //  _1 (radians)
delta_f_degree = delta_f*180/%pi             //  _1 (  )
gamma_1 = gamma_1_P_max/P_max                //  _1
gamma_2 = gamma_2_P_max/P_max                //  _2
delta_c = acosd(1.0/(gamma_2-gamma_1)*((delta_f-
    delta_0)*sin(delta_0)+(gamma_2*cos(delta_f)-
    gamma_1*cos(delta_0)))) // Clearing angle(  )

// Results
disp("PART II - EXAMPLE : 10.27 : SOLUTION :-")
printf("\nCritical angle,  _c = %.2f  ", delta_c)
```

Scilab code Exa 17.28 Critical clearing angle

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.28 :
// Page number 310
clear ; clc ; close ; // Clear the work space and
    console

```

```
// Given data
sin_delta_0 = 0.45      // Supplying percent of peak
    power capacity before fault
x = 4.0                 // Reactance under fault
    increased
gamma_2 = 0.7           // Peak power delivered after
     fault clearance

// Calculations
delta_0 = asin(sin_delta_0)
    //  _0 (radians)
delta_0_degree = delta_0*180/%pi
    //  _0 (  )
gamma_1 = 1.0/x
    //  _1
delta_m = %pi-asin(sin_delta_0/(gamma_2))
    //  _m (radians)
delta_m_degree = delta_m*180/%pi
    //  _m (  )
delta_c = acosd(1.0/(gamma_2-gamma_1)*((delta_m-
    delta_0)*sin(delta_0)+(gamma_2*cos(delta_m)-
    gamma_1*cos(delta_0)))) // Clearing angle(  )

// Results
disp("PART II - EXAMPLE : 10.28 : SOLUTION :-")
printf("\nCritical clearing angle,  _c  = %.f  ",
    delta_c)
```

Scilab code Exa 17.30 Power angle and Swing curve data

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 10: POWER SYSTEM STABILITY

// EXAMPLE : 10.30 :
// Page number 310-311
clear ; clc ; close ; // Clear the work space and console

// Given data
f = 60.0          // Frequency(Hz)
P = 6.0           // Number of poles
H = 4.0           // Inertia constant(p.u)
P_e = 1.0         // Power supplied by generator(p.u)
E = 1.2           // Internal voltage(p.u)
V = 1.0           // Infinite bus voltage(p.u)
X = 0.3           // Line reactance(p.u)
del_t = 0.05      //  t  = Interval step size(sec)

// Calculations
P_max = E*V/X                           // Maximum power(p.u)
delta_0 = asind(P_e/P_max)              //  _0 (  )
G = P_e
M = G*H/(180*f)                         // Angular momentum(p.u)
P_a_0 = 1.0/2*(P_e-0)                   // (p.u)
alpha_0 = P_a_0/M                       //  _0 (  /sec^2)
del_w_r_1 = alpha_0*del_t               //  _r_1  (  /sec)
w_r_1 = 0+del_w_r_1                     //  _r_1 (  /sec)
del_delta_1 = w_r_1*del_t               //  _1 (  )
delta_1 = delta_0+del_delta_1           //  _1
```

```
    (   )
P_a_1 = 1.0*(P_e-0)                                   // (p.u
    )
alpha_1 = P_a_1/M                                     //   _1
    (   /sec^2)
del_w_r_2 = alpha_1*del_t                             //
        _r_2    (   /sec)
w_r_2 = del_w_r_1+del_w_r_2                           //
      _r_2   (   /sec)
del_delta_2 = w_r_2*del_t                             //
       _2   (   )
delta_2 = delta_1+del_delta_2                         //   _2
    (   )
del_w_r_3 = del_w_r_2                                 //
        _r_3    (   /sec)
w_r_3 = w_r_2+del_w_r_3                               //
      _r_3   (   /sec)
del_delta_3 = w_r_3*del_t                             //
       _3   (   )
delta_3 = delta_2+del_delta_3                         //   _3
    (   )
del_w_r_4 = del_w_r_2                                 //
        _r_4    (   /sec)
w_r_4 = w_r_3+del_w_r_4                               //
      _r_4   (   /sec)
del_delta_4 = w_r_4*del_t                             //
       _4   (   )
delta_4 = delta_3+del_delta_4                         //   _4
    (   )
del_w_r_5 = del_w_r_2                                 //
        _r_5    (   /sec)
w_r_5 = w_r_4+del_w_r_5                               //
      _r_5   (   /sec)
del_delta_5 = w_r_5*del_t                             //
       _5   (   )
delta_5 = delta_4+del_delta_5                         //   _5
    (   )

```

```
// Results
disp("PART II - EXAMPLE : 10.30 : SOLUTION :-")
printf("\nPower angle ,  _0  = %.2f   ", delta_0)
printf("\nValue of    vs t are:")
printf("\n_________________________")
printf("\n  t(Sec)   :      (degree)")
printf("\n_________________________")
printf("\n   %.1f       :     %.2f  ", 0,delta_0)
printf("\n   %.2f      :    %.2f  ", (del_t),delta_1)
printf("\n   %.2f      :    %.2f  ", (del_t+del_t),delta_2)
printf("\n   %.2f      :    %.2f  ", (del_t*3),delta_3)
printf("\n   %.2f      :    %.2f  ", (del_t*4),delta_4)
printf("\n   %.2f      :    %.2f  ", (del_t*5),delta_5)
printf("\n_________________________")
```

Chapter 18

LOAD FREQUENCY CONTROL AND LOAD SHARING OF POWER GENERATING SOURCES

Scilab code Exa 18.1 Load shared by two machines and Load at which one machine cea

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD
   SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.1 :
// Page number 330
clear ; clc ; close ; // Clear the work space and
   console
funcprot(0)

```

```
// Given data
rating = 1000.0          // Rating of alternator(kW)
load = 1600.0            // Total load(kW)
X_fl = 100.0             // Full load speed regulation of alernator X(%)
Y_fl = 104.0             // Full load speed regulation of alernator Y(%)
X_nl = 100.0             // No load speed regulation of alernator X(%)
Y_nl = 105.0             // No load speed regulation of alernator Y(%)

// Calculations
h = poly(0,"h")
PB = (Y_nl-X_nl)-h
PR = rating/(Y_nl-X_nl)*PB              // Load shared by machine X(kW) in terms of h
QQ = (Y_fl-X_fl)-h
RQ = rating/(Y_fl-X_fl)*QQ              // Load shared by machine Y(kW) in terms of h
h_1 = roots(PR+RQ-load)
PB_1 = (Y_nl-X_nl)-h_1
PR_1 = rating/(Y_nl-X_nl)*PB_1          // Load shared by machine X(kW)
QQ_1 = (Y_fl-X_fl)-h_1
RQ_1 = rating/(Y_fl-X_fl)*QQ_1          // Load shared by machine Y(kW)
load_cease = rating/(Y_nl-X_nl)         // Y cease supply load(kW)

// Results
disp("PART II - EXAMPLE : 11.1 : SOLUTION :-")
printf("\nLoad shared by machine X, PR = %.f kW", PR_1)
printf("\nLoad shared by machine Y, RQ = %.f kW", RQ_1)
printf("\nLoad at which machine Y ceases to supply any portion of load = %.f kW", load_cease)
```

Scilab code Exa 18.2 Synchronizing power and Synchronizing torque for no load and

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD
   SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.2 :
// Page number 330-331
clear ; clc ; close ; // Clear the work space and
   console

// Given data
kVA = 5000.0         // Rating of alternator(kVA)
N = 1500.0           // Speed(rpm)
V = 6600.0           // Voltage(V)
f = 50.0             // Frequency(Hz)
PF = 0.8             // Lagging power factor
x = 0.15             // Short circuit reactance

// Calculations
E = V/3**0.5
   // Phase voltage(V)
I = kVA*1000/(3**0.5*V)
                                               // Full
   load current of alternator(A)
V_drop = E*x
   // Synchronous reactance drop(V)
```

```
X = V_drop/I                // Synchronous reactance per phase(ohm)
P = 120*f/N                 // Number of poles
n = N/60                    // Speed(rps)
phi = acosd(PF)             //   (   )
// Case(a)
theta_a = 2.0               // For a 4 pole m/c. 1 mech degree = 2 elect degree
E_s_a = E*sind(theta_a)                                        // Synchronizing voltage(V)
I_s_a = E_s_a/X             // Synchronizing current(A)
P_s_a = E*I_s_a             // Synchronizing power per phase(W)
P_s_a_total = 3.0*P_s_a                                        // Total synchronizing power(W)
P_s_a_total_kw = P_s_a_total/1000.0                  // Total synchronizing power(kW)
T_s_a = P_s_a_total/(2*%pi*n)                             // Synchronizing torque(N-m)
// Case(b)
sin_phi = sind(phi)
OB = ((E*PF)**2+(E*sin_phi+V_drop)**2)**0.5
```

```
                              // Voltage(V)
E_b = OB
    // Voltage(V)
alpha_phi = atand((E*sin_phi+V_drop)/(E*PF))
                              //    +  (  )
alpha = alpha_phi-phi
                                                    //    (
    )
E_s_b = 2.0*E_b*sind(2.0/2)
                                               //
    Synchronizing voltage(V)
I_s_b = E_s_b/X
    // Synchronizing current(A)
P_s_b = E*I_s_b*cosd((alpha+1.0))
                                    // Synchronizing
    power per phase(W)
P_s_b_total = 3.0*P_s_b
                                                 // Total
    synchronizing power(W)
P_s_b_total_kw = P_s_b_total/1000.0
                                  // Total
    synchronizing power(kW)
T_s_b = P_s_b_total/(2*%pi*n)
                                             //
    Synchronizing torque(N-m)

// Results
disp("PART II - EXAMPLE : 11.2 : SOLUTION :-")
printf("\nCase(a): Synchronizing power for no-load,
    P_s = %.1f kW", P_s_a_total_kw)
printf("\n         Synchronizing torque for no-load,
     T_s = %.f N-m", T_s_a)
printf("\nCase(b): Synchronizing power at full-load,
     P_s = %.1f kW", P_s_b_total_kw)
printf("\n         Synchronizing torque at full-load
    , T_s = %.f N-m \n", T_s_b)
```

```
printf("\nNOTE: ERROR: Calculation mistakes in textbook")
```

Scilab code Exa 18.3 Armature current EMF and PF of the other alternator

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.3 :
// Page number 331-332
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 6600.0                  // Voltage(V)
R = 0.045                   // Resistance(ohm)
X = 0.45                    // Reactance(ohm)
Load = 10000.0*10**3        // Total load(W)
PF = 0.8                    // Lagging power factor
I_a = 437.5                 // Armature current(A)

// Calculations
I = Load/(3**0.5*V*PF)                                  // Load current(A)
I_working = PF*I                                        // Working component of current(A)
I_watless = (1-PF**2)**0.5*I                            // Watless component of current(A)
I_second = (I_a**2+I_watless**2)**0.5                   //
```

```
Load current supplied by second alternator(A)
PF_second = I_a/I_second                                  // Lagging power factor of second alternator
V_ph = V/3**0.5                                           // Terminal voltage per phase(V)
I_R = I_second*R                                          // Voltage drop due to resistance(V)
I_X = I_second*X                                          // Voltage drop due to reactance(V)
sin_phi_second = (1-PF_second**2)**0.5
E = ((V_ph+I_R*PF_second+I_X*sin_phi_second)**2+(I_X*PF_second-I_R*sin_phi_second)**2)**0.5    // EMF of the alternator(V/phase)
E_ll = 3**0.5*E                                           // Line-to-line EMF of the alternator(V)

// Results
disp("PART II - EXAMPLE : 11.3 : SOLUTION :-")
printf("\nArmature current of other alternator = %.1f A", I_second)
printf("\ne.m.f of other alternator = %.f V (line-to-line)", E_ll)
printf("\nPower factor of other alternator = %.3f (lagging)", PF_second)
```

Scilab code Exa 18.4 New value of machine current and PF Power output Current and

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD SHARING OF POWER GENERATING SOURCES
```

```
// EXAMPLE : 11.4 :
// Page number 332-333
clear ; clc ; close ; // Clear the work space and console

// Given data
X = 10.0                  // Reactance(ohm)
I_a = 220.0               // Armature current(A)
PF = 1.0                  // Unity power factor
V = 11000.0               // Phase voltage(V)
emf_raised = 0.2          // EMF rasied by 20%

// Calculations
I_X = I_a*X                          // Reactance drop (V)
E_0 = (V**2+I_X**2)**0.5             // EMF(V)
E_00 = (1+emf_raised)*E_0            // New value of induced emf(V)
U = ((E_00**2-I_X**2)**0.5-V)/X      // Current(A)
I_1 = (I_a**2+U**2)**0.5             // Current(A)
PF_1 = I_a/I_1                       // Lagging power factor
I_X_2 = (E_00**2+V**2)**0.5          // Reactance drop (V)
I_2 = I_X_2/X                        // Current corresponding to this drop(A)
PF_2 = E_00/I_X_2                    // Leading power factor
P_max = V*I_2*PF_2/1000              // Maximum power output(kW)

// Results
disp("PART II - EXAMPLE : 11.4 : SOLUTION :-")
printf("\nNew value of machine current = %.1f A", I_1)
printf("\nNew vaue of power factor, p.f = %.4f (lagging)", PF_1)
```

```
printf("\nPower output at which alternator break
    from synchronism = %.f kW", P_max)
printf("\nCurrent corresponding to maximum load = %.
    f A", I_2)
printf("\nPower factor corresponding to maximum load
     = %.4f (leading) \n", PF_2)
printf("\nNOTE: ERROR: Calculation mistakes in the
    textbook solution")
```

Scilab code Exa 18.5 Phase angle between busbar sections

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD
    SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.5 :
// Page number 333
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 10000.0               // Voltage(V)
rating = 10000.0          // Full load rating(kW)
V_drop_per = 0.2          // Voltage drop of 20% for
    10000 kW

// Calculations
V_drop = V_drop_per*rating                        //
    Voltage drop(V)
sin_theta_2 = (V_drop/2)/V                        // Sin
```

```
    (   /2)
theta_2 = asind(sin_theta_2)                    //
    /2(   )
theta = 2.0*theta_2                             //
    Phase angle between busbar sections ,    (   )

// Results
disp("PART II − EXAMPLE : 11.5 : SOLUTION :−")
printf("\nPhase angle between busbar sections ,     =
    %.2 f    \n", theta)
printf("\nNOTE: ERROR: Calculation mistakes in the
    textbook solution")
```

Scilab code Exa 18.6 `Voltage and Power factor at this latter station`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD
    SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.6 :
// Page number 334
clear ; clc ; close ; // Clear the work space and
    console

// Given data
load_1 = 20000.0          // Total load(kW)
V = 11000.0               // Voltage(V)
PF_1 = 1.0                // Unity power factor
load_2 = 8000.0           // Load supplied(kW)
PF_2 = 0.8                // Lagging power factor
```

```
R = 0.5                     // Resistance(ohm/phase)
X = 0.8                     // Reactance(ohm/phase)

// Calculations
I_1 = load_1*1000/(3**0.5*V*PF_1)                                   // Load current(A)
I_2 = load_2*1000/(3**0.5*V*PF_2)*exp(%i*-acos(PF_2))               // Current supplied by local generators(A)
I_3 = I_1-I_2                                                       // Current through interconnector(A)
angle_I_3 = phasemag(I_3)                                           // Current through interconnector leads reference phasor by angle( )
V_drop = (R+%i*X)*I_3                                               // Voltage drop across interconnector(V)
V_ph = V/3**0.5                                                     // Phase voltage(V)
V_S = V_ph+V_drop                                                   // Sending end voltage(V/phase)
V_S_ll = 3**0.5*V_S                                                 // Sending end voltage(V)
angle_V_S_ll = phasemag(V_S_ll)                                     // Angle of sending end voltage( )
PF_S = cosd(angle_I_3-angle_V_S_ll)                                 // Power factor at sending station

// Results
disp("PART II - EXAMPLE : 11.6 : SOLUTION :-")
```

```
printf("\nVoltage at this latter station = %.f % .2f  V (line-to-line)", abs(V_S_ll),angle_V_S_ll)
printf("\nPower factor at this latter station = %.4f (leading)", PF_S)
```

Scilab code Exa 18.7 Load received Power factor and Phase difference between volta

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.7 :
// Page number 334
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 33000.0                 // Voltage(V)
R = 0.7                     // Resistance(ohm/phase)
X = 3.5                     // Reactance(ohm/phase)
load_1 = 60.0               // Load on generator at station X(MW)
PF_1 = 0.8                  // Lagging power factor
load_2 = 40.0               // Local load taken by consumer(MW)
PF_2 = 0.707                // Lagging power factor

// Calculations
V_ph = V/3**0.5
```

```
    // Phase voltage(V)
I_1 = load_1*10**6/(3**0.5*V*PF_1)*exp(%i*-acos(PF_1
    ))              // Load current on generator at X
    (A)
I_2 = load_2*10**6/(3**0.5*V*PF_2)*exp(%i*-acos(PF_2
    ))              // Current due to local load(A)
I_3 = I_1-I_2
    // Current through interconnector(A)
angle_I_3 = phasemag(I_3)
                                                  //
    Current through interconnector leads reference
    phasor by angle( )
V_drop = (R+%i*X)*I_3
                                                     //
    Voltage drop across interconnector(V)
V_Y = V_ph-V_drop
    // Voltage at Y(V)
angle_V_Y = phasemag(V_Y)
                                                  //
    Angle of voltage at Y( )
phase_diff = angle_I_3-angle_V_Y
                                             // Phase
    difference b/w Y_Y and I_3( )
PF_Y = cosd(phase_diff)
                                                   //
    Power factor of current received by Y
P_Y = 3*abs(V_Y*I_3)*PF_Y/1000.0
                                             // Power
    received by station Y(kW)
phase_XY = abs(angle_V_Y)
                                                  //
    Phase angle b/w voltages of X & Y( )

// Results
disp("PART II - EXAMPLE : 11.7 : SOLUTION :-")
printf("\nLoad received from station X to station Y
```

```
= %.f kW", P_Y)
printf("\nPower factor of load received by Y = %.4f (lagging)", PF_Y)
printf("\nPhase difference between voltage of X & Y = %.2f  (lagging) \n", phase_XY)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 18.8 Percentage increase in voltage and Phase angle difference bet

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.8 :
// Page number 335
clear ; clc ; close ; // Clear the work space and console

// Given data
V_tie = 11000.0          // Tie line Voltage(V)
Z = (3.5+%i*7.0)         // Impedance of tie line(ohm/conductor)
V = 6600.0               // Bus bar voltage(V)
Z_per = (2.5+%i*7.5)     // Percentage impedance on 1000kVA rating
kVA = 2500.0             // Load receieved by other(kVA)

// Calculations
```

```
V_ph = V/3**0.5                              // Phase voltage(V)
I_fl_LV = 100.0*V_tie/V_ph                   // LV side Full load current of each transformer(A)
R_eq = V_ph*real(Z_per)/(100*I_fl_LV)        // Equivalent resistance of transformer(ohm/phase)
X_eq = 3.0*R_eq                              // Equivalent reactance of transformer(ohm/phase)
R_phase = real(Z)*(V/V_tie)**2               // Resistance of line per phase(ohm)
X_phase = imag(Z)*(V/V_tie)**2               // Resistance of line per phase(ohm)
R_total = 2.0*R_eq+R_phase                   // Total resistance per phase(ohm)
X_total = 2.0*X_eq+X_phase                   // Total resistance per phase(ohm)
Z_total = R_total+%i*X_total                 // Total impedance(ohm/phase)
I = kVA*1000/(3**0.5*V)                      // Load current(A)
V_drop = I*Z_total                           // Voltage drop per phase(V)
V_A = V_ph
V_AA = V_A+V_drop                            // Sending end voltage per phase(V)
V_increase = abs(V_AA)-V_A                   // Increase in voltage required(V/phase)
percentage_increase = V_increase/V_A*100     // Percentage increase required(%)
phase_diff = phasemag(V_AA)                  // Angle at which V_A & V_B are displaced( )

// Results
disp("PART II - EXAMPLE : 11.8 : SOLUTION :-")
printf("\nCase(a): Percentage increase in voltage = %.2f percent", percentage_increase)
printf("\nCase(b): Phase angle difference between the two busbar voltages = %.2f \n", phase_diff)
```

```
printf("\nNOTE: ERROR: Several calculation mistakes
    in the textbook")
```

Scilab code Exa 18.9 Station power factors and Phase angle between two busbar volt

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD
   SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.9 :
// Page number 335-336
clear ; clc ; close ; // Clear the work space and
    console

// Given data
X = 2.80              // Combined reactance(ohm/phase
    )
load_1 = 7000.0       // Consumer load at station A(
   kW)
PF_1 = 0.9            // Lagging power factor
V = 11000.0           // Voltage(V)
load_2 = 10000.0      // Load supplied by station B(
   kW)
PF_2 = 0.75           // Lagging power factor

// Calculations
V_ph = V/3**0.5

    // Phase voltage(V)
I_1 = load_1*10**3/(3**0.5*V*PF_1)*exp(%i*-acos(PF_1
```

```
))                    // Current at A due to local load(A)
I_2 = load_2*10**3/(3**0.5*V*PF_2)*exp(%i*-acos(PF_2))                    // Current at B due to local load(A)
IA_X = 0.5*(load_1+load_2)*1000/(3**0.5*V)                              // Current(A)
Y_1 = 220.443/V_ph                                                      // Solved manually referring textbook
X_1 = (1-Y_1**2)**0.5
angle_1 = atand(Y_1/X_1)                                                // Phasor lags by an angle( )
IA_Y = (6849.09119318-V_ph*X_1)/X                                       // Current(A)
Y_X = IA_Y/IA_X
angle_2 = atand(Y_X)                                                    // Angle by which I_A lags behind V_A( )
PF_A = cosd(angle_2)                                                    // Power factor of station A
angle_3 = acosd(PF_2)+angle_1                                           // Angle by which I_2 lags V_A( )
I_22 = load_2*10**3/(3**0.5*V*PF_2)*exp(%i*-angle_3*%pi/180)           // Current(A)
I = 78.7295821622-%i*(IA_Y-177.942225747)                              // Current(A)
I_B = I_22-I                                                            // Current(A)
angle_4 = abs(phasemag(I_B))-angle_1                                    // Angle by which I_B lags behind V_B( )
PF_B = cosd(angle_4)
```

```
    // Power factor of station B

// Results
disp("PART II - EXAMPLE : 11.9 : SOLUTION :-")
printf("\nPower factor of station A = %.4f (lagging)
    ", PF_A)
printf("\nPower factor of station B = %.4f (lagging)
    ", PF_B)
printf("\nPhase angle between two bus bar voltages =
    %.f  (V_B lagging V_A)", angle_1)
```

Scilab code Exa 18.10 Constants of the second feeder

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD
    SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.10 :
// Page number 336
clear ; clc ; close ; // Clear the work space and
    console

// Given data
load_1 = 10000.0        // Total balanced load(kW)
V = 33000.0             // Voltage(V)
PF_1 = 0.8              // Lagging power factor
R = 1.6                 // Resistance of feeder(ohm/
    phase)
X = 2.5                 // Reactance of feeder(ohm/
```

```
phase)
load_2 = 4460.0          // Load delivered by feeder(kW)
PF_2 = 0.72              // Lagging power factor

// Calculations
I = load_1*1000/(3**0.5*V*PF_1)*exp(%i*-acos(PF_1))          // Total line current(A)
I_1 = load_2*1000/(3**0.5*V*PF_2)*exp(%i*-acos(PF_2))      // Line current of first feeder(A)
I_2 = I-I_1                                                  // Line current of first feeder(A)
Z_1 = complex(R,X)                                           // Impedance of first feeder(ohm)
Z_2 = I_1*Z_1/I_2                                            // Impedance of second feeder(ohm)

// Results
disp("PART II - EXAMPLE : 11.10 : SOLUTION :-")
printf("\nImpedance of second feeder, Z_2 = %.2f %.1f   ohm \n", abs(Z_2),phasemag(Z_2))
printf("\nNOTE: ERROR: Changes in the obtained answer from that of textbook is due to wrong values of substitution")
```

Scilab code Exa 18.11 Necessary booster voltages

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.11 :
// Page number 337
clear ; clc ; close ; // Clear the work space and console

// Given data
P = 9.0                       // Load supplied from substation(MW)
V = 33000.0                   // Voltage(V)
PF_1 = 1.0                    // Unity power factor
Z_A = complex(2.0,8.0)        // Impedance of circuit A(ohm)
Z_B = complex(4.0,4.0)        // Impedance of circuit B(ohm)

// Calculations
V_ph = V/3**0.5                                    // Voltage at receiving end per phase(V)
P_A = 1.0/3*P                                      // Power supplied by line A(MW)
P_B = 2.0/3*P                                      // Power supplied by line B(MW)
I_A = P_A*10**6/(3**0.5*V)                         // Current through line A(A)
I_B = P_B*10**6/(3**0.5*V)                         // Current through line B(A)
IA_ZA_drop = I_A*Z_A                               // I_A Z_A drop(V/phase)
IB_ZB_drop = I_B*Z_B                               // I_B Z_B drop(V/phase)
phase_boost = real(IB_ZB_drop)-real(IA_ZA_drop)    // Voltage in phase boost(V/phase)
quad_boost = imag(IB_ZB_drop)-imag(IA_ZA_drop)     // Voltage in quadrature boost(V/phase)
```

```
constant_P = V_ph+IA_ZA_drop                    // Assumed that sending end voltage at P is kept constant(V/phase)

// Results
disp("PART II - EXAMPLE : 11.11 : SOLUTION :-")
printf("\nVoltage in-phase boost = %.2f V/phase", phase_boost)
printf("\nVoltage in quadrature boost = %.f V/phase", quad_boost)
```

Scilab code Exa 18.12 Load on C at two different conditions of load in A and B

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.12 :
// Page number 337
clear ; clc ; close ; // Clear the work space and console

// Given data
cap_A = 15000.0              // Capacity of station A(kW)
cap_B = 10000.0              // Capacity of station B(kW)
cap_C = 2000.0               // Capacity of station C(kW)
speed_reg_A = 2.4/100        // Speed regulation of
```

```
A
speed_reg_B = 3.2/100          // Speed regulation of B
slip_C = 4.5/100               // Full load slip
local_load_B_a = 10000.0       // Local load on station B(kW)
local_load_A_a = 0             // Local load on station A(kW)
local_load_both = 10000.0      // Local load on both station(kW)

// Calculations
// Case(a)
speed_A = speed_reg_A/cap_A                                    // % of speed drop for A
speed_C = slip_C/cap_C                                         // % of speed drop for C
speed_B = speed_reg_B/cap_B                                    // % of speed drop for B
X = local_load_B_a*speed_B/(speed_A+speed_B+speed_C)           // Load on C when local load of B is 10000 kW and A has no load(kW)
// Case(b)
Y = local_load_both*(speed_B-speed_A)/(speed_A+speed_B+speed_C)    // Load on C when both station have local loads of 10000 kW(kW)

// Results
disp("PART II - EXAMPLE : 11.12 : SOLUTION :-")
printf("\nCase(a): Load on C when local load of B is 10000 kW and A has no load, X = %.f kW", X)
printf("\nCase(b): Load on C when both station have local loads of 10000 kW, Y = %.f kW", Y)
```

Scilab code Exa 18.13 Loss in the interconnector as a percentage of power received

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 11: LOAD FREQUENCY CONTROL AND LOAD SHARING OF POWER GENERATING SOURCES

// EXAMPLE : 11.13 :
// Page number 337-338
clear ; clc ; close ; // Clear the work space and console

// Given data
l = 20.0                  // Length of cable(km)
r = 0.248                 // Resistance(ohm/km)
x = 0.50*10**-3           // Inductance(H/m)
V_gen = 6600.0            // Generation voltage(V)
f = 50.0                  // Frequency(Hz)
V = 33000.0               // Transmission voltage(V)
rating = 10.0             // Transformer rating(MVA)
loss_cu = 100.0           // Copper loss at full load(kW)
x_tr = 2.5/100            // Transformer reactance
load = 7.5                // Load to be transmitted(MW)
PF = 0.71                 // Lagging power factor

// Calculations
R = l*r
```

```
// Resistance of the cable(ohm)
I_fl = rating*10**6/(3**0.5*V)                    // Transformer current at full load(A)
R_eq = loss_cu*1000/(3*I_fl**2)                   // Equivalent resistance per phase of transformer(ohm)
R_total_hv = R+2.0*R_eq                           // Total resistance per conductor in terms of hv side(ohm)
X = 2.0*%pi*f*l*x                                 // Reactance of cable per conductor(ohm)
per_X_tr = V/3**0.5*x_tr/I_fl                     // % reactance of transformer(ohm)
X_total_hv = X+2.0*per_X_tr                       // Total reactance per conductor in terms of hv side(ohm)
I = load*10**6/(3**0.5*V*PF)                      // Line current at receiving end(A)
IR = I*R_total_hv                                 // IR drop(V)
IX = I*X_total_hv                                 // IX drop(V)
E_r = V/3**0.5                                    // Phase voltage at station B(V)
cos_phi_r = PF
sin_phi_r = (1-PF**2)**0.5
E_s = ((E_r*cos_phi_r+IR)**2+(E_r*sin_phi_r+IX)**2)**0.5/1000    // Sending end voltage(kV)
E_s_ll = 3**0.5*E_s                               // Sending end line voltage(kV)
```

```
V_booster = 3**0.5*(E_s-E_r/1000)                        // Booster voltage between lines(kV)
tan_phi_s = (E_r*sin_phi_r+IX)/(E_r*cos_phi_r+IR)        // tan _s
phi_s = atand(tan_phi_s)                                 //  _s ( )
cos_phi_s = cosd(phi_s)                                  // cos _s
P_s = 3.0*E_s*I*cos_phi_s                                // Power at sending end(kW)
loss = P_s-load*1000                                     // Loss(kW)
loss_per = loss/(load*1000)*100                          // loss percentage

// Results
disp("PART II - EXAMPLE : 11.13 : SOLUTION :-")
printf("\nLoss in the interconnector as a percentage of power received = %.3f percent", loss_per)
printf("\nRequired voltage of the booster = %.3f kV (in terms of H.V) \n", V_booster)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
printf("\n    kVA rating of booster is not calculated in textbook and here")
```

Chapter 20

WAVE PROPAGATION ON TRANSMISSION LINES

Scilab code Exa 20.4 Reflected and Transmitted wave of Voltage and Current at the

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 13: WAVE PROPAGATION ON TRANSMISSION
   LINES

// EXAMPLE : 13.4 :
// Page number 366
clear ; clc ; close ; // Clear the work space and
   console

// Given data
R_1 = 60.0      // Surge impedance of underground
   cable(ohm)
R_2 = 400.0     // Surge impedance of overhead line(
   ohm)
```

```
e = 100.0       // Maximum value of surge(kV)

// Calculations
i = e*1000/R_1                  // Current(A)
k = (R_2-R_1)/(R_2+R_1)
e_ref = k*e                     // Reflected voltage(kV)
e_trans = e+e_ref               // Transmitted voltage (kV)
e_trans_alt = (1+k)*e           // Transmitted voltage (kV). Alternative method
i_ref = -k*i                    // Reflected current(A)
i_trans = e_trans*1000/R_2      // Transmitted current (A)
i_trans_alt = (1-k)*i           // Transmitted current (A). Alternative method

// Results
disp("PART II - EXAMPLE : 13.4 : SOLUTION :-")
printf("\nReflected voltage at the junction = %.f kV", e_ref)
printf("\nTransmitted voltage at the junction = %.f kV", e_trans)
printf("\nReflected current at the junction = %.f A", i_ref)
printf("\nTransmitted current at the junction = %.f A\n", i_trans)
printf("\nNOTE: ERROR: Calculation mistake in textbook in finding Reflected current")
```

Scilab code Exa 20.5 First and Second voltages impressed on C

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
```

```
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 13: WAVE PROPAGATION ON TRANSMISSION
   LINES

// EXAMPLE : 13.5 :
// Page number 366
clear ; clc ; close ; // Clear the work space and
   console

// Given data
R_A = 500.0      // Surge impedance of line A(ohm)
R_B = 70.0       // Surge impedance of line B(ohm)
R_C = 600.0      // Surge impedance of line C(ohm)
e = 20.0         // Rectangular voltage wave(kV)

// Calculations
E_2 = e*(1+((R_B-R_A)/(R_B+R_A)))        //
   Transmitted wave(kV)
E_4 = E_2*(1+((R_C-R_B)/(R_C+R_B)))      // First
   voltage impressed on C(kV)
E_3 = E_2*(R_C-R_B)/(R_C+R_B)            // Reflected
    wave(kV)
E_5 = E_3*(R_A-R_B)/(R_A+R_B)            // Reflected
    wave(kV)
E_6 = E_5*(1+((R_C-R_B)/(R_C+R_B)))      //
   Transmitted wave(kV)
second = E_4+E_6                         // Second
   voltage impressed on C(kV)

// Results
disp("PART II - EXAMPLE : 13.5 : SOLUTION :-")
printf("\nFirst voltage impressed on C = %.1f kV",
   E_4)
printf("\nSecond voltage impressed on C = %.1f kV",
   second)
```

Scilab code Exa 20.6 `Voltage and Current in the cable and Open wire lines`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 13: WAVE PROPAGATION ON TRANSMISSION LINES

// EXAMPLE : 13.6 :
// Page number 367
clear ; clc ; close ; // Clear the work space and console

// Given data
Z = 100.0           // Surge impedance of cable(ohm)
Z_1 = 600.0         // Surge impedance of open wire(ohm)
Z_2 = 1000.0        // Surge impedance of open wire(ohm)
e = 2.0             // Steep fronted voltage(kV)

// Calculations
Z_t = Z_1*Z_2/(Z_1+Z_2)        // Resultant surge impedance(ohm)
E = e*(1+((Z_t-Z)/(Z_t+Z)))    // Transmitted voltage(kV)
I_1 = E*1000/Z_1               // Current(A)
I_2 = E*1000/Z_2               // Current(A)
E_ref = e*(Z_t-Z)/(Z_t+Z)      // Reflected voltage(kV)
I_ref = -E_ref*1000/Z          // Reflected current(A
```

```
)

// Results
disp("PART II - EXAMPLE : 13.6 : SOLUTION :-")
printf("\nVoltage in the cable = %.3f kV", E)
printf("\nCurrent in the cable, I_1 = %.2f A", I_1)
printf("\nCurrent in the cable, I_2 = %.3f A", I_2)
printf("\nVoltage in the open-wire lines i.e
    Reflected voltage = %.3f kV", E_ref)
printf("\nCurrent in the open-wire lines i.e
    Reflected current = %.2f A", I_ref)
```

Chapter 21

LIGHTNING AND PROTECTION AGAINST OVERVOLTAGES DUE TO LIGHTNING

Scilab code Exa 21.1 Ratio of voltages appearing at the end of a line when line is

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 14: LIGHTNING AND PROTECTION AGAINST
   OVERVOLTAGES DUE TO LIGHTNING

// EXAMPLE : 14.1 :
// Page number 382
clear ; clc ; close ; // Clear the work space and
   console

// Given data
```

```
RI_072 = 72000.0      // Charactersistic of lightning
    arrester
Z_c = 500.0           // Surge impedance(ohm)
V = 500.0             // Surge voltage(kV)

// Calculations
// Case(a)
V_a = 2.0*V             // Voltage at the end of line
    at open-circuit(kV)
ratio_a = V_a/V         // Ratio of voltage when line
    in open-circuited
// Case(b)
I = V*1000/Z_c          // Surge current(A)
R = RI_072/(I)**0.72    // Resistance of LA(ohm)
ratio_b = R/Z_c         // Ratio of voltage when line
    is terminated by arrester

// Results
disp("PART II - EXAMPLE : 14.1 : SOLUTION :-")
printf("\nCase(a): Ratio of voltages appearing at
   the end of a line when line is open-circuited = %
   .f", ratio_a)
printf("\nCase(b): Ratio of voltages appearing at
   the end of a line when line is terminated by
   arrester = %.f", ratio_b)
```

Scilab code Exa 21.2 Choosing suitable arrester rating

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 14: LIGHTNING AND PROTECTION AGAINST
```

OVERVOLTAGES DUE TO LIGHTNING

```

// EXAMPLE : 14.2 :
// Page number 383
clear ; clc ; close ; // Clear the work space and
    console

// Given data
rating = 5000.0        // Rating of transformer(kVA)
V_hv = 66.0            // HV voltage(kV)
V_lv = 11.0            // LV voltage(kV)
V = 66.0               // System voltage(kV)
fluctuation = 0.1      // Voltage fluctuations
BIL = 350.0            // BIL for 66kV(kV)
dynamic_ov = 1.3       // Dynamic over-voltage = 1.3*
    system operating voltage
V_power_freq = 1.5     // Power frequency breakdown
    voltage of arrester = 1.5*arrester rating(kV)
lower_limit = 0.05     // Margin of lower limit of
    arrester rating

// Calculation & Result
disp("PART II - EXAMPLE : 14.2 : SOLUTION :-")
V_rating = V*(1+fluctuation)*0.8*(1+lower_limit)
            // Voltage rating of arrester(kV)
if(round(V_rating)==51) then
    V_rating_choosen = 50.0
                                    // Arrester
        rating choosen(kV)
    V_discharge = 176.0
                                         //
        Discharge voltage for 50kV arrester(kV)
    protective_margin = BIL-V_discharge
                        // Protective margin
        available(kV)
    V_power_frequency_bd = V_rating_choosen*
        V_power_freq  // Power frequency breakdown
        voltage(kV)
```

```
    Over_voltage_dynamic = dynamic_ov*V/3**0.5                // Dynamic overvoltage(kV)
    if(V_power_frequency_bd>Over_voltage_dynamic) then
        printf("\nFirst arrester with rating 50 kV (rms) & discharge voltage 176 kV chosen is suitable")
    end
elseif(round(V_rating)==61) then
    V_rating_choosen = 60.0                                   // Arrester rating choosen(kV)
    V_discharge = 220.0                                       // Discharge voltage for 50kV arrester(kV)
    protective_margin = BIL-V_discharge                       // Protective margin available(kV)
    V_power_frequency_bd = V_rating_choosen*V_power_freq      // Power frequency breakdown voltage(kV)
    Over_voltage_dynamic = dynamic_ov*V/3**0.5                // Dynamic overvoltage(kV)
    if(V_power_frequency_bd>Over_voltage_dynamic)
        printf("\nSecond arrester with rating 60 kV (rms) & discharge voltage 220 kV chosen is suitable")
    end
else(round(V_rating)==74) then
    V_rating_choosen = 73.0                                   // Arrester rating choosen(kV)
    V_discharge = 264.0                                       // Discharge voltage for 50kV arrester(kV)
    protective_margin = BIL-V_discharge                       // Protective margin available(kV)
```

```
    V_power_frequency_bd = V_rating_choosen*V_power_freq  // Power frequency breakdown voltage(kV)
    Over_voltage_dynamic = dynamic_ov*V/3**0.5                // Dynamic overvoltage(kV)
    if(V_power_frequency_bd>Over_voltage_dynamic) then
      printf("\nThird arrester with rating 73 kV (rms) & discharge voltage 264 kV chosen is suitable")
    end
end
```

Chapter 22

INSULATION COORDINATION

Scilab code Exa 22.1 `Highest voltage to which the transformer is subjected`

```
// A Textbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 15: INSULATION CO-ORDINATION

// EXAMPLE : 15.1 :
// Page number 398-399
clear ; clc ; close ; // Clear the work space and console

// Given data
L = 30.0           // Height of arrester located(m)
BIL = 650.0        // BIL(kV)
de_dt = 1000.0     // Rate of rising surge wave front (kV/ -sec)
V = 132.0          // Transformer voltage at HV side(
```

```
kV)
E_a = 400.0        // Discharge voltage of arrester(kV)
v = 3.0*10**8      // Velocity of surge propagation(m/sec)

// Calculations
E_t = E_a+(2.0*de_dt*L/300)   // Highest voltage the transformer is subjected(kV)

// Results
disp("PART II - EXAMPLE : 15.1 : SOLUTION :-")
printf("\nHighest voltage to which the transformer is subjected , E_t = %.f kV", E_t)
```

Scilab code Exa 22.2 `Rating of LA and Location with respect to transformer`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 15: INSULATION CO-ORDINATION

// EXAMPLE : 15.2 :
// Page number 399
clear ; clc ; close ; // Clear the work space and console

// Given data
V_hv = 132.0       // Voltage at the HV side of transformer(kV)
V_lv = 33.0        // Voltage at the LV side of transformer(kV)
```

```
V = 860.0          // Insulator allowable voltage(kV)
Z = 400.0          // Line surge impedance(ohm)
BIL = 550.0        // BIL(kV)

// Calculations
V_rating_LA = V_hv*1.1*0.8                        // Voltage rating of LA(kV)
E_a = 351.0                                       // Discharge voltage at 5 kA(kV)
I_disc = (2*V-E_a)*1000/Z                         // Discharge current(A)
L_1 = 37.7                                        // Separation distance in current b/w arrester tap and power transformer tap(m)
dist = 11.0                                       // Lead length from tap point to ground level(m)
de_dt = 500.0                                     // Maximum rate of rise of surge(kV/ -sec)
Inductance = 1.2                                  // Inductance( H /metre)
di_dt = 5000.0                                    // di/dt(A/ -sec)
lead_drop = Inductance*dist*di_dt/1000            // Drop in the lead(kV)
E_d = E_a+lead_drop                               // (kV)
V_tr_terminal = E_d+2*de_dt*L_1/300               // Voltage at transformer terminals(kV)
E_t = BIL/1.2                                     // Highest voltage the transformer is subjected(kV)
L = (E_t-E_a)/(2*de_dt)*300                       // Distance at which lightning arrester located from transformer(m)
L_lead = (E_t-E_a*1.1)/(2*de_dt)*300              // Distance at which lightning arrester located from transformer taken 10% lead drop(m)

// Results
disp("PART II - EXAMPLE : 15.2 : SOLUTION :-")
```

```
printf("\nRating of L.A = %.1f kV", V_rating_LA)
printf("\nLocation of L.A, L = %.f m", L)
printf("\nLocation of L.A if 10 percent lead drop is
    considered, L = %.1f m", L_lead)
printf("\nMaximum distance at which a ligtning
   arrester is usually connected from transformer is
   %.f-%.f m", L-2,L+3)
```

Chapter 23

POWER SYSTEM GROUNDING

Scilab code Exa 23.1 Inductance and Rating of arc suppression coil

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 16: POWER SYSTEM GROUNDING

// EXAMPLE : 16.1 :
// Page number 409
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 132.0*10**3          // Voltage(V)
n = 3.0                  // Number of phase
f = 50.0                 // Frequency(Hz)
l = 50.0                 // Line length(km)
C = 0.0157*10**-6        // Capacitance to earth(F/km)
```

```
// Calculations
L = 1/(n*(2*%pi*f)**2*C*l)          // Inductance(H)
X_L = 2*%pi*f*L                     // Reactance(ohm)
I_F = V/(3**0.5*X_L)                // Current(A)
rating = I_F*V/(3**0.5*1000)        // Rating of arc
    suppression coil(kVA)

// Results
disp("PART II - EXAMPLE : 16.1 : SOLUTION :-")
printf("\nInductance, L = %.1f Henry", L)
printf("\nRating of arc suppression coil = %.f kVA \
    n", rating)
printf("\nNOTE: Changes in the obtained answer from
    that of textbook is due to more approximation in
    the textbook")
```

Chapter 24

ELECTRIC POWER SUPPLY SYSTEMS

Scilab code Exa 24.1 Weight of copper required for a three phase transmission syst

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 17: ELECTRIC POWER SUPPLY SYSTEMS

// EXAMPLE : 17.1 :
// Page number 422-423
clear ; clc ; close ; // Clear the work space and console

// Given data
no_phase = 3.0           // Number of phases in ac transmission system
V = 380.0*10**3          // Voltage b/w lines(V)
load = 100.0             // Load(MW)
PF = 0.9                 // Power factor
```

```
l = 150.0                  // Line length(km)
n = 0.92                   // Efficiency
r = 0.045                  // Resistance(ohm/km/sq.cm)
w_cu_1 = 0.01              // Weight of 1 cm^3 copper(kg)

// Calculations
// Case(i)
P_loss = (1-n)*load                    // Power loss in the line(MW)
I_L = load*10**6/(3**0.5*V*PF)         // Line current(A)
loss_cu = P_loss/no_phase*10**6        // I^2*R loss per conductor(W)
R = loss_cu/I_L**2                     // Resistance per conductor(ohm)
R_km = R/l                             // Resistance per conductor per km(ohm)
area = r/R_km                          // Conductor area(Sq.cm)
volume = area*100.0                    // Volume of copper per km run(cm^3)
W_cu_km = volume*w_cu_1                // Weight of copper per km run(kg)
W_cu = no_phase*l*1000*W_cu_km         // Weight of copper for 3 conductors of 150 km(kg)
// Case(ii)
W_cu_dc = 1.0/2*PF**2*W_cu             // Weight of copper conductor in dc(kg)

// Results
disp("PART II - EXAMPLE : 17.1 : SOLUTION :-")
printf("\nWeight of copper required for a three-phase transmission system = %.f kg", W_cu)
printf("\nWeight of copper required for the d-c transmission system = %.f kg \n", W_cu_dc)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision")
```

Scilab code Exa 24.2 Percentage increase in power transmitted

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 17: ELECTRIC POWER SUPPLY SYSTEMS

// EXAMPLE : 17.2 :
// Page number 423
clear ; clc ; close ; // Clear the work space and
    console

// Given data
P_1 = 1.0     // Assume P1 to be 1

// Calculations
P_2 = (3.0*2)**0.5                // 3-phase power
    transmitted in terms of P_1
inc_per = (P_2-P_1)/P_1*100    // Increase in power
    transmitted(%)

// Results
disp("PART II - EXAMPLE : 17.2 : SOLUTION :-")
printf("\nPercentage increase in power transmitted =
    %.f percent", inc_per)
```

Scilab code Exa 24.3 Percentage additional balanced load

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 17: ELECTRIC POWER SUPPLY SYSTEMS

// EXAMPLE : 17.3 :
// Page number 424
clear ; clc ; close ; // Clear the work space and
    console

// Given data
PF = 0.95     // Lagging power factor

// Calculations
P_1 = 1.0                                    //
    Power in terms of V*I_1
P_2 = 2.0*PF**2                              //
    Power in terms of V*I_1
P_additional_percentage = (P_2-P_1)/P_1*100   //
    Percentage additional power transmitted in a 3-
    phase 3-wire system

// Results
disp("PART II - EXAMPLE : 17.3 : SOLUTION :-")
printf("\nPercentage additional power transmitted in
     a 3-phase 3-wire system = %.f percent",
    P_additional_percentage)
```

Scilab code Exa 24.4 Amount of copper required for 3 phase 4 wire system with that

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
```

```
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 17: ELECTRIC POWER SUPPLY SYSTEMS

// EXAMPLE : 17.4 :
// Page number 424-425
clear ; clc ; close ; // Clear the work space and
   console

// Given data
n = 3.0          // 3-phase 4 wire ac system

// Calculations
a2_a1 = 1.0/6               // Ratio of cross-sectional
    area of 2 wire dc to 3-phase 4-wire system
ratio_cu = 3.5/2*a2_a1      // Copper for 3 phase 4
   wire system to copper for 2 wire dc system

// Results
disp("PART II - EXAMPLE : 17.4 : SOLUTION :-")
printf("\nCopper for 3-phase 4-wire system/Copper
   for 2-wire dc system = %.3f : 1", ratio_cu)
```

Scilab code Exa 24.5 Weight of copper required and Reduction of weight of copper p

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 17: ELECTRIC POWER SUPPLY SYSTEMS

```

```
// EXAMPLE : 17.5 :
// Page number 425
clear ; clc ; close ; // Clear the work space and console

// Given data
L = 60.0                // Line length(km)
P = 5.0                 // Load(MW)
PF = 0.8                // Lagging power factor
V = 33.0*10**3          // Voltage(V)
n = 0.85                // Transmission efficiency
rho = 1.73*10**-8       // Specific resistance of copper (ohm-mt)
density = 8900.0        // Density(kg/mt^3)

// Calculations
I = P*10**6/(3**0.5*V*PF)                    // Line current(A)
line_loss = (1-n)*P*1000/n                   // Line loss (kW)
line_loss_phase = line_loss/3.0              // Line loss /phase(kW)
R = line_loss_phase*1000/I**2                // Resistance/phase(ohm)
a = rho*L*1000/R                             // Area of cross section of conductor(m^2)
volume = 3.0*a*L*1000                        // Volume of copper(m^3)
W_cu = volume*density                        // Weight of copper in 3-phase system(kg)
I_1 = P*10**6/V                              // Current in single phase system(A)
R_1 = line_loss*1000/(2*I_1**2)              // Resistance in single phase system(ohm)
a_1 = rho*L*1000/R_1                         // Area of cross section of conductor in single phase system (m^2)
volume_1 = 2.0*a_1*L*1000                    // Volume of
```

```
    copper(m^3)
W_cu_1 = volume_1*density                    // Weight of
    copper in 1-phase system(kg)
reduction_cu = (W_cu-W_cu_1)/W_cu*100       // Reduction
    in copper(%)

// Results
disp("PART II - EXAMPLE : 17.5 : SOLUTION :-")
printf("\nWeight of copper required for 3-phase 2-
   wire system = %.2e kg", W_cu)
printf("\nReduction of weight of copper possible = %
   .1f percent \n", reduction_cu)
printf("\nNOTE: ERROR: Calculation mistakes in the
   textbook solution")
```

Scilab code Exa 24.6 Economical cross section of a 3 core distributor cable

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 17: ELECTRIC POWER SUPPLY SYSTEMS

// EXAMPLE : 17.6 :
// Page number 427-428
clear ; clc ; close ; // Clear the work space and
   console

// Given data
L = 250.0                   // Cable length(m)
P = 80.0*10**3              // Load(W)
V = 400.0                   // Voltage(V)
PF = 0.8                    // Lagging power factor
```

```
time = 4000.0                  // Time of operation(hours/annum)
a = poly(0,'a')                // Area of each conductor(Sq.cm)
cost_instal = 15.0*a+25        // Cost of cable including installation(Rs/m)
interest_per = 0.1             // Interest & depreciation
cost_waste_per = 0.1           // Cost of energy wasted(Rs/unit)
r = 0.173                      // Resistance per km of 1 cm^2(ohm)

// Calculations
I = P/(3**0.5*V*PF)                                       // Line current(A)
energy_waste = 3.0*I**2*r/a*L*10**-3*time*10**-3         // Energy wasted per annum(kWh)
cost_energy_waste = cost_waste_per*energy_waste          // Annual cost of energy wasted as losses(Rs)
capitaL_cost_cable = cost_instal*L                       // Capital cost of cable(Rs)
annual_cost_cable = capitaL_cost_cable*cost_waste_per    // Annual cost on cable(Rs)
area = (1081.25/375)**0.5                                // Area = a(Sq.cm). Simplified and taken final answer

// Results
disp("PART II - EXAMPLE : 17.6 : SOLUTION :-")
printf("\nEconomical cross-section of a 3-core distributor cable, a = %.1f cm^2", area)
```

Scilab code Exa 24.7 Most economical cross section

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 17: ELECTRIC POWER SUPPLY SYSTEMS

// EXAMPLE : 17.7 :
// Page number 428
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 110.0*10**3                // Voltage(V)
l_1 = 24.0*10**6               // Load(MW)
t_1 = 6.0                      // Time(hours)
l_2 = 8.0*10**6                // Load(MW)
t_2 = 6.0                      // Time(hours)
l_3 = 4.0*10**6                // Load(MW)
t_3 = 12.0                     // Time(hours)
PF = 0.8                       // Lagging power factor
a = poly(0,'a')                // Cross-section of each conductor(Sq.cm)
cost_line = 12000.0+8000*a     // Cost of line including erection(Rs/km)
R = 0.19/a                     // Resistance per km of each conductor(ohm)
cost_energy = 8.0/100          // Energy cost(Rs/unit)
interest_per = 0.1             // Interest & depreciation. Assumption

// Calculations
annual_charge = interest_per*cost_line      // Total annual charge(Rs)
I_1 = l_1/(3**0.5*V*PF)                     // Line
```

```
current for load 1(A)
I_2 = l_2/(3**0.5*V*PF)                          // Line current for load 2(A)
I_3 = l_3/(3**0.5*V*PF)                          // Line current for load 3(A)
I_2_t = I_1**2*t_1+I_2**2*t_2+I_3**2*t_3         // I^2*t
annual_energy = 3.0*R*365/1000*I_2_t             // Annual energy consumption on account of losses(kWh)
cost_waste = annual_energy*cost_energy           // Cost of energy wasted per annum(Rs)
area = (2888.62809917355/800.0)**0.5             // Economical cross−section = a(Sq.cm). Simplified and taken final answer

// Results
disp("PART II − EXAMPLE : 17.7 : SOLUTION :−")
printf("\nMost economical cross−section, a = %.2f cm^2", area)
```

Scilab code Exa 24.8 `Most economical current density for the transmission line`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 17: ELECTRIC POWER SUPPLY SYSTEMS

// EXAMPLE : 17.8 :
// Page number 428−429
clear ; clc ; close ; // Clear the work space and console

// Given data
```

```
cost_km_cu = 2800.0           // Cost per km for each copper conductor of sq.cm(Rs)
LF_I = 80.0/100               // Load factor of load current
LF_loss = 65.0/100            // Load factor of losses
interest_per = 10.0/100       // Rate of interest and depreciation
cost_energy = 5.0/100         // Cost of energy(Rs/kWh)
rho = 1.78*10**-8             // Resistivity(ohm-m)

// Calculations
P_2 = cost_km_cu*interest_per                        // Cost in terms of L(Rs)
time_year = 365.0*24                                 // Total hours in a year
P_3 = cost_energy*rho*10**4*time_year*LF_loss        // Cost in terms of I^2 & L(Rs)
delta = (P_2/P_3)**0.5                               // Economical current density for the transmission line(A/sq.cm)

// Results
disp("PART II - EXAMPLE : 17.8 : SOLUTION :-")
printf("\nMost economical current density for the transmission line,     = %.f A/sq.cm", delta)
```

Scilab code Exa 24.9 Most economical cross section of the conductor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
```

```
// CHAPTER 17: ELECTRIC POWER SUPPLY SYSTEMS

// EXAMPLE : 17.9 :
// Page number 429
clear ; clc ; close ; // Clear the work space and console

// Given data
MD = 1000.0                   // Maximum demand(kW)
energy_cons = 5.0*10**6       // Annual energy consumption(kWh)
PF = 0.85                     // Power factor
capital_cost = 80000.0        // Capital cost of cable (Rs/km)
cost_energy = 5.0/100         // Energy cost(Rs/kWh)
interest_per = 10.0/100       // Rate of interest and depreciation
r_specific = 1.72*10**-6      // Specific resistance of copper(ohm/cubic.cm)
V = 11.0                      // Voltage(kV)

// Calculations
I = MD/(3**0.5*V*PF)                                            // Line current corresponding to maximum demand(A)
hours_year = 365.0*24                                           // Total hours in a year
LF = energy_cons/(MD*hours_year)                                // Load factor
loss_LF = 0.25*LF+0.75*LF**2                                    // Loss load factor
P_2 = capital_cost*interest_per                                 // Cost in terms of L(Rs)
P_3 = 3.0*I**2*r_specific*10**4*hours_year*loss_LF*cost_energy  // Cost in terms of I^2 & L(Rs)
```

```
a = (P_3/P_2)**0.5                                         //
    Most economical cross-section of conductor(sq.cm)

// Results
disp("PART II - EXAMPLE : 17.9 : SOLUTION :-")
printf("\nMost economical cross-section of the
    conductor, a = %.2f cm^2 \n", a)
printf("\nNOTE: ERROR: Calculation mistake in the
    textbook solution")
```

Chapter 25

POWER DISTRIBUTION SYSTEMS

Scilab code Exa 25.1 Potential of O and Current leaving each supply point

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 18: POWER DISTRIBUTION SYSTEMS

// EXAMPLE : 18.1 :
// Page number 437
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V_A = 225.0     // Potential at point A(V)
R_A = 5.0       // Resistance of line A(ohm)
V_B = 210.0     // Potential at point B(V)
R_B = 1.0       // Resistance of line B(ohm)
V_C = 230.0     // Potential at point C(V)
```

```
R_C = 1.0       // Resistance of line C(ohm)
V_D = 230.0     // Potential at point D(V)
R_D = 2.0       // Resistance of line D(ohm)
V_E = 240.0     // Potential at point E(V)
R_E = 2.0       // Resistance of line E(ohm)

// Calculations
V_O = ((V_A/R_A)+(V_B/R_B)+(V_C/R_C)+(V_D/R_D)+(V_E/R_E))/((1/R_A)+(1/R_B)+(1/R_C)+(1/R_D)+(1/R_E))    // Potential at point O(V)
I_A = (V_A-V_O)/R_A          // Current leaving supply point A(A)
I_B = (V_B-V_O)/R_B          // Current leaving supply point B(A)
I_C = (V_C-V_O)/R_C          // Current leaving supply point C(A)
I_D = (V_D-V_O)/R_D          // Current leaving supply point D(A)
I_E = (V_E-V_O)/R_E          // Current leaving supply point E(A)

// Results
disp("PART II - EXAMPLE : 18.1 : SOLUTION :-")
printf("\nPotential of point O, V_O = %.f V", V_O)
printf("\nCurrent leaving supply point A, I_A = %.f A", I_A)
printf("\nCurrent leaving supply point B, I_B = %.f A", I_B)
printf("\nCurrent leaving supply point C, I_C = %.f A", I_C)
printf("\nCurrent leaving supply point D, I_D = %.2f A", I_D)
printf("\nCurrent leaving supply point E, I_E = %.2f A", I_E)
```

Scilab code Exa 25.2 Point of minimum potential along the track and Currents suppl

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 18: POWER DISTRIBUTION SYSTEMS

// EXAMPLE : 18.2 :
// Page number 437-438
clear ; clc ; close ; // Clear the work space and console

// Given data
I = 600.0          // Constant current drawn(A)
D = 8.0            // Distance b/w two sub-stations(km)
V_A = 575.0        // Potential at point A(V)
V_B = 590.0        // Potential at point B(V)
R = 0.04           // Track resistance(ohm/km)

// Calculations
x = poly(0,'x')                                      // x(km)
I_A = ((-V_B+R*I*D+V_A)-(R*I)*x)/(D*R)               // Simplifying
V_P = V_A-I_A*R*x                                    // Potential at P in terms of x(V)
dVP_dx = derivat(V_P)                                // dV_P/dx
x_sol = roots(dVP_dx)                                // Value of x(km)
I_A_1 = ((-V_B+R*I*D+V_A)-(R*I)*x_sol)/(D*R)         // Current drawn from end A(A)
I_B = I-I_A_1                                        // Current drawn from end B(A)
```

```

// Results
disp("PART II - EXAMPLE : 18.2 : SOLUTION :-")
printf("\nPoint of minimum potential along the track, x = %.2f km", x_sol)
printf("\nCurrent supplied by station A, I_A = %.f A", I_A_1)
printf("\nCurrent supplied by station B, I_B = %.f A\n", I_B)
printf("\nNOTE: ERROR: Calculation mistake in the textbook solution")
```

Scilab code Exa 25.3 `Position of lowest run lamp and its Voltage`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 18: POWER DISTRIBUTION SYSTEMS

// EXAMPLE : 18.3 :
// Page number 438-439
clear ; clc ; close ; // Clear the work space and console

// Given data
l = 400.0           // Length of cable(m)
i = 1.0             // Load(A/m)
I_1 = 120.0         // Current at 40m from end A(A)
l_1 = 40.0          // Distance from end A(A)
I_2 = 72.0          // Current at 72m from end A(A)
l_2 = 120.0         // Distance from end A(A)
I_3 = 48.0          // Current at 200m from end A(A)
```

```
l_3 = 200.0          // Distance from end A(A)
I_4 = 120.0          // Current at 320m from end A(A)
l_4 = 320.0          // Distance from end A(A)
r = 0.15             // Cable resistance(ohm/km)
V_A = 250.0          // Voltage at end A(A)
V_B = 250.0          // Voltage at end A(A)

// Calculations
I = poly(0,"I")                                                     // Current from end A(A)
A_A1 = l_1*r*(I-(1.0/2)*i*l_1)                                      // Drop over length(V)
I_d_1 = 40.0                                                        // Distributed tapped off current(A)
I_A1_A2 = I-l_1-l_2                                                 // Current fed in over length(A)
A1_A2 = (l_2-l_1)*r*(I_A1_A2-(1.0/2)*i*(l_2-l_1))                   // Drop over length(V)
I_d_2 = 80.0                                                        // Distributed tapped off current(A)
I_A2_A3 = I_A1_A2-(I_2+I_d_2)                                       // Current fed in over length(A)
A2_A3 = (l_3-l_2)*r*(I_A2_A3-(1.0/2)*i*(l_3-l_2))                   // Drop over length(V)
I_d_3 = 80.0                                                        // Distributed tapped off current(A)
I_A3_A4 = I_A2_A3-(I_3+I_d_3)                                       // Current fed in over length(A)
A3_A4 = (l_4-l_3)*r*(I_A3_A4-(1.0/2)*i*(l_4-l_3))                   // Drop over length(V)
```

```
I_d_4 = 120.0
    // Distributed tapped off current(A)
I_A4_B = I_A3_A4-(I_4+I_d_4)                              //
    Current fed in over length(A)
A4_B = (l-l_4)*r*(I_A4_B-(1.0/2)*i*(l-l_4))
                            // Drop over length(V)
V_drop = A_A1+A1_A2+A2_A3+A3_A4+A4_B
                                   // Total voltage
    drop in terms of I
I = roots(V_drop)
    // Current(A)
I_total = 760.0
    // Total load current(A)
I_B = I_total-I
    // Current from B(A)
A_A3 = 2.0*r/1000*(l_1*(I-20)+(l_2-l_1)*(I-200)+(l_3
    -l_2)*(I-352))    // Potential drop over length
    A_A3(V)
V_A3 = V_A-A_A3
    // Voltage at the lowest run lamp(V)

// Results
disp("PART II - EXAMPLE : 18.3 : SOLUTION :-")
printf("\nPosition of lowest-run lamp, A_3 = %.f m",
    l_3)
printf("\nVoltage at the lowest-run lamp = %.1f V",
    V_A3)
```

Scilab code Exa 25.4 Point of minimum potential and its Potential

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 18: POWER DISTRIBUTION SYSTEMS

// EXAMPLE : 18.4 :
// Page number 439
clear ; clc ; close ; // Clear the work space and console

// Given data
l = 450.0           // Length of wire(m)
V_A = 250.0         // Voltage at end A(V)
V_B = 250.0         // Voltage at end A(V)
r = 0.05            // Conductor resistance(ohm/km)
i = 1.5             // Load(A/m)
I_C = 20.0          // Current at C(A)
l_C = 60.0          // Distance to C from A(m)
I_D = 40.0          // Current at D(A)
l_D = 100.0         // Distance to D from A(m)
l_E = 200.0         // Distance to E from A(m)

// Calculations
x = poly(0,"x")                                   // Current to point D from end A(A)
AD = (I_C+x)*r*l_C+x*r*(l_D-l_C)                  // Drop in length AD
BD = (i*r*V_A**2/2)+(I_D-x)*r*(450-l_D)           // Drop in length BD
x_sol = roots(AD-BD)                              // Current(A)
I_F = x_sol-I_D                                   // Current supplied to load from end A(A)
l_F = l_E+(I_F/i)                                 // Point of minimum potential at F from A(m)
```

```
V_F = V_B-(375.0-I_F)*(250-(l_F-200))*r/1000      //
   Potential at F from end B(V)

// Results
disp("PART II - EXAMPLE : 18.4 : SOLUTION :-")
printf("\nPoint of minimum potential occurs at F
   from A = %.2f metres", l_F)
printf("\nPotential at point F = %.2f V", V_F)
```

Scilab code Exa 25.6 Ratio of weight of copper with and without interconnector

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 18: POWER DISTRIBUTION SYSTEMS

// EXAMPLE : 18.6 :
// Page number 440-441
clear ; clc ; close ; // Clear the work space and
   console

// Given data
l_AB = 100.0          // Length between A & B(m)
l_BC = 150.0          // Length between B & C(m)
l_CD = 200.0          // Length between C & D(m)
l_AD = 350.0          // Length between A & D(m)
l_AE = 200.0          // Length between A & E(m)
l_ED = 250.0          // Length between E & D(m)
I_B = 10.0            // Current at B(A)
I_C = 20.0            // Current at C(A)
I_D = 50.0            // Current at D(A)
I_E = 39.0            // Current at E(A)
```

```

// Calculations
x = poly(0,"x")                                                 // Current in section AB(A)
ABCDEA = x*l_AB+(x-I_B)*l_BC+(x-I_B-I_C)*l_CD+(x-I_B-I_C-I_D)*l_ED+(x-I_B-I_C-I_D-I_E)*l_AE    // KVL around loop ABCDEA
x_sol = roots(ABCDEA)                                           // Current in section AB(A)
V_AD = x_sol*l_AB+(x_sol-I_B)*l_BC+(x_sol-I_B-I_C)*l_CD              // Voltage drop from A to D in terms of   /a_1(V)
R_AD = (l_AB+l_BC+l_CD)*(l_AE+l_ED)/(l_AB+l_BC+l_CD+l_AE+l_ED)       // Resistance of n/w across terminals AD in terms of   /a
I_AD = V_AD/(R_AD+l_AD)                                         // Current in interconnector AD(A)
V_A_D = I_AD*l_AD                                               // Voltage drop between A & D in terms of   /a_2
a2_a1 = V_A_D/V_AD
length_with = (l_AB+l_BC+l_CD+l_AE+l_ED+l_AD)                        // Length of conductor with interconnector(m)
length_without = (l_AB+l_BC+l_CD+l_AE+l_ED)                          // Length of conductor without interconnector(m)
volume_with = a2_a1*length_with/length_without                       // Weight of copper with interconnector

// Results
disp("PART II - EXAMPLE : 18.6 : SOLUTION :-")
printf("\nRatio of weight of copper with & without interconnector = %.3f : 1 (or) 1 : %.2f",
```

```
volume_with,1/volume_with)
```

Scilab code Exa 25.7 Potential difference at each load point

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 18: POWER DISTRIBUTION SYSTEMS

// EXAMPLE : 18.7 :
// Page number 441-442
clear ; clc ; close ; // Clear the work space and console

// Given data
r_out = 0.05              // Resistance of each outer per 100 metre length(ohm)
r_neutral = 0.10          // Resistance of each neutral per 100 metre length(ohm)
V_A = 200.0               // Potential at point A(V)
V_B = 200.0               // Potential at point B(V)
l_AC = 100.0              // Length between A & C(m)
l_CD = 150.0              // Length between C & D(m)
l_DB = 200.0              // Length between D & B(m)
l_AF = 200.0              // Length between A & F(m)
l_FE = 100.0              // Length between F & E(m)
l_EB = 150.0              // Length between E & B(m)
I_C = 20.0                // Current at point C(A)
I_D = 30.0                // Current at point D(A)
I_F = 60.0                // Current at point F(A)
I_E = 40.0                // Current at point E(A)

```

```
// Calculations
x = poly(0,"x")                                          // Current in positive outer alone(A)
equ_1 = r_out*(l_DB*(I_D-x))-r_out*(l_AC*(I_C+x)+l_CD*x)
x_sol = roots(equ_1)                                     // Current in positive outer alone(A)
y = poly(0,"y")                                          // Current in negative outer alone(A)
equ_2 = r_out*((I_E-y)*l_FE+(I_E+I_F-y)*l_AF)-r_out*(l_EB*y)
y_sol = roots(equ_2)                                     // Current in negative outer alone(A)
I_pos_out = I_C+x_sol                                    // Current entering positive outer(A)
I_neg_out = I_E+I_F-y_sol                                // Current returning via negative outer(A)
I_middle = I_neg_out-I_pos_out                           // Current in the middle wire towards G(A)
r_CD = r_out*l_CD/100.0                                  // Resistance between C & D(ohm)
r_D = r_out*l_DB/100.0                                   // Resistance between D & B(ohm)
r_IH = r_neutral*l_FE*0.5/100.0                          // Resistance between I & H(ohm)
r_IJ = r_neutral*l_FE*0.5/100.0                          // Resistance between I & J(ohm)
```

```
r_GH = r_neutral*l_AF*0.5/100.0                                     // Resistance between G & H(ohm)
r_AF = r_out*l_AF/100.0                                                  // Resistance between A & F(ohm)
I_CD = x_sol                                                   // Current flowing into D from C(A)
I_out_D = I_D-x_sol                                                      // Current flowing into D from outer side(A)
I_GH = I_C+I_middle                                                      // Current flowing into H from G(A)
I_IH = I_F-I_GH                                                // Current flowing into H from I(A)
I_BJ = I_E-(I_D-I_IH)                                                    // Current flowing into J from B(A)
I_FE = y_sol-I_E                                               // Current flowing into E from F(A)
I_IJ = I_D-I_IH                                                // Current flowing into J from I(A)
V_C = V_A-(I_pos_out*r_out-I_middle*r_neutral)               // Potential at load point C(A)
V_D = V_C-(I_CD*r_CD+I_IH*r_IH-I_GH*r_GH)                    // Potential at load point D(A)
V_F = V_A-(I_middle*r_neutral+I_GH*r_neutral+I_neg_out*r_AF)    // Potential at load point F(A)
V_E = V_F-(-I_IH*r_IH+I_IJ*r_IJ-I_FE*r_out)                  // Potential at load point
```

```
E(A)

// Results
disp("PART II - EXAMPLE : 18.7 : SOLUTION :-")
printf("\nPotential difference at load point C = %.3f V", V_C)
printf("\nPotential difference at load point D = %.3f V", V_D)
printf("\nPotential difference at load point E = %.3f V", V_E)
printf("\nPotential difference at load point F = %.3f V", V_F)
```

Scilab code Exa 25.8 `Load on the main generators and On each balancer machine`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 18: POWER DISTRIBUTION SYSTEMS

// EXAMPLE : 18.8 :
// Page number 442-443
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 440.0              // Voltage between outer(V)
I_pos = 210.0          // Ligting load current on positive side(A)
I_neg = 337.0          // Ligting load current on negative side(A)
I_power = 400.0        // Power load current(A)
```

```
P_loss = 1.5           // Loss in each balancer machine(kW)

// Calculations
P = I_power*V/1000.0                             // Power(kW)
load_pos = I_pos*V*0.5/1000.0                    // Load on positive side(kW)
load_neg = I_neg*V*0.5/1000.0                    // Load on negative side(kW)
loss_total = 2*P_loss                            // Total loss on rotary balancer set(kW)
load_main = P+load_pos+load_neg+loss_total       // Load on main machine(kW)
I = load_main*1000/V                             // Current(A)
I_M = I-610.0                                    // Current through balancer machine(A)
I_G = 127.0-I_M                                  // Current through generator(A)
output_G = I_G*V*0.5/1000.0                      // Output of generator(kW)
input_M = I_M*V*0.5/1000.0                       // Input to balancer machine(kW)

// Results
disp("PART II - EXAMPLE : 18.8 : SOLUTION :-")
printf("\nLoad on the main machine = %.2f kW", load_main)
printf("\nOutput of generator = %.2f kW", output_G)
printf("\nInput to balancer machine = %.2f kW", input_M)
```

Scilab code Exa 25.9 `Currents in various sections and Voltage at load point C`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART II : TRANSMISSION AND DISTRIBUTION
// CHAPTER 18: POWER DISTRIBUTION SYSTEMS

// EXAMPLE : 18.9 :
// Page number 444
clear ; clc ; close ; // Clear the work space and console

// Given data
V_a = 11.0*10**3             // Line voltage at A(V)
Z_AB = complex(1.0,0.8)      // Impedance between A & B(ohm)
Z_AC = complex(3.0,2.0)      // Impedance between A & C(ohm)
Z_BD = complex(3.0,4.0)      // Impedance between B & D(ohm)
Z_CD = complex(1.0,0.7)      // Impedance between C & D(ohm)
I_B = 60.0                   // Current at B(A)
I_C = 30.0                   // Current at C(A)
I_D = 50.0                   // Current at D(A)
pf_B = 0.8                   // Power factor at B
pf_C = 0.9                   // Power factor at C
pf_D = 0.707                 // Power factor at D

// Calculations
sin_phi_B = (1-pf_B**2)**0.5
I_B1 = I_B*(pf_B-%i*sin_phi_B)      // Load current(A)
sin_phi_C = (1-pf_C**2)**0.5
I_C1 = I_C*(pf_C-%i*sin_phi_C)      // Load current(A)
sin_phi_D = (1-pf_D**2)**0.5
```

```
I_D1 = I_D*(pf_D-%i*sin_phi_D)       // Load current(A)
V_A = V_a/3**0.5                     // Phase voltage at A(V)
I_AC = I_C1                          // Current in section AC when C & D is removed(A)
I_BD = I_D1                          // Current in section BD when C & D is removed(A)
I_AB = I_B1+I_D1                     // Current in section AB when C & D is removed(A)
V_AC_drop = I_AC*Z_AC                // Voltage drop at section AC(V)
V_AB_drop = I_AB*Z_AB                // Voltage drop at section AB(V)
V_BD_drop = I_BD*Z_BD                // Voltage drop at section BD(V)
V_drop_D = V_BD_drop+V_AB_drop       // Total drop upto D(V)
pd_CD = V_drop_D-V_AC_drop           // Potential difference between C & D(V)
Z_C_D = Z_AB+Z_BD+Z_AC               // Impedance of network looking from terminal C & D(ohm)
I_CD = pd_CD/(Z_C_D+Z_CD)            // Current flowing in section CD(A)
I_AC = I_CD+I_C1                     // Current flowing in section AC(A)
I_BD = I_D1-I_CD                     // Current flowing in section BD(A)
I_AB = I_BD+I_B1                     // Current flowing in section AB(A)
V_drop_AC = I_AC*Z_AC                // Drop caused by current flowing in section AC(V/phase)
V_drop_AC_line = V_drop_AC*3**0.5    // Drop caused by current flowing in section AC(V)
V_C = V_a-V_drop_AC_line             // Voltage at C(V)

// Results
```

```
disp("PART II - EXAMPLE : 18.9 : SOLUTION :-")
printf("\nCurrent in section CD, I_CD = (%.2f%.2fj)
    A", real(I_CD),imag(I_CD))
printf("\nCurrent in section AC, I_AC = (%.2f%.2fj)
    A", real(I_AC),imag(I_AC))
printf("\nCurrent in section BD, I_BD = (%.2f%.2fj)
    A", real(I_BD),imag(I_BD))
printf("\nCurrent in section AB, I_AB = (%.2f%.2fj)
    A", real(I_AB),imag(I_AB))
printf("\nVoltage at load point C = %.2f % .2f    kV
    ", abs(V_C)/1000,phasemag(V_C))
```

Chapter 27

SYMMETRICAL SHORT CIRCUIT CAPACITY CALCULATIONS

Scilab code Exa 27.1 Per unit current

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY
   CALCULATIONS

// EXAMPLE : 1.1 :
// Page number 466-467
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 500.0                       // Generator voltage(V)
rating = 10.0                   // Rating of the
```

```
generator(kVA)
n_up = 1.0/2                  // Turns ratio of step-up transformer
Z_line = complex(1.0,2.0)     // Transmission line impedance(ohm)
n_down = 10.0/1               // Turns ratio of step-down transformer
load = complex(2.0,4.0)       // Load(ohm)

// Calculations
V_base_gen = V                                        // Base voltage(V)
kVA_base_gen = rating                                 // Base rating(kVA)
I_base_gen = kVA_base_gen*1000/V_base_gen             // Base current(A)
Z_base_gen = V_base_gen/I_base_gen                    // Base impedance(ohm)
V_base_line = V_base_gen/n_up                         // Voltage base of the transmission line(V)
kVA_base_line = rating                                // Base rating of transmission line(kVA)
I_base_line = kVA_base_line*1000/V_base_line          // Base current of transmission line(A)
Z_base_line = V_base_line/I_base_line                 // Base impedance of transmission line(ohm)
Z_line_1 = Z_line/Z_base_line                         // Impedance of transmission line(p.u)
V_base_load = V_base_line/n_down                      // Base voltage at the load(V)
kVA_base_load = rating                                // Base rating of load(kVA)
I_base_load = kVA_base_load*1000/V_base_load          // Base current of load(A)
Z_base_load = V_base_load/I_base_load                 // Base impedance of load(ohm)
Z_load = load/Z_base_load                             // Load impedance(p.u)
```

```
Z_total = Z_line_1+Z_load                              // Total impedance(p.u)
I = 1.0/Z_total                                        // Current(p.u)

// Results
disp("PART III - EXAMPLE : 1.1 : SOLUTION :-")
printf("\nCurrent, I = %.3f %.2f p.u", abs(I), phasemag(I))
```

Scilab code Exa 27.2 kVA at a short circuit fault between phases at the HV termina

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY CALCULATIONS

// EXAMPLE : 1.2 :
// Page number 467-468
clear ; clc ; close ; // Clear the work space and console

// Given data
kV = 33.0                     // Transmission line operating voltage(kV)
R = 5.0                       // Transmission line resistance(ohm)
X = 20.0                      // Transmission line reactance(ohm)
kVA_tr = 5000.0               // Rating of step-up transformer(kVA)
```

```
X_tr = 6.0                    // Reactance of transformer(%)
kVA_A = 10000.0               // Rating of alternator A(kVA)
X_A = 10.0                    // Reactance of alternator A(%)
kVA_B = 5000.0                // Rating of alternator B(kVA)
X_B = 7.5                     // Reactance of alternator B(%)

// Calculations
kVA_base = kVA_A                                          // Base rating(kVA)
X_gen_A = X_A*kVA_base/kVA_A                  // Reactance of generator A(%)
X_gen_B = X_B*kVA_base/kVA_B                  // Reactance of generator B(%)
X_trans = X_tr*kVA_base/kVA_tr              // Reactance of transformer(%)
X_per = kVA_base*X/(10*kV**2)                  // X(%)
R_per = kVA_base*R/(10*kV**2)                  // R(%)
Z_F1 = (X_gen_A*X_gen_B/(X_gen_A+X_gen_B))+X_trans      // Impedance upto fault(%)
kVA_F1 = kVA_base*(100/Z_F1)                  // Short-circuit kVA fed into the fault(kVA)
R_per_F2 = R_per                                          // R(%)
X_per_F2 = X_per+Z_F1                                // X(%)
Z_F2 = (R_per_F2**2+X_per_F2**2)**0.5
```

```
                    // Total impedance upto F2(%)
kVA_F2 = kVA_base*(100/Z_F2)
                    // Short-circuit kVA fed into the fault at F2(kVA)

// Results
disp("PART III - EXAMPLE : 1.2 : SOLUTION :-")
printf("\nCase(a): kVA at a short-circuit fault between phases at the HV terminal of transformers = %.f kVA", kVA_F1)
printf("\nCase(b): kVA at a short-circuit fault between phases at load end of transmission line = %.f kVA \n", kVA_F2)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here & approximation in textbook")
```

Scilab code Exa 27.3 Transient short circuit current and Sustained short circuit c

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY CALCULATIONS

// EXAMPLE : 1.3 :
// Page number 468-469
clear ; clc ; close ; // Clear the work space and console

// Given data
kVA_a = 40000.0          // Capacity of transmission
```

```
line(kVA)
x_a = 10.0                    // Reactance of transmission line(%)
kVA_b = 20000.0               // Capacity of transmission line(kVA)
x_b = 5.0                     // Reactance of transmission line(%)
kVA_c = 50000.0               // Capacity of transmission line(kVA)
x_c = 20.0                    // Reactance of transmission line(%)
kVA_d = 30000.0               // Capacity of transmission line(kVA)
x_d = 15.0                    // Reactance of transmission line(%)
kVA_e = 10000.0               // Capacity of transmission line(kVA)
x_e = 6.0                     // Reactance of transmission line(%)
kVA_T1 = 150000.0             // Capacity of transformer(kVA)
x_T1 = 10.0                   // Reactance of transformer(%)
kVA_T2 = 50000.0              // Capacity of transformer(kVA)
x_T2 = 8.0                    // Reactance of transformer(%)
kVA_T3 = 20000.0              // Capacity of transformer(kVA)
x_T3 = 5.0                    // Reactance of transformer(%)
kVA_GA = 150000.0             // Capacity of generator(kVA)
x_sA = 90.0                   // Synchronous reactance of generator(%)
x_tA = 30.0                   // Transient reactance of generator(%)
kVA_GB = 50000.0              // Capacity of generator(
```

```
    kVA)
x_sB = 50.0                    // Synchronous reactance of
    generator(%)
x_tB = 17.5                    // Transient reactance of
    generator(%)
V = 33.0                       // Feeder voltage(kV)

// Calculations
kVA_base = 200000.0
                                          // Base rating(
    kVA)
X_a = kVA_base/kVA_a*x_a
                                    // Reactance(%)
X_b = kVA_base/kVA_b*x_b
                                    // Reactance(%)
X_c = kVA_base/kVA_c*x_c
                                    // Reactance(%)
X_d = kVA_base/kVA_d*x_d
                                    // Reactance(%)
X_e = kVA_base/kVA_e*x_e
                                    // Reactance(%)
X_T1 = kVA_base/kVA_T1*x_T1
                               // Reactance(%)
X_T2 = kVA_base/kVA_T2*x_T2
                               // Reactance(%)
X_T3 = kVA_base/kVA_T3*x_T3
                               // Reactance(%)
X_sA = kVA_base/kVA_GA*x_sA
                               // Synchronous reactance
    (%)
X_tA = kVA_base/kVA_GA*x_tA
                               // Transient reactance(%
    )
X_sB = kVA_base/kVA_GB*x_sB
                               // Synchronous reactance
    (%)
X_tB = kVA_base/kVA_GB*x_tB
                               // Transient reactance(%
```

```
)
X_eq_ab = X_a+X_b                                       // Equivalent reactance of transmission lines a & b(%)
X_eq_abc = X_eq_ab*X_c/(X_eq_ab+X_c)                    // Equivalent reactance of transmission line c with series combination of a & b(%)
X_CF = (X_eq_abc+X_sA)*X_d/(X_eq_abc+X_sA+X_d)          // Total reactance b/w sub-station C & F(%)
// Case(i)
X_tr_genA = kVA_base/kVA_GA*x_tA                        // Reactance in transient state of generator A(%)
X_T1_tr = kVA_base/kVA_T1*x_T1                          // Reactance in transient state of transformer T1(%)
X_CF_tr = X_CF                                          // Total reactance in transient state b/w sub-station C & F(%)
X_eq_tAF = X_tr_genA+X_T1_tr+X_CF_tr                    // Equivalent transient reactance from generator A to substation F(%)
X_tr_genB = kVA_base/kVA_GB*x_tB                        // Reactance in transient state of generator B(%)
X_T2_tr = kVA_base/kVA_T2*x_T2                          // Reactance in transient state of transformer T2(%)
X_eq_tBF = X_tr_genB+X_T2_tr                            // Equivalent transient reactance from generator B to substation F(%)
X_eq_tF = X_eq_tAF*X_eq_tBF/(X_eq_tAF+X_eq_tBF)         // Equivalent transient reactance upto substation F(%)
X_eq_tfault = X_eq_tF+X_T3                              // Equivalent transient
```

```
    reactance upto fault point(%)
kVA_t_sc = kVA_base/X_eq_tfault*100                     // Transient short circuit kVA(kVA)
I_t_sc = kVA_t_sc/(3**0.5*V)                            // Transient short circuit rms current(A)
I_t_sc_peak = 2**0.5*I_t_sc                             // Peak value of transient short circuit current(A)
// Case(ii)
X_S_genA = kVA_base/kVA_GA*x_sA                         // Reactance in steady state of generator A(%)
X_eq_SAF = X_S_genA+X_T1+X_CF                           // Equivalent steady state reactance from generator A to substation F(%)
X_eq_SBF = X_sB+X_T2                                    // Equivalent steady state reactance from generator B to substation F(%)
X_eq_SF = X_eq_SAF*X_eq_SBF/(X_eq_SAF+X_eq_SBF)         // Equivalent steady state reactance upto substation F(%)
X_eq_Sfault = X_eq_SF+X_T3                              // Equivalent steady state reactance upto fault point(%)
kVA_S_sc = kVA_base/X_eq_Sfault*100                     // Steady state short circuit kVA(kVA)
I_S_sc = kVA_S_sc/(3**0.5*V)                            // Sustained short circuit rms current(A)
I_S_sc_peak = 2**0.5*I_S_sc                             // Peak value of sustained short circuit current(A)

// Results
```

```
disp("PART III - EXAMPLE : 1.3 : SOLUTION :-")
printf("\nCase(i) : Transient short circuit current
    at X = %.f A (peak value)", I_t_sc_peak)
printf("\nCase(ii): Sustained short circuit current
    at X = %.f A (peak value) \n", I_S_sc_peak)
printf("\nNOTE: Changes in the obtained answer from
    that of textbook is due to more precision here")
```

Scilab code Exa 27.4 Current in the short circuit

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY
    CALCULATIONS

// EXAMPLE : 1.4 :
// Page number 469-470
clear ; clc ; close ; // Clear the work space and
    console

// Given data
kVA_gen = 21000.0         // Generator rating(kVA)
kV_gen = 13.8             // Voltage rating of
    generator(kV)
X_tr_gen = 30.0           // Transient reactance of
    generator(%)
kVA_trans = 7000.0        // Transformer rating(kVA)
kV_trans_lv = 13.8        // LV voltage rating of
    transformer(kV)
kV_trans_hv = 66.0        // HV voltage rating of
    transformer(kV)
```

```
X_trans = 8.4              // Reactance of transformer(%)
l = 50.0                   // Tie line length(miles)
x = 0.848                  // Reactance of tie line(ohm/mile)
l_fault = 20.0             // Location of fault from station A(miles)

// Calculations
kVA_base = kVA_gen                                      // Base rating(kVA)
X_A = X_tr_gen                                          // Reactance of generator A(%)
X_B = X_tr_gen                                          // Reactance of generator B(%)
X_T1 = 3.0*X_trans                                      // Reactance of transformer T1(%)
X_T2 = 3.0*X_trans                                      // Reactance of transformer T2(%)
X_1 = kVA_base/(10*kV_trans_hv**2)*x*l_fault            // Reactance(%)
X_2 = X_1*(l-l_fault)/l_fault                           // Reactance(%)
X_AF = X_A+X_T1+X_1                                     // Resultant reactance A to F(%)
X_BF = X_B+X_T2+X_2                                     // Resultant reactance B to F(%)
X_eq_fault = X_AF*X_BF/(X_AF+X_BF)                      // Equivalent reactance upto fault(%)
kVA_SC = kVA_base/X_eq_fault*100                        // Short circuit kVA((kVA)
I_SC = kVA_SC/(3**0.5*kV_trans_hv)                      // Short circuit current(A)

// Results
disp("PART III - EXAMPLE : 1.4 : SOLUTION :-")
printf("\nShort circuit current = %.f A \n", I_SC)
printf("\nNOTE: Changes in the obtained answer from
```

```
that of textbook is due to more precision here")
```

Scilab code Exa 27.5 Per unit values of the single line diagram

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY CALCULATIONS

// EXAMPLE : 1.5 :
// Page number 470-471
clear ; clc ; close ; // Clear the work space and console

// Given data
MVA_G1 = 100.0                // Generator rating(MVA)
X_G1 = 30.0                   // Reactance of generator(%)
MVA_G2 = 150.0                // Generator rating(MVA)
X_G2 = 20.0                   // Reactance of generator(%)
MVA_G3 = 200.0                // Generator rating(MVA)
X_G3 = 15.0                   // Reactance of generator(%)
MVA_T1 = 150.0                // Transformer rating(MVA)
X_T1 = 10.0                   // Reactance of transformer(%)
MVA_T2 = 175.0                // Transformer rating(MVA)
X_T2 = 8.0                    // Reactance of transformer(%)
MVA_T3 = 200.0                // Transformer rating(MVA)
```

```
X_T3 = 6.0                 // Reactance of transformer(%)
MVA_T4 = 100.0             // Transformer rating(MVA)
X_T4 = 5.0                 // Reactance of transformer(%)
MVA_T5 = 150.0             // Transformer rating(MVA)
X_T5 = 5.0                 // Reactance of transformer(%)
Z_L1 = complex(0.5,1.0)    // Line impedance(ohm/km)
L1 = 100.0                 // Line length(km)
Z_L2 = complex(0.4,1.2)    // Line impedance(ohm/km)
L2 = 50.0                  // Line length(km)
Z_L3 = complex(0.4,1.2)    // Line impedance(ohm/km)
L3 = 50.0                  // Line length(km)
Z_L4 = complex(0.3,1.0)    // Line impedance(ohm/km)
L4 = 60.0                  // Line length(km)
kV_L1 = 220.0              // Voltage towards line(kV)
kV_L2 = 220.0              // Voltage towards line(kV)
kV_L3 = 132.0              // Voltage towards line(kV)
kV_L4 = 132.0              // Voltage towards line(kV)

// Calculations
MVA_base = 200.0                          // Base rating(MVA)
X_d_G1 = (MVA_base/MVA_G1)*(X_G1/100)     // Reactance of generator(p.u)
X_d_G2 = (MVA_base/MVA_G2)*(X_G2/100)     // Reactance of generator(p.u)
X_d_G3 = (MVA_base/MVA_G3)*(X_G3/100)     // Reactance of generator(p.u)
X_T_1 = (MVA_base/MVA_T1)*(X_T1/100)      // Reactance of transformer(p.u)
X_T_2 = (MVA_base/MVA_T2)*(X_T2/100)      // Reactance of transformer(p.u)
```

```
X_T_3 = (MVA_base/MVA_T3)*(X_T3/100)      // Reactance of transformer(p.u)
X_T_4 = (MVA_base/MVA_T4)*(X_T4/100)      // Reactance of transformer(p.u)
X_T_5 = (MVA_base/MVA_T5)*(X_T5/100)      // Reactance of transformer(p.u)
Z_L1_base = kV_L1**2/MVA_base             // L1 base impedance(ohm)
Z_L_1 = Z_L1*L1/Z_L1_base                 // Line impedance(p.u)
Z_L2_base = kV_L2**2/MVA_base             // L2 base impedance(ohm)
Z_L_2 = Z_L2*L2/Z_L2_base                 // Line impedance(p.u)
Z_L3_base = kV_L3**2/MVA_base             // L3 base impedance(ohm)
Z_L_3 = Z_L3*L3/Z_L3_base                 // Line impedance(p.u)
Z_L4_base = kV_L4**2/MVA_base             // L4 base impedance(ohm)
Z_L_4 = Z_L4*L4/Z_L4_base                 // Line impedance(p.u)

// Results
disp("PART III - EXAMPLE : 1.5 : SOLUTION :-")
printf("\np.u values of the single line diagram are as below")
printf("\nGenerators p.u reactances :")
printf("\n X_d_G1 = %.1f p.u", X_d_G1)
printf("\n X_d_G2 = %.3f p.u", X_d_G2)
printf("\n X_d_G3 = %.2f p.u", X_d_G3)
printf("\nTransformers p.u reactances :")
printf("\n X_T1 = %.3f p.u", X_T_1)
printf("\n X_T2 = %.4f p.u", X_T_2)
printf("\n X_T3 = %.2f p.u", X_T_3)
printf("\n X_T4 = %.1f p.u", X_T_4)
printf("\n X_T5 = %.3f p.u", X_T_5)
printf("\nLines p.u impedances :")
```

```
printf("\n Z_L1 = (%.3f + %.3fj) p.u", real(Z_L_1),imag(Z_L_1))
printf("\n Z_L2 = (%.3f + %.3fj) p.u", real(Z_L_2),imag(Z_L_2))
printf("\n Z_L3 = (%.3f + %.3fj) p.u", real(Z_L_3),imag(Z_L_3))
printf("\n Z_L4 = (%.3f + %.3fj) p.u \n", real(Z_L_4),imag(Z_L_4))
printf("\nNOTE: ERROR: (1). Reactance of T2 is 8 percent & not 1 percent as mentioned in the textbook problem statement")
printf("\n             (2). Several calculation mistakes in the textbook")
```

Scilab code Exa 27.6 Actual fault current using per unit method

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY CALCULATIONS

// EXAMPLE : 1.6 :
// Page number 471
clear ; clc ; close ; // Clear the work space and console

// Given data
kVA_gen = 21000.0        // Generator rating(kVA)
kV_gen = 13.8            // Voltage rating of generator(kV)
X_tr_gen = 30.0          // Transient reactance of
```

```
generator(%)
kVA_trans = 7000.0        // Transformer rating(kVA)
kV_trans_lv = 13.8        // LV voltage rating of transformer(kV)
kV_trans_hv = 66.0        // HV voltage rating of transformer(kV)
X_trans = 8.4             // Reactance of transformer(%)
l = 50.0                  // Tie line length(miles)
x = 0.848                 // Reactance of tie line(ohm/mile)
l_fault = 20.0            // Location of fault from station A(miles)

// Calculations
kVA_base = kVA_gen        // Base rating(kVA)
kV_base_lv = kV_trans_lv  // Base voltage on L.V side(kV)
kV_base_hv = kV_trans_hv  // Base voltage on H.V side(kV)
Z_gen_pu = %i*X_tr_gen/100        // Impedance of generator(p.u)
Z_trans_pu = %i*X_trans*3/100     // Impedance of transformer(p.u)
Z_F_left = %i*x*l_fault*kVA_base/(kV_base_hv**2*1000)   // Impedance of line to left of fault F(p.u)
Z_F_right = Z_F_left*(l-l_fault)/l_fault     // Impedance of line to right of fault(p.u)
Z_AF = Z_gen_pu+Z_trans_pu+Z_F_left         // Impedance(p.u)
```

```
Z_BF = Z_gen_pu+Z_trans_pu+Z_F_right
                        // Impedance(p.u)
Z_eq = Z_AF*Z_BF/(Z_AF+Z_BF)
                                // Equivalent impedance
    (p.u)
I_F = 1.0/abs(Z_eq)
                                          // Fault
    current(p.u)
I_base = kVA_base/(3**0.5*kV_base_hv)
                       // Base current(A)
I_F_actual = I_F*I_base
                                      // Actual fault
    current(A)

// Results
disp("PART III - EXAMPLE : 1.6 : SOLUTION :-")
printf("\nActual fault current = %.f A \n",
    I_F_actual)
printf("\nNOTE: Changes in the obtained answer from
    that of textbook is due to more precision here")
```

Scilab code Exa 27.7 Sub transient fault current

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY
    CALCULATIONS

// EXAMPLE : 1.7 :
// Page number 471-472
clear ; clc ; close ; // Clear the work space and
```

```
console

// Given data
MVA_G1 = 50.0          // Generator rating(MVA)
kV_G1 = 15.0           // Voltage rating of generator(kV)
X_G1 = 0.2             // Reactance of generator(p.u)
MVA_G2 = 25.0          // Generator rating(MVA)
kV_G2 = 15.0           // Voltage rating of generator(kV)
X_G2 = 0.2             // Reactance of generator(p.u)
kV_T = 66.0            // Voltage rating of transformer(kV)
X_T = 0.1              // Reactance of transformer(p.u)
kV_fault = 66.0        // Voltage at fault occurence(kV)
kv_base = 69.0         // Base voltage(kV)
MVA_base = 100.0       // Base MVA

// Calculations
X_d_G1 = X_G1*MVA_base/MVA_G1                // Sub-transient reactance referred to 100 MVA(p.u)
E_G1 = kV_fault/kv_base                      // Voltage(p.u)
X_d_G2 = X_G2*MVA_base/MVA_G2                // Sub-transient reactance referred to 100 MVA(p.u)
E_G2 = kV_fault/kv_base                      // Voltage(p.u)
X_net = X_d_G1*X_d_G2/(X_d_G1+X_d_G2)        // Net sub-transient reactance(p.u)
E_g = (E_G1+E_G2)/2                          // Net voltage(p.u). NOTE: Not sure how this comes
I_fault = E_g/(%i*(X_net+X_T))               // Sub-transient fault current(p.u)

// Results
disp("PART III - EXAMPLE : 1.7 : SOLUTION :-")
```

```
printf("\nSub−transient fault current = %.3 fj p.u \n
    ", imag(I_fault))
printf("\nNOTE: Changes in the obtained answer from
    that of textbook is due to more precision here")
```

Scilab code Exa 27.8 Voltage behind the respective reactances

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY
    CALCULATIONS

// EXAMPLE : 1.8 :
// Page number 472
clear ; clc ; close ; // Clear the work space and
    console

// Given data
X_d_st = 0.2        // Sub−transient reactance(p.u)
X_d_t = 0.4         // Transient reactance(p.u)
X_d = 1.0           // Direct axis reactance(p.u)
I_pu = 1.0          // Load current(p.u)
PF = 0.80           // Lagging power factor

// Calculations
V = 1.0                      // Terminal voltage(p.u)
sin_phi = (1-PF**2)**0.5
I = I_pu*(PF-%i*sin_phi)     // Load current(p.u)
E_st = V+%i*I*X_d_st         // Voltage behind sub−
    transient reactance(p.u)
E_t = V+%i*I*X_d_t           // Voltage behind
```

```
    transient reactance(p.u)
E = V+%i*I*X_d                   // Voltage behind direct
     axis reactance(p.u)

// Results
disp("PART III - EXAMPLE : 1.8 : SOLUTION :-")
printf("\nVoltage behind sub-transient reactance = %
    .2f %.2f p.u", abs(E_st),phasemag(E_st))
printf("\nVoltage behind transient reactance = %.2
    f %.2f p.u", abs(E_t),phasemag(E_t))
printf("\nVoltage behind direct axis reactance, E =
    %.2f %.2f p.u", abs(E),phasemag(E))
```

Scilab code Exa 27.9 Initial symmetrical rms current in the hv side and lv side

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY
    CALCULATIONS

// EXAMPLE : 1.9 :
// Page number 472
clear ; clc ; close ; // Clear the work space and
    console

// Given data
kVA_G = 7500.0             // Generator rating(kVA)
kV_G = 6.9                 // Voltage rating of
    generator(kV)
X_d_st = 9.0/100           // Sub-transient reactance of
     generator
```

```
X_d_t = 15.0/100        // Transient reactance of
    generator
X_d = 100.0             // Synchronous reactance of
    generator(%)
kVA_T = 7500.0          // Transformer rating(kVA)
kV_T_delta = 6.9        // Voltage rating of
    transformer delta side(kV)
kV_T_wye = 115.0        // Voltage rating of
    transformer wye side(kV)
X = 10.0/100            // Transformer reactance

// Calculations
I_base_ht = kVA_T/(3**0.5*kV_T_wye)      // Base
    current at ht side(A)
I_base_lt = kVA_T/(3**0.5*kV_T_delta)    // Base
    current at lt side(A)
I_f_st = 1.0/(%i*(X_d_st+X))             // Sub-
    transient current after fault(p.u)
I_f_ht = abs(I_f_st)*I_base_ht           // Initial
    fault current in h.t side(A)
I_f_lt = abs(I_f_st)*I_base_lt           // Initial
    fault current in l.t side(A)

// Results
disp("PART III - EXAMPLE : 1.9 : SOLUTION :-")
printf("\nInitial symmetrical rms current in the h.v
     side = %.f A", I_f_ht)
printf("\nInitial symmetrical rms current in the l.v
     side = %.f A \n", I_f_lt)
printf("\nNOTE: Changes in the obtained answer from
    that of textbook is due to more precision here")
```

Scilab code Exa 27.10 Initial symmetrical rms current at the generator terminal

```
// A Texbook on POWER SYSTEM ENGINEERING
```

```
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY CALCULATIONS

// EXAMPLE : 1.10 :
// Page number 472
clear ; clc ; close ; // Clear the work space and console

// Given data
kVA_alt = 625.0          // Alternator rating(kVA)
V_alt = 480.0            // Voltage rating of alternator(V)
load = 500.0             // Load(kW)
V_load = 480.0           // Load voltage(V)
X_st = 8.0/100           // Sub-transient reactance

// Calculations
kVA_base = 625.0                // Base kVA
V_base = 480.0                  // Base voltage(V)
I_load = load/kVA_base          // Load cuurent(A)
V = 1.0                         // Terminal voltage(p.u)
E_st = V+%i*I_load*X_st         // Sub-transient voltage (p.u)
I_st = E_st/(%i*X_st)           // Sub-transient current (p.u)

// Results
disp("PART III - EXAMPLE : 1.10 : SOLUTION :-")
printf("\nInitial symmetrical rms current at the generator terminal = (%.1f%.1fj) p.u", real(I_st),imag(I_st))
```

Scilab code Exa 27.11 Sub transient current in the fault in generator and Motor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY CALCULATIONS

// EXAMPLE : 1.11 :
// Page number 472-473
clear ; clc ; close ; // Clear the work space and console

// Given data
X_d_st_G = 0.15      // Sub-transient reactance of generator(p.u)
X_d_st_M = 0.45      // Sub-transient reactance of motor(p.u)
X = 0.10             // Leakage reactance of transformer(p.u)
V = 0.9              // Terminal voltage of the generator(p.u)
I_G = 1.0            // Output current of the generator(p.u)
PF = 0.8             // Power factor of the load

// Calculations
sin_phi = (1-PF**2)**0.5
I = I_G*(PF+%i*sin_phi)              // Load current(p.u)
E_st_G = V+%i*I*X_d_st_G             // Sub-transient
```

```
    voltage of the generator(p.u)
E_st_M = V-%i*I*X_d_st_M           // Sub-transient
    voltage of the motor(p.u)
I_st_g = E_st_G/(%i*(X_d_st_G+X))  // Sub-transient
    current in the generator at fault(p.u)
I_st_m = E_st_M/(%i*(X_d_st_M-X))  // Sub-transient
    current in the motor at fault(p.u)

// Results
disp("PART III - EXAMPLE : 1.11 : SOLUTION :-")
printf("\nCase(a): Sub-transient current in the
    fault in generator = %.3f % .3f  p.u", abs(
    I_st_g),phasemag(I_st_g))
printf("\nCase(b): Sub-transient current in the
    fault in motor = %.3f % .2f  p.u \n", abs(
    I_st_m),180+phasemag(I_st_m))
printf("\nNOTE: ERROR: Sub-transient reactance of
    motor is 0.45 p.u & not 0.35 p.u as mentioned in
    textbook statement")
```

Scilab code Exa 27.12 Sub transient fault current Fault current rating of generato

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 1: SYMMETRICAL SHORT CIRCUIT CAPACITY
   CALCULATIONS

// EXAMPLE : 1.12 :
// Page number 473-474
clear ; clc ; close ; // Clear the work space and
    console
```

```

// Given data
kVA_G = 625.0                // Generator rating(kVA)
V_G = 2.4                    // Voltage rating of generator(kV)
X_st_G = 8.0/100             // Sub-transient reactance of generator
rating_M = 250.0             // Motor rating(HP)
V_M = 2.4                    // Voltage rating of motor(kV)
n = 90.0/100                 // Efficiency of motor
X_st_M = 20.0/100            // Sub-transient reactance of motor

// Calculations
kVA_base = 625.0                                                    // Base kVA
input_M = rating_M*0.746/n                                // Each motor input(kVA)
X_st_m_pu = X_st_M*kVA_base/input_M                       // Sub-transient reactance of motor(p.u)
I_base = kVA_base/(3**0.5*V_M)                            // Base current(A)
Z_th = %i*X_st_m_pu/3*X_st_G/(X_st_m_pu/3+X_st_G)         // Thevenin impedance(p.u)
I_st = 1.0/Z_th                                                     // Initial symmetrical current at F(p.u)
I_st_g = I_st*(X_st_m_pu/3/(X_st_m_pu/3+X_st_G))          // Fault current rating of generator breaker(p.u)
I_st_m = (I_st-I_st_g)/3                                  // Fault current rating of each motor breaker(p.u)

// Results
```

```
disp("PART III - EXAMPLE : 1.12 : SOLUTION :-")
printf("\nSub-transient fault current at F = %.2fj p
    .u", imag(I_st))
printf("\nFault current rating of generator breaker
    = %.1fj p.u", imag(I_st_g))
printf("\nFault current rating of each motor breaker
     = %.2fj p.u", imag(I_st_m))
```

Chapter 28

FAULT LIMITING REACTORS

Scilab code Exa 28.1 Reactance necessary to protect the switchgear

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 2: FAULT LIMITING REACTORS

// EXAMPLE : 2.1 :
// Page number 479-480
clear ; clc ; close ; // Clear the work space and
   console

// Given data
kVA_A = 2500.0               // Rating of alternator A(
   kVA)
x_A = 8.0                    // Reactance of alternator
   A(%)
kVA_B = 5000.0               // Rating of alternator B(
```

```
kVA)
x_B = 6.0                // Reactance of alternator B(%)
kVA_CB = 150000.0        // Rating of circuit breaker(kVA)
kVA_T = 10000.0          // Rating of transformer(kVA)
x_T = 7.5                // Reactance of transformer (%)
V = 3300.0               // System voltage(V)

// Calculations
kVA_base = 10000.0                              // Base kVA
X_A = kVA_base/kVA_A*x_A                        // Reactance of generator A(%)
X_B = kVA_base/kVA_B*x_B                        // Reactance of generator B(%)
X_eq = X_A*X_B/(X_A+X_B)                        // Combined reactance of A & B(%)
kVA_SC_G = kVA_base/X_eq*100                    // Short-circuit kVA due to generators(kVA)
kVA_SC_T = kVA_base/x_T*100                     // Short-circuit kVA due to grid supply(kVA)
X = (kVA_base*100/(kVA_CB-kVA_SC_G))-x_T        // Reactance necessary to protect switchgear(%)
I_fl = kVA_base*1000/(3**0.5*V)                 // Full load current corresponding to 10000 kVA(A)
X_phase = X*V/(3**0.5*I_fl*100)                 // Actual value of reactance per phase(ohm)

// Results
disp("PART III - EXAMPLE : 2.1 : SOLUTION :-")
printf("\nReactance necessary to protect the switchgear = %.3f ohm/phase", X_phase)
```

Scilab code Exa 28.2 `kVA developed under short circuit when reactors are in circui`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 2: FAULT LIMITING REACTORS

// EXAMPLE : 2.2 :
// Page number 480
clear ; clc ; close ; // Clear the work space and console

// Given data
X = 10.0          // Reactance of reactor(%)
kVA = 30000.0     // Rating of generator(kVA)
X_sc = 20.0       // Short-circuit reactance(%)

// Calculations
X_1 = 1.0/3*(X_sc+X)              // Combined reactance of generator A,B,C & associated reactors(%)
X_2 = X_1+X                       // Combined reactance upto fault(%)
X_total_a = X_2/2.0               // Total reactance upto fault(%)
kVA_SC_a = 100/X_total_a*kVA      // Short-circuit kVA(kVA)
X_total_b = 1.0/4*X_sc            // Total reactance upto fault when E,F,G & H are short-circuited(%)
kVA_SC_b = 100/X_total_b*kVA      // Short-circuit kVA(kVA)

```

```
// Results
disp("PART III - EXAMPLE : 2.2 : SOLUTION :-")
printf("\nCase(a): kVA developed under short-circuit
    when reactors are in circuit = %.f kVA",
   kVA_SC_a)
printf("\nCase(b): kVA developed under short-circuit
    when reactors are short-circuited = %.f kVA",
   kVA_SC_b)
```

Scilab code Exa 28.4 Reactance of each reactor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 2: FAULT LIMITING REACTORS

// EXAMPLE : 2.4 :
// Page number 481
clear ; clc ; close ; // Clear the work space and
   console

// Given data
kVA = 20000.0    // Rating of generator(kVA)
f = 50.0         // Frequency(Hz)
V = 11.0*10**3   // Voltage of generator(V)
X_G = 20.0       // Generator short-circuit reactance
   (%)
x = 60.0         // Reactance falls to 60% normal
   value

// Calculations
kVA_base = 20000.0
```

```
// Base kVA
X = poly(0,"X")                                          // Reactance of each reactors E,F,G & H(%)
X_AE = X+X_G                                             // Reactances of A & E in series(%)
X_BF = X+X_G                                             // Reactances of B & F in series(%)
X_CD = X+X_G                                             // Reactances of C & D in series(%)
X_eq = X_AE/3                                            // X_eq = X_AE*X_BF*X_CD/(X_BF*X_CD+X_AE*X_CD+X_AE*X_BF). Combined reactances of 3 groups in parallel(%)
X_f = X_eq+X                                             // Reactances of these groups to fault via tie-bar(%)
X_sol = roots(6.66666666666667-(100-x)/100*(X_f))        // Value of reactance of each reactors E,F,G & H(%)
I_fl = kVA_base*1000/(3**0.5*V)                          // Full load current corresponding to 20000 kVA & 11 kV(A)
X_ohm = X_sol*V/(3**0.5*100*I_fl)                        // Ohmic value of reactance X(ohm)

// Results
disp("PART III - EXAMPLE : 2.4 : SOLUTION :-")
printf("\nReactance of each reactor = %.4f ohm \n",X_ohm)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 28.5 Instantaneous symmetrical short circuit MVA for a fault at X

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 2: FAULT LIMITING REACTORS

// EXAMPLE : 2.5 :
// Page number 481-482
clear ; clc ; close ; // Clear the work space and
    console

// Given data
kVA_base = 10000.0    // Base kVA
V = 6.6*10**3         // Voltage of generator(V)
X_A = 7.5             // Reactance of generator A(%)
X_B = 7.5             // Reactance of generator B(%)
X_C = 10.0            // Reactance of generator C(%)
X_D = 10.0            // Reactance of generator D(%)
X_E = 8.0             // Reactance of reactor E(%)
X_F = 8.0             // Reactance of reactor F(%)
X_G = 6.5             // Reactance of reactor G(%)
X_H = 6.5             // Reactance of reactor H(%)

// Calculations
Z_1 = X_B*X_C/(X_H+X_B+X_C)              // Impedance(
    %). Fig E2.7
Z_2 = X_H*X_C/(X_H+X_B+X_C)              // Impedance(
    %). Fig E2.7
Z_3 = X_B*X_H/(X_H+X_B+X_C)              // Impedance(
    %). Fig E2.7
```

```
Z_4 = Z_2+X_F                          // Impedance(%). Fig E2.8 & Fig 2.9
Z_5 = Z_3+X_E                          // Impedance(%). Fig E2.8 & Fig 2.9
Z_6 = X_D*Z_1/(X_D+Z_1+Z_4)            // Impedance(%). Fig E2.10
Z_7 = X_D*Z_4/(X_D+Z_1+Z_4)            // Impedance(%). Fig E2.10
Z_8 = Z_1*Z_4/(X_D+Z_1+Z_4)            // Impedance(%). Fig E2.10
Z_9 = Z_7+X_G                          // Impedance(%). Fig E2.11 & Fig 2.12
Z_10 = Z_8+Z_5                         // Impedance(%). Fig E2.11 & Fig 2.12
Z_11 = Z_9*Z_10/(Z_9+Z_10)             // Impedance(%). Fig 2.12 & Fig 2.13
Z_12 = Z_6+Z_11                        // Impedance(%). Fig 2.13
Z_eq = X_A*Z_12/(X_A+Z_12)             // Final Impedance(%). Fig 2.13 & Fig 2.14
MVA_SC = kVA_base*100/(Z_eq*1000)      // Instantaneous symmetrical short-circuit MVA for a fault at X(MVA)

// Results
disp("PART III - EXAMPLE : 2.5 : SOLUTION :-")
printf("\nInstantaneous symmetrical short-circuit MVA for a fault at X = %.f MVA \n", MVA_SC)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more approximation in the textbook")
```

Chapter 29

SYMMETRICAL COMPONENTS ANALYSIS

Scilab code Exa 29.1 Positive Negative and Zero sequence currents

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 3: SYMMETRICAL COMPONENTS' ANALYSIS

// EXAMPLE : 3.1 :
// Page number 487-488
clear ; clc ; close ; // Clear the work space and
   console

// Given data
I_R = complex(12.0,24.0)    // Line current(A)
I_Y = complex(16.0,-2.0)    // Line current(A)
I_B = complex(-4.0,-6.0)    // Line current(A)

// Calculations
```

```
alpha = exp(%i*120.0*%pi/180)                    // Operator
I_R0 = 1.0/3*(I_R+I_Y+I_B)                       // Zero sequence component(A)
I_R1 = 1.0/3*(I_R+alpha*I_Y+alpha**2*I_B)        // Positive sequence component(A)
I_R2 = 1.0/3*(I_R+alpha**2*I_Y+alpha*I_B)        // Negative sequence component(A)

// Results
disp("PART III - EXAMPLE : 3.1 : SOLUTION :-")
printf("\nPositive sequence current, I_R1 = (%.3f + %.1fj) A", real(I_R1),imag(I_R1))
printf("\nNegative sequence current, I_R2 = (%.3f + %.2fj) A", real(I_R2),imag(I_R2))
printf("\nZero sequence current, I_R0 = (%.1f + %.2fj) A", real(I_R0),imag(I_R0))
```

Scilab code Exa 29.4 Sequence components of currents in the resistors and Supply l

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 3: SYMMETRICAL COMPONENTS' ANALYSIS

// EXAMPLE : 3.4 :
// Page number 489-490
clear ; clc ; close ; // Clear the work space and console

// Given data
R_bc = 5.0     // Resistance of resistor connected b/
```

```
w b & c(ohm)
R_ca = 10.0    // Resistance of resistor connected b/w c & a(ohm)
R_ab = 20.0    // Resistance of resistor connected b/w a & b(ohm)
V = 100.0      // Voltage of balanced system(V)

// Calculations
E_A = -V                                          // Voltage across resistor connected b/w b & c(V)
angle = 60.0                                      // Angle in delta system( )
E_B = V*exp(%i*60.0*%pi/180)                      // Voltage across resistor connected b/w c & a(V)
E_C = V*exp(%i*-60.0*%pi/180)                     // Voltage across resistor connected b/w a & b(V)
I_A = E_A/R_bc                                    // Current flowing across resistor connected b/w b & c(A)
I_B = E_B/R_ca                                    // Current flowing across resistor connected b/w c & a(A)
I_C = E_C/R_ab                                    // Current flowing across resistor connected b/w a & b(A)
alpha = exp(%i*120.0*%pi/180)                     // Operator
I_A0 = 1.0/3*(I_A+I_B+I_C)                        // Zero sequence delta current(A)
I_A1 = 1.0/3*(I_A+alpha*I_B+alpha**2*I_C)         // Positive sequence delta current(A)
I_A2 = 1.0/3*(I_A+alpha**2*I_B+alpha*I_C)         // Negative sequence delta current(A)
I_a0 = 0.0                                        // Zero sequence star current(A)
I_a1 = (alpha-alpha**2)*I_A1                      // Positive sequence star current(A)
I_a2 = (alpha**2-alpha)*I_A2                      //
```

```
   Negative sequence star current(A)

// Results
disp("PART III - EXAMPLE : 3.4 : SOLUTION :-")
printf("\nCurrent in the resistors are:")
printf("\n I_A = (%.f+%.fj) A", real(I_A),imag(I_A))
printf("\n I_B = (%.f+%.2fj) A", real(I_B),imag(I_B)
   )
printf("\n I_C = (%.1f%.2fj) A", real(I_C),imag(I_C)
   )
printf("\nSequence components of currents in the
   resistors:")
printf("\n Zero-sequence current, I_A0 = (%.3f+%.2fj
   ) A", real(I_A0),imag(I_A0))
printf("\n Positive-sequence current, I_A1 = (%.2f+%
   .fj) A", real(I_A1),imag(I_A1))
printf("\n Negative-sequence current, I_A2 = (%.2f%
   .2fj) A", real(I_A2),imag(I_A2))
printf("\nSequence components of currents in the
   supply lines:")
printf("\n Zero-sequence current, I_a0 = %.f A",
   I_a0)
printf("\n Positive-sequence current, I_a1 = %.1fj A
   ", imag(I_a1))
printf("\n Negative-sequence current, I_a2 = (%.1f+%
   .2fj) A", real(I_a2),imag(I_a2))
```

Scilab code Exa 29.5 Magnitude of positive and Negative sequence components of the

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
```

```
// CHAPTER 3: SYMMETRICAL COMPONENTS' ANALYSIS

// EXAMPLE : 3.5 :
// Page number 490-491
clear ; clc ; close ; // Clear the work space and console

// Given data
E_a = 100.0     // Line to line voltage(V)
E_b = 150.0     // Line to line voltage(V)
E_c = 200.0     // Line to line voltage(V)

// Calculations
e_A = 1.0                                          // 100 V = 1 unit
e_B = 1.5                                          // 150 V = 1 unit
e_C = 2.0                                          // 200 V = 1 unit
cos_alpha = (e_C**2-e_A-e_B**2)/(2*e_B)
alpha = acosd(cos_alpha)                           // angle( )
cos_beta = (e_A+e_B*cos_alpha)/e_C
beta = acosd(cos_beta)                             // angle( )
E_A = E_a*exp(%i*180.0*%pi/180)                    // Voltage(V)
E_B = E_b*exp(%i*(180.0-alpha)*%pi/180)            // Voltage(V)
E_C = E_c*exp(%i*-beta*%pi/180)                    // Voltage(V)
a = exp(%i*120.0*%pi/180)                          // Operator
E_A0 = 1.0/3*(E_A+E_B+E_C)                         // Zero sequence voltage(V)
E_A1 = 1.0/3*(E_A+a*E_B+a**2*E_C)                  // Positive sequence delta voltage(V)
E_A1_mag = abs(E_A1)                               //
```

```
Magnitude of positive sequence delta voltage(V)
E_a1 = -%i/3**0.5*E_A1                                  // Positive sequence star voltage(V)
E_a1_mag = abs(E_a1)                                    // Magnitude of positive sequence star voltage(V)
E_A2 = 1.0/3*(E_A+a**2*E_B+a*E_C)                       // Negative sequence delta voltage(V)
E_A2_mag = abs(E_A2)                                    // Magnitude of negative sequence delta voltage(V)
E_a2 = %i/3**0.5*E_A2                                   // Negative sequence star voltage(V)
E_a2_mag = abs(E_a2)                                    // Magnitude of negative sequence star voltage(V)

// Results
disp("PART III - EXAMPLE : 3.5 : SOLUTION :-")
printf("\nMagnitude of positive sequence delta voltage, |E_A1| = %.f V", E_A1_mag)
printf("\nMagnitude of positive sequence star voltage, |E_a1| = %.1f V", E_a1_mag)
printf("\nMagnitude of negative sequence delta voltage, |E_A2| = %.f V", E_A2_mag)
printf("\nMagnitude of negative sequence star voltage, |E_a2| = %.f V", E_a2_mag)
```

Scilab code Exa 29.6 Current in each line by the method of symmetrical components

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 3: SYMMETRICAL COMPONENTS' ANALYSIS

```

```
// EXAMPLE : 3.6 :
// Page number 491-492
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 2300.0                                      // Rated voltage(V)
kVA = 500.0                                     // kVA rating
E_A = 2760.0*exp(%i*0*%pi/180)                  // Line voltage(V)
E_B = 2300.0*exp(%i*-138.6*%pi/180)             // Line voltage(V)
E_C = 1840.0*exp(%i*124.2*%pi/180)              // Line voltage(V)

// Calculations
a = exp(%i*120.0*%pi/180)                   // Operator
E_A1 = 1.0/3*(E_A+a*E_B+a**2*E_C)           // Positive sequence voltage(V)
E_A2 = 1.0/3*(E_A+a**2*E_B+a*E_C)           // Negative sequence voltage(V)
E_a1 = -%i/3**0.5*E_A1                      // Positive sequence star voltage(V)
E_a2 = %i/3**0.5*E_A2                       // Negative sequence star voltage(V)
E_a0 = 0.0                                  // Zero sequence voltage(V)
E_a = E_a1+E_a2+E_a0                        // Symmetrical voltage component(V)
R = V**2/(kVA*1000)                         // Resistance(ohm)
I_a = abs(E_a)/R                            // Current in line a(A)
E_b = a**2*E_a1+a*E_a2+E_a0                 // Symmetrical voltage component(V)
```

```
I_b = abs(E_b)/R                              // Current in line b(A)
E_c = a*E_a1+a**2*E_a2+E_a0                   // Symmetrical voltage component(V)
I_c = abs(E_c)/R                              // Current in line c(A)

// Results
disp("PART III - EXAMPLE : 3.6 : SOLUTION :-")
printf("\nCurrent in line a, |I_a| = %.1f A", I_a)
printf("\nCurrent in line b, |I_b| = %.f A", I_b)
printf("\nCurrent in line c, |I_c| = %.1f A \n", I_c)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 29.7 Symmetrical components of line current if phase 3 is only swi

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 3: SYMMETRICAL COMPONENTS' ANALYSIS

// EXAMPLE : 3.7 :
// Page number 492-493
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 2300.0                                    // Rated voltage(V)
kVA = 500.0                                   // kVA
```

```
    rating
I_1 = 100.0                                    // Line current(A)
I_2 = 100.0*exp(%i*180*%pi/180)                // Line current(A)
I_3 = 0                                        // Line current(A)

// Calculations
a = exp(%i*120.0*%pi/180)                 // Operator
I_10 = 1.0/3*(I_1+I_2+I_3)                // Symmetrical component of line current for phase 1(A)
I_11 = 1.0/3*(I_1+a*I_2+a**2*I_3)         // Symmetrical component of line current for phase 1(A)
I_12 = 1.0/3*(I_1+a**2*I_2+a*I_3)         // Symmetrical component of line current for phase 1(A)
I_20 = I_10                               // Symmetrical component of line current for phase 2(A)
I_21 = a**2*I_11                          // Symmetrical component of line current for phase 2(A)
I_22 = a*I_12                             // Symmetrical component of line current for phase 2(A)
I_30 = I_10                               // Symmetrical component of line current for phase 3(A)
I_31 = a*I_11                             // Symmetrical component of line current for phase 3(A)
I_32 = a**2*I_12                          // Symmetrical component of line current for phase 3(A)

```

```
// Results
disp("PART III - EXAMPLE : 3.7 : SOLUTION :-")
printf("\nSymmetrical component of line current for phase 1:")
printf("\n I_10 = %.1f A", abs(I_10))
printf("\n I_11 = %.2f % .f  A", abs(I_11), phasemag(I_11))
printf("\n I_12 = %.2f % .f  A", abs(I_12), phasemag(I_12))
printf("\nSymmetrical component of line current for phase 2:")
printf("\n I_20 = %.1f A", abs(I_20))
printf("\n I_21 = %.2f % .f  A", abs(I_21), phasemag(I_21))
printf("\n I_22 = %.2f % .f  A", abs(I_22), phasemag(I_22))
printf("\nSymmetrical component of line current for phase 3:")
printf("\n I_30 = %.1f A", abs(I_30))
printf("\n I_31 = %.2f % .f  A", abs(I_31), phasemag(I_31))
printf("\n I_32 = %.2f % .f  A", abs(I_32), phasemag(I_32))
```

Scilab code Exa 29.8 Positive Negative and Zero sequence components of currents fo

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 3: SYMMETRICAL COMPONENTS' ANALYSIS

// EXAMPLE : 3.8 :
```

```
// Page number 493
clear ; clc ; close ; // Clear the work space and
    console

// Given data
I_a = 1000.0                                // Current to
     earth(A)
I_b = 0                                     // Current(A)
I_c = 0                                     // Current(A)

// Calculations
a = exp(%i*120.0*%pi/180)                   // Operator
I_a0 = 1.0/3*(I_a+I_b+I_c)                  // Zero
    sequence component of current(A)
I_b0 = I_a0                                 // Zero
    sequence component of current(A)
I_c0 = I_a0                                 // Zero
    sequence component of current(A)
I_a1 = 1.0/3*(I_a+a*I_b+a**2*I_c)           // Positive
    sequence component of current(A)
I_b1 = a**2*I_a1                            // Positive
    sequence component of current(A)
I_c1 = a*I_a1                               // Positive
    sequence component of current(A)
I_a2 = 1.0/3*(I_a+a**2*I_b+a*I_c)           // Negative
    sequence component of current(A)
I_b2 = a*I_a2                               // Negative
    sequence component of current(A)
I_c2 = a**2*I_a2                            // Negative
    sequence component of current(A)

// Results
disp("PART III - EXAMPLE : 3.8 : SOLUTION :-")
printf("\nZero sequence component of current for all
     phases are")
printf("\n I_a0 = %.1f% .f  A", abs(I_a0),
    phasemag(I_a0))
printf("\n I_b0 = %.1f% .f  A", abs(I_b0),
```

```
phasemag(I_b0))
printf("\n I_c0 = %.1f % .f  A", abs(I_c0), phasemag(I_c0))
printf("\nPositive sequence component of current for all phases are")
printf("\n I_a1 = %.1f % .f  A", abs(I_a1), phasemag(I_a1))
printf("\n I_b1 = %.1f % .f  A", abs(I_b1),360+phasemag(I_b1))
printf("\n I_c1 = %.1f % .f  A", abs(I_c1), phasemag(I_c1))
printf("\nNegative sequence component of current for all phases are")
printf("\n I_a2 = %.1f % .f  A", abs(I_a2), phasemag(I_a2))
printf("\n I_b2 = %.1f % .f  A", abs(I_b2), phasemag(I_b2))
printf("\n I_c2 = %.1f % .f  A", abs(I_c2),360+phasemag(I_c2))
```

Scilab code Exa 29.9 Currents in all the lines and their symmetrical components

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 3: SYMMETRICAL COMPONENTS' ANALYSIS

// EXAMPLE : 3.9 :
// Page number 493-494
clear ; clc ; close ; // Clear the work space and console

```

```
// Given data
I_A = 1000.0                              // Current through line A(A)
I_C = 0                                   // Current through line C(A)

// Calculations
I_B = 1000.0*exp(%i*180.0*%pi/180)        // Current through line B(A)
a = exp(%i*120.0*%pi/180)                 // Operator
I_a0 = 1.0/3*(I_A+I_B+I_C)                // Zero sequence component of current(A)
I_b0 = I_a0                               // Zero sequence component of current(A)
I_c0 = I_a0                               // Zero sequence component of current(A)
I_a1 = 1.0/3*(I_A+a*I_B+a**2*I_C)         // Positive sequence component of current(A)
I_b1 = a**2*I_a1                          // Positive sequence component of current(A)
I_c1 = a*I_a1                             // Positive sequence component of current(A)
I_a2 = 1.0/3*(I_A+a**2*I_B+a*I_C)         // Negative sequence component of current(A)
I_b2 = a*I_a2                             // Negative sequence component of current(A)
I_c2 = a**2*I_a2                          // Negative sequence component of current(A)

// Results
disp("PART III - EXAMPLE : 3.9 : SOLUTION :-")
printf("\nCurrent in line A, I_A = %.f %.f  A", abs(I_A),phasemag(I_A))
printf("\nCurrent in line B, I_B = %.f %.f  A", abs(I_B),phasemag(I_B))
printf("\nCurrent in line C, I_C = %.f A", I_C)
printf("\nSymmetrical current components of line A are:")
```

```
printf("\n I_a0 = %.f A", abs(I_a0))
printf("\n I_a1 = %.1 f % . f   A", abs(I_a1),
   phasemag(I_a1))
printf("\n I_a2 = %.1 f % . f   A", abs(I_a2),
   phasemag(I_a2))
printf("\nSymmetrical current components of line B
   are:")
printf("\n I_b0 = %.f A", abs(I_b0))
printf("\n I_b1 = %.1 f % . f   A", abs(I_b1),
   phasemag(I_b1))
printf("\n I_b2 = %.1 f % . f   A", abs(I_b2),
   phasemag(I_b2))
printf("\nSymmetrical current components of line C
   are:")
printf("\n I_c0 = %.f A", abs(I_c0))
printf("\n I_c1 = %.1 f % . f   A", abs(I_c1),
   phasemag(I_c1))
printf("\n I_c2 = %.1 f % . f   A", abs(I_c2),
   phasemag(I_c2))
```

Scilab code Exa 29.10 Radius of voltmeter connected to the yellow line and Current

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 3: SYMMETRICAL COMPONENTS' ANALYSIS

// EXAMPLE : 3.10 :
// Page number 494
clear ; clc ; close ; // Clear the work space and
   console

```

```
// Given data
R = 20000.0                                         // Resistance of voltmeter(ohm)
E_R = 100.0                                         // Line-to-neutral voltage(A)
E_Y = 200.0*exp(%i*270.0*%pi/180)                   // Line-to-neutral voltage(A)
E_B = 100.0*exp(%i*120.0*%pi/180)                   // Line-to-neutral voltage(A)

// Calculations
a = exp(%i*120.0*%pi/180)                   // Operator
V_R0 = 1.0/3*(E_R+E_Y+E_B)                  // Zero sequence voltage(V)
V_R1 = 1.0/3*(E_R+a*E_Y+a**2*E_B)           // Positive sequence voltage(V)
V_R2 = 1.0/3*(E_R+a**2*E_Y+a*E_B)           // Negative sequence voltage(V)
I_R1 = V_R1/R                               // Positive sequence current(A)
I_R2 = V_R2/R                               // Negative sequence current(A)
V_Y1 = a**2*V_R1                            // Positive sequence voltage of line Y(V)
V_Y2 = a*V_R2                               // Negative sequence voltage of line Y(V)
V_Y = V_Y1+V_Y2                             // Voltmeter reading connected to the yellow line(V)
I_Y = abs(V_Y)/R*1000                       // Current through voltmeter(mA)

// Results
disp("PART III - EXAMPLE : 3.10 : SOLUTION :-")
printf("\nVoltmeter reading connected to the yellow line, |V_Y| = %.1f V", abs(V_Y))
printf("\nCurrent through voltmeter, I_Y = %.3f mA \n", I_Y)
printf("\nNOTE: Changes in the obtained answer from
```

```
that of textbook is due to more precision here")
```

Scilab code Exa 29.11 `Three line currents and Wattmeter reading`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 3: SYMMETRICAL COMPONENTS' ANALYSIS

// EXAMPLE : 3.11 :
// Page number 495
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 400.0                // Voltage(V)
Z_ab = 20.0              // Resistor load(ohm)
Z_bc = -%i*40.0          // Capacitor load(ohm)
Z_ca = 5.0+%i*10.0       // Inductor and resistance
   load(ohm)

// Calculations
V_ab = V                                          //
    Line voltage(V)
V_bc = V*exp(%i*-120.0*%pi/180)                   //
    Line voltage(V)
V_ca = V*exp(%i*120.0*%pi/180)                    //
    Line voltage(V)
I_ab = V_ab/Z_ab                                  //
    Current(A)
I_bc = V_bc/Z_bc                                  //
    Current(A)
```

```
I_ca = V_ca/Z_ca                                    // Current(A)
I_a = I_ab-I_ca                                     // Line current(A)
I_b = I_bc-I_ab                                     // Line current(A)
I_c = I_ca-I_bc                                     // Line current(A)
phi = -120.0-phasemag(I_a)                          // (  )
P = abs(I_a*V_bc)*cosd(phi)/1000                    // Wattmeter reading(kW)

// Results
disp("PART III - EXAMPLE : 3.11 : SOLUTION :-")
printf("\nLine currents are:")
printf("\n I_a = %.1f % .1f  A", abs(I_a),phasemag(I_a))
printf("\n I_b = %.1f % .2f  A", abs(I_b),phasemag(I_b))
printf("\n I_c = %.2f % .f  A", abs(I_c),phasemag(I_c))
printf("\nWattmeter reading, P = %.2f kW \n", P)
printf("\nNOTE: ERROR: Calculation mistakes in the textbook solution")
```

Chapter 30

UNSYMMETRICAL FAULTS IN POWER SYSTEMS

Scilab code Exa 30.1 Initial symmetrical rms line currents Ground wire currents an

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.1 :
// Page number 510-512
clear ; clc ; close ; // Clear the work space and
   console

// Given data
MVA = 15.0     // Generator rating(MVA)
kV = 6.9       // Generator voltage(kV)
X_1 = 25.0     // Positive sequence reactance(%)
X_2 = 25.0     // Negative sequence reactance(%)
X_0 = 8.0      // Zero sequence reactance(%)
```

```
X = 6.0         // Reactor placed in line(%)

// Calculations
a = exp(%i*120.0*%pi/180)                                   // Operator
Z_1 = %i*X_1/100                                            // Positive sequence impedance(p.u)
Z_2 = %i*X_2/100                                            // Negative sequence impedance(p.u)
Z_g0 = %i*X_0/100                                           // Impedance(p.u)
Z = %i*X/100                                                // Impedance(p.u)
Z_0 = Z_g0+3*Z                                              // Zero sequence impedance(p.u)
E_a = 1.0                                                   // Voltage(p.u)
E_b = a**2*E_a                                              // Voltage(p.u)
// Case(a)
I_a0_a = 0                                                  // Current(A)
I_a1_a_pu = 1.0/(Z_1+Z_2)                                   // Current(p.u)
I_a1_a = I_a1_a_pu*MVA*1000/(3**0.5*kV)                     // Current(A)
I_a2_a = -I_a1_a                                            // Current(A)
```

```
I_b0_a = 0
    // Current(A)
I_b1_a = a**2*I_a1_a                                              //
    Current(A)
I_b2_a = a*I_a2_a
    // Current(A)
I_a_a = I_a1_a+I_a2_a                                             //
    Line current(A)
I_b_a = I_b1_a+I_b2_a                                             //
    Line current(A)
I_c_a = -I_b_a
    // Line current(A)
I_g_a = 0
    // Ground wire current(A)
V_a_a = (E_a-I_a1_a*Z_1-I_a2_a*Z_2-I_a0_a*Z_0)*kV
    *1000/3**0.5      // Voltage(V)
V_b_a = (a**2*E_a+%i*3**0.5*I_a1_a_pu*Z_1)*kV
    *1000/3**0.5          // Voltage(V)
V_c_a = V_b_a
    // Voltage(V)
// Case(b)
I_a1_b_pu = E_a/(Z_1+(Z_2*Z_0/(Z_2+Z_0)))
                             // Current(p.u)
I_a1_b = I_a1_b_pu*MVA*1000/(3**0.5*kV)
                               // Current(A)
I_a2_b_pu = -Z_0*Z_2/(Z_2*(Z_0+Z_2))*I_a1_b_pu
                         // Current(p.u)
I_a2_b = -Z_0*Z_2/(Z_2*(Z_0+Z_2))*I_a1_b
                              // Current(A)
I_a0_b_pu = -Z_0*Z_2/(Z_0*(Z_0+Z_2))*I_a1_b_pu
```

```
                    // Current(p.u)
I_a0_b = -Z_0*Z_2/(Z_0*(Z_0+Z_2))*I_a1_b                    // Current(A)
I_a_b = I_a0_b+I_a1_b+I_a2_b                    // Line current(A)
I_b_b = I_a0_b+a**2*I_a1_b+a*I_a2_b                    // Line current(A)
I_c_b = I_a0_b+a*I_a1_b+a**2*I_a2_b                    // Line current(A)
I_0_b = 3*I_a0_b                    // Current in the ground resistor(A)
V_a_b_pu = E_a-I_a1_b_pu*Z_1-I_a2_b_pu*Z_2-I_a0_b_pu*Z_0          // Voltage(p.u)
V_a_b = abs(V_a_b_pu)*kV*1000/(3**0.5)                    // Voltage(V)
V_b_b = 0                    // Voltage(V)
V_c_b = 0                    // Voltage(V)

// Results
disp("PART III - EXAMPLE : 4.1 : SOLUTION :-")
printf("\nCase(a): Initial symmetrical rms line current when ground is not involved in fault, I_a = %.f A", abs(I_a_a))
printf("\n         Initial symmetrical rms line current when ground is not involved in fault, I_b = %.f A", real(I_b_a))
printf("\n         Initial symmetrical rms line current when ground is not involved in fault, I_c = %.f A", real(I_c_a))
printf("\n         Ground wire current = %.f A", I_g_a)
printf("\n         Line to neutral voltage, V_a = %.
```

```
f V", real(V_a_a))
printf("\n          Line to neutral voltage, V_b = %.f V", real(V_b_a))
printf("\n          Line to neutral voltage, V_c = %.f V", real(V_c_a))
printf("\nCase(b): Initial symmetrical rms line current when fault is solidly grounded, I_a = %.f A", abs(I_a_b))
printf("\n          Initial symmetrical rms line current when fault is solidly grounded, I_b = (%.f+%.fj) A", real(I_b_b),imag(I_b_b))
printf("\n          Initial symmetrical rms line current when fault is solidly grounded, I_c = (%.f+%.fj) A", real(I_c_b),imag(I_c_b))
printf("\n          Ground wire current = %.fj A", imag(I_0_b))
printf("\n          Line to neutral voltage, V_a = %.f V", V_a_b)
printf("\n          Line to neutral voltage, V_b = %.f V", V_b_b)
printf("\n          Line to neutral voltage, V_c = %.f V\n", V_c_b)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here and approximation in textbook")
```

Scilab code Exa 30.2 `Current in the line with two lines short circuited`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS
```

```

// EXAMPLE : 4.2 :
// Page number 512
clear ; clc ; close ; // Clear the work space and console

// Given data
kVA = 10000.0      // Generator rating(kVA)
f = 50.0           // Frequency(Hz)
I_1 = 30.0         // Positive sequence current(%)
I_2 = 10.0         // Negative sequence current(%)
I_0 = 5.0          // Zero sequence current(%)
d = 1.0/100        // Diameter of conductor(m)
D = 5.0            // Triangular spacing(m)
kV = 30.0          // Generator voltage on open-circuit(kV)
l = 20.0           // Distance of line at short circuit occurance(km)

// Calculations
a = exp(%i*120.0*%pi/180)                                   // Operator
Z_g1 = kV**2*I_1*I_2/kVA                                    // Positive phase sequence reactance of generator(ohm)
Z_g2 = Z_g1*I_2/I_1                                         // Negative phase sequence reactance of generator(ohm)
Z_g0 = Z_g1*I_0/I_1                                         // Zero phase sequence reactance of generator(ohm)
r = d/2                                                     // Radius of conductor(m)
Z_l1 = 2.0*%pi*f*(0.5+4.606*log10(D/r))*10**-7*l
```

```
    *1000               // Positive phase sequence reactance of line(ohm)
Z_l2 = 2.0*%pi*f*(0.5+4.606*log10(D/r))*10**-7*l*1000               // Negative phase sequence reactance of line(ohm)
Z_1 = %i*(Z_g1+Z_l1)                                   // Z1 upto the point of fault(ohm)
Z_2 = %i*(Z_g2+Z_l2)                                   // Z2 upto the point of fault(ohm)
E_a = kV*1000/3**0.5                                   // Phase voltage(V)
I_a1 = E_a/(Z_1+Z_2)                                   // Positive sequence current in line a(A)
I_a2 = -I_a1                                           // Negative sequence current in line a(A)
I_a0 = 0                                               // Zero sequence current in line a(A)
I_b0 = 0                                               // Zero sequence current in line b(A)
I_c0 = 0                                               // Zero sequence current in line c(A)
I_a = I_a0+I_a1+I_a2                                   // Current in line a(A)
I_b = I_b0+a**2*I_a1+a*I_a2                            // Current in line b(A)
I_c = I_c0+a*I_a1+a**2*I_a2                            // Current in line c(A)
```

```
// Results
disp("PART III - EXAMPLE : 4.2 : SOLUTION :-")
printf("\nCurrent in line a, I_a = %.f A", abs(I_a))
printf("\nCurrent in line b, I_b = %.f A", real(I_b))
printf("\nCurrent in line c, I_c = %.f A", real(I_c))
```

Scilab code Exa 30.3 Fault current Sequence component of current and Voltages of t

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.3 :
// Page number 512-513
clear ; clc ; close ; // Clear the work space and console

// Given data
kVA = 10000.0                    // Alternator rating(kVA)
Z_g1 = complex(0.5,4.7)          // Positive sequence impedance(ohm/phase)
Z_g2 = complex(0.2,0.6)          // Negative sequence impedance(ohm/phase)
Z_g0 = complex(0,0.43)           // Zero sequence impedance(ohm/phase)
Z_l1 = complex(0.36,0.25)        // Impedance(ohm)
Z_l2 = complex(0.36,0.25)        // Impedance(ohm)
```

```
Z_l0 = complex(2.9,0.95)         // Impedance(ohm)
V = 6600.0                       // Voltage(V)

// Calculations
a = exp(%i*120.0*%pi/180)                                   // Operator
// Case(a)
E_a = V/3**0.5                                              // Phase voltage(V)
Z_1 = Z_g1+Z_l1                                             // Z1 upto the point of fault(ohm)
Z_2 = Z_g2+Z_l2                                             // Z2 upto the point of fault(ohm)
Z_0 = Z_g0+Z_l0                                             // Z0 upto the point of fault(ohm)
I_a = 3*E_a/(Z_1+Z_2+Z_0)                                   // Fault current(A)
// Case(b)
I_a0 = abs(I_a)/3                                           // Zero sequence current of line a(A)
I_a1 = abs(I_a)/3                                           // Positive sequence current of line a(A)
I_a2 = abs(I_a)/3                                           // Negative sequence current of line a(A)
I_b0 = I_a0                                                 // Zero sequence current of line b(A)
I_b1 = a**2*I_a1                                            // Positive sequence current of line b(A)
I_b2 = a*I_a2
```

```
                                                    //
    Negative sequence current of line b(A)
I_c0 = I_a0                                         // Zero sequence current of line c(A)
I_c1 = a*I_a1                                       // Positive sequence current of line c(A)
I_c2 = a**2*I_a2                                    // Negative sequence current of line c(A)
// Case(c)
V_b = E_a/(Z_1+Z_2+Z_0)*((a**2-a)*Z_2+(a**2-1)*Z_0)     // Voltage of the line b(V)
V_c = E_a/(Z_1+Z_2+Z_0)*((a-a**2)*Z_2+(a-1)*Z_0)        // Voltage of the line c(V)

// Results
disp("PART III - EXAMPLE : 4.3 : SOLUTION :-")
printf("\nCase(a): Fault current, |I_a| = %.f A", abs(I_a))
printf("\nCase(b): Zero sequence current of line a, I_a0 = %.f A", I_a0)
printf("\n         Positive sequence current of line a, I_a1 = %.f A", I_a1)
printf("\n         Negative sequence current of line a, I_a2 = %.f A", I_a2)
printf("\n         Zero sequence current of line b, I_b0 = %.f A", I_b0)
printf("\n         Positive sequence current of line b, I_b1 = (%.1f%.1fj) A", real(I_b1),imag(I_b1))
printf("\n         Negative sequence current of line b, I_b2 = (%.1f+%.1fj) A", real(I_b2),imag(I_b2))
printf("\n         Zero sequence current of line c, I_c0 = %.f A", I_c0)
printf("\n         Positive sequence current of line c, I_c1 = (%.1f+%.1fj) A", real(I_c1),imag(I_c1)
```

```
)
printf("\n           Negative sequence current of line c, I_c2 = (%.1f%.1fj) A", real(I_c2),imag(I_c2))
printf("\nCase(c): Voltage of the sound line to earth at fault, |V_b| = %.f V", abs(V_b))
printf("\n           Voltage of the sound line to earth at fault, |V_c| = %.f V\n", abs(V_c))
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 30.4 Fault currents in each line and Potential above earth attaine

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.4 :
// Page number 513-514
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 11000.0                      // Alternator voltage(V)
kVA = 50000.0                    // Alternator rating(kVA)
Z_l1 = complex(0.4,0.7)          // Positive sequence impedance of feeder(ohm)
Z_l2 = complex(0.4,0.7)          // Negative sequence impedance of feeder(ohm)
Z_l0 = complex(0.7,3.0)          // Zero sequence
```

```
    impedance of feeder(ohm)
Z_g1_A = complex(0,0.6)          // Positive sequence reactance(ohm)
Z_g1_B = complex(0,0.6)          // Positive sequence reactance(ohm)
Z_g2_A = complex(0,0.4)          // Negative sequence reactance(ohm)
Z_g2_B = complex(0,0.4)          // Negative sequence reactance(ohm)
Z_g0_A = complex(0,0.2)          // Zero sequence reactance(ohm)
Z_g0_B = complex(0,0.2)          // Zero sequence reactance(ohm)
Z_n_A = complex(0,0.2)           // Neutral reactance(ohm)
Z_n_B = complex(0,0.2)           // Neutra reactance(ohm)

// Calculations
a = exp(%i*120.0*%pi/180)                    // Operator
Z_g1 = 1.0/((1/Z_g1_A)+(1/Z_g1_B))           // Equivalent positive sequence impedance(ohm)
Z_g2 = 1.0/((1/Z_g2_A)+(1/Z_g2_B))           // Equivalent negative sequence impedance(ohm)
Z_g0 = 1.0/((1/Z_g0_A)+(1/Z_g0_B))           // Equivalent zero sequence impedance(ohm)
Z_n = 1.0/((1/Z_n_A)+(1/Z_n_B))              // Equivalent neutral impedance(ohm)
Z_1 = Z_l1+Z_g1                              // Positive sequence impedance(ohm)
Z_2 = Z_l2+Z_g2                              // Negative sequence impedance(ohm)
Z_0 = Z_l0+Z_g0+3*Z_n                        // Zero sequence impedance(ohm)
Z = Z_0*Z_2/(Z_0+Z_2)                        // Impedance(ohm)
E_R = V/3**0.5                               //
```

```
Phase voltage(V)
I_R1 = E_R/(Z_1+Z)                          // Postive sequence current(A)
I_R2 = -Z*I_R1/Z_2                          // Negative sequence current(A)
I_R0 = -Z*I_R1/Z_0                          // Zero sequence current(A)
I_R = I_R0+I_R1+I_R2                        // Fault current in line(A)
I_Y = I_R0+a**2*I_R1+a*I_R2                 // Fault current in line(A)
I_B = I_R0+a*I_R1+a**2*I_R2                 // Fault current in line(A)
I_earth = 3.0*I_R0                          // Current through earth reactance(A)
V_neutral = abs(I_earth*Z_n)                // Magnitude of potential above earth attained by generator neutral(V)

// Results
disp("PART III - EXAMPLE : 4.4 : SOLUTION :-")
printf("\nFault current in the line R, I_R = %.f A", abs(I_R))
printf("\nFault current in the line Y, I_Y = (%.f%.fj) A", real(I_Y),imag(I_Y))
printf("\nFault current in the line B, I_B = (%.f+%.fj) A", real(I_B),imag(I_B))
printf("\nPotential above earth attained by the alternator neutrals = %.f V\n", V_neutral)
printf("\nNOTE: ERROR: Voltage is 11000 not 11000 kV as given in textbook statement")
printf("\n      Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 30.5 Fault currents

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.5 :
// Page number 514-515
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 6600.0                 // Alternator voltage(V)
kVA = 10000.0              // Alternator rating(kVA)
x_1 = 15.0                 // Reactance to positive
    sequence current(%)
x_2 = 75.0                 // Reactance to negative
    sequence current(%)
x_0 = 30.0                 // Reactance to zero sequence
     current(%)
R_earth = 0.3              // Earth resistance(ohm)

// Calculations
a = exp(%i*120.0*%pi/180)                  // Operator
E_g = V/3**0.5                             // Phase
    voltage(V)
// Case(a)
I = kVA*1000/(3**0.5*V)                    // Full load
     current of each alternator(A)
X = x_1*V/(100*3**0.5*I)                   // Positive
    sequence reactance(ohm)
Z_g1 = %i*X                                //
    Equivalent positive sequence impedance(ohm)
Z_g2 = Z_g1*x_2/100                        //
    Equivalent negative sequence impedance(ohm)
Z_g0 = Z_g1*x_0/100                        //
```

```
                                        Equivalent zero sequence impedance(ohm)
Z_1 = Z_g1/3                            // Positive sequence impedance(ohm)
Z_2 = Z_g2/3                            // Negative sequence impedance(ohm)
Z_0 = Z_g0/3                            // Zero sequence impedance(ohm)
I_a_a = 3*E_g/(Z_1+Z_2+Z_0)             // Fault current(A)
// Case(b)
Z_0_b = Z_g0                            // Impedance (ohm)
I_a_b = 3*E_g/(Z_1+Z_2+Z_0_b)           // Fault current(A)
// Case(c)
Z_0_c = R_earth*3+Z_g0                  // Impedance (ohm)
I_a_c = 3*E_g/(Z_1+Z_2+Z_0_c)           // Fault current(A)

// Results
disp("PART III - EXAMPLE : 4.5 : SOLUTION :-")
printf("\nCase(a): Fault current if all the alternator neutrals are solidly earthed, I_a = %.fj A", imag(I_a_a))
printf("\nCase(b): Fault current if only one of the alternator neutrals is solidly earthed & others isolated = %.fj A", imag(I_a_b))
printf("\nCase(c): Fault current if one of alternator neutrals is earthed through resistance & others isolated = %.f A\n", abs(I_a_c))
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 30.6 Fault current for line fault and Line to ground fault

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.6 :
// Page number 515-516
clear ; clc ; close ; // Clear the work space and
    console

// Given data
kVA_G = 2000.0          // Generator rating(kVA)
X_G = 10.0              // Generator reactance(%)
kVA_T1 = 2000.0         // Transformer rating(kVA)
lv_T1 = 6.6             // LV side voltage(kV)
hv_T1 = 11.0            // HV side voltage(kV)
X_T1 = 5.0              // Transformer reactance(%)
X_cable = 0.5           // Cable reactance(ohm)
V_cable = 11.0          // Cable voltage(V)
kVA_T2 = 2000.0         // Transformer rating(kVA)
lv_T2 = 6.6             // LV side voltage(kV)
hv_T2 = 11.0            // HV side voltage(kV)
X_T2 = 5.0              // Transformer reactance(%)

// Calculations
a = exp(%i*120.0*%pi/180)                    //
    Operator
kVA_base = 2000.0                            // Base
     kVA
kV = 6.6                                     // Base
     voltage(kV)
X_1 = X_G*kV**2*10/kVA_base                  // 10%
    reactance at 6.6 kV(ohm)
X_2 = X_T1*kV**2*10/kVA_base                 // 5%
    reactance at 6.6 kV(ohm)
```

```
X_3 = (kV/hv_T1)**2*X_cable                        // 0.5 ohm at 11kV when referred to 6.6kV(ohm)
Z_g1 = %i*X_1                                      // Positive sequence impedance of generator(ohm)
Z_g2 = Z_g1*0.7                                    // Negative sequence impedance of generator equal to 70% of +ve sequence impedance(ohm)
T1_Z_T1_1 = %i*X_2                                 // Positive sequence impedance of transformer(ohm)
T1_Z_T1_2 = %i*X_2                                 // Negative sequence impedance of transformer(ohm)
Z_C1 = %i*X_3                                      // Positive sequence impedance of cable(ohm)
Z_C2 = %i*X_3                                      // Negative sequence impedance of cable(ohm)
T2_Z_T2_1 = %i*X_2                                 // Positive sequence impedance of transformer(ohm)
T2_Z_T2_2 = %i*X_2                                 // Negative sequence impedance of transformer(ohm)
Z_1 = Z_g1+T1_Z_T1_1+Z_C1+T2_Z_T2_1                // Positive sequence impedance(ohm)
Z_2 = Z_g2+T1_Z_T1_2+Z_C2+T2_Z_T2_2                // Negative sequence impedance(ohm)
Z_0 = %i*X_2                                       // Zero sequence impedance(ohm)
E_a = kV*1000/3**0.5                               // Phase voltage(V)
// Case(a)
I_a1 = E_a/(Z_1+Z_2)                               // Positive sequence current(A)
I_a2 = -I_a1                                       // Negative sequence current(A)
I_a0 = 0                                           // Zero sequence current(A)
I_a = I_a1+I_a2+I_a0                               // Fault current in line a(A)
I_b = (a**2-a)*I_a1                                // Fault current in line b(A)
```

```
I_c = -I_b                                          // Fault current in line c(A)
// Case(b)
I_a_b = 3*E_a/(Z_1+Z_2+Z_0)                         // Fault current for line to ground fault(A)

// Results
disp("PART III - EXAMPLE : 4.6 : SOLUTION :-")
printf("\nCase(a): Fault current for line fault are")
printf("\n          I_a = %.f A", abs(I_a))
printf("\n          I_b = %.f A", abs(I_b))
printf("\n          I_c = %.f A", abs(I_c))
printf("\nCase(b): Fault current for line to ground fault, |I_a| = %.f A\n", abs(I_a_b))
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 30.7 Fault current for a LG fault at C

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.7 :
// Page number 516-518
clear ; clc ; close ; // Clear the work space and console

// Given data
MVA_G1 = 40.0        // Generator rating(MVA)
```

```
kV_G1 = 13.2         // Generator voltage(kV)
X_st_G1 = 0.15       // Sub−transient reactance(p.u)
X_2_G1 = 0.15        // Negative sequence reactance(p.u)
X_0_G1 = 0.08        // Zero sequence reactance(p.u)
MVA_G3 = 60.0        // Generator rating(MVA)
kV_G3 = 13.8         // Generator voltage(kV)
X_st_G3 = 0.20       // Sub−transient reactance(p.u)
X_2_G3 = 0.20        // Negative sequence reactance(p.u)
X_0_G3 = 0.08        // Zero sequence reactance(p.u)
MVA_T1 = 40.0        // Transformer rating(MVA)
kV_lv_T1 = 13.8      // Transformer low voltage(kV)
kV_hv_T1 = 138       // Transformer high voltage(kV)
X_1_T1 = 0.10        // Positive sequence reactance(p.u)
X_2_T1 = 0.10        // Negative sequence reactance(p.u)
X_0_T1 = 0.08        // Zero sequence reactance(p.u)
MVA_T5 = 30.0        // Transformer rating(MVA)
kV_lv_T5 = 13.8      // Transformer low voltage(kV)
kV_hv_T5 = 138       // Transformer high voltage(kV)
X_1_T5 = 0.10        // Positive sequence reactance(p.u)
X_2_T5 = 0.10        // Negative sequence reactance(p.u)
X_0_T5 = 0.08        // Zero sequence reactance(p.u)
X_neutral = 0.05      // Reactance of reactor connected to generator neutral(p.u)

// Calculations
MVA_base = 100.0     // Base MVA
kV_line = 138.0      // Base voltage for line(kV)
kV_G = 13.8
```

```
    // Base voltage for generator(kV)
X_st_G1_pu = %i*X_st_G1*(kV_G1/kV_G)**2*MVA_base/MVA_G1            // Impedance of G1 & G2(p.u)
X_2_G1_pu = %i*X_2_G1*(kV_G1/kV_G)**2*MVA_base/MVA_G1               // Impedance of G1 & G2(p.u)
X_g0_G1_pu = %i*X_0_G1*(kV_G1/kV_G)**2*MVA_base/MVA_G1             // Impedance of G1 & G2(p.u)
X_gn_G1_pu = %i*X_neutral*(kV_G1/kV_G)**2*MVA_base/MVA_G1         // Impedance of G1 & G2(p.u)
X_st_G3_pu = %i*X_st_G3*(kV_G3/kV_G)**2*MVA_base/MVA_G3            // Impedance of G3(p.u)
X_2_G3_pu = %i*X_2_G3*(kV_G3/kV_G)**2*MVA_base/MVA_G3               // Impedance of G3(p.u)
X_g0_G3_pu = %i*X_0_G3*(kV_G3/kV_G)**2*MVA_base/MVA_G3             // Impedance of G3(p.u)
X_gn_G3_pu = %i*X_neutral*(kV_G3/kV_G)**2*MVA_base/MVA_G3         // Impedance of G3(p.u)
X_1_T1_pu = %i*X_1_T1*MVA_base/MVA_T1                                  // Impedance of T1,T2,T3 & T4(p.u)
X_2_T1_pu = %i*X_2_T1*MVA_base/MVA_T1                                  // Impedance of T1,T2,T3 & T4(p.u)
X_0_T1_pu = %i*X_0_T1*MVA_base/MVA_T1                                  // Impedance of T1,T2,T3 & T4(p.u)
X_1_T5_pu = %i*X_1_T5*MVA_base/MVA_T5                                  // Impedance of T5 & T6(p.u)
X_2_T5_pu = %i*X_2_T5*MVA_base/MVA_T5                                  // Impedance of T5 & T6(p.u)
X_0_T5_pu = %i*X_0_T5*MVA_base/MVA_T5                                  // Impedance of T5 & T6(p.u)
X_1_line_20 = %i*20.0*100/kV_line**2                                    // Impedance of 20
```

```
ohm line(p.u)
X_2_line_20 = %i*20.0*100/kV_line**2                    // Impedance of 20 ohm line(p.u)
X_0_line_20 = 3.0*X_1_line_20                           // Impedance of 20 ohm line(p.u)
X_1_line_10 = %i*10.0*100/kV_line**2                    // Impedance of 10 ohm line(p.u)
X_2_line_10 = %i*10.0*100/kV_line**2                    // Impedance of 10 ohm line(p.u)
X_0_line_10 = 3.0*X_1_line_10                           // Impedance of 10 ohm line(p.u)
// Positive,negative and zero sequence network
Z_1_1 = X_1_T1_pu+X_1_T1_pu+X_1_line_20                 // Impedance(p.u)
Z_2_1 = X_1_T1_pu+X_1_T5_pu+X_1_line_10                 // Impedance(p.u)
Z_3_1 = X_1_T1_pu+X_1_T5_pu+X_1_line_10                 // Impedance(p.u)
Z_4_1 = Z_1_1*Z_2_1/(Z_1_1+Z_2_1+Z_3_1)                 // Impedance after star-delta transformation(p.u)
Z_5_1 = Z_3_1*Z_1_1/(Z_1_1+Z_2_1+Z_3_1)                 // Impedance after star-delta transformation(p.u)
Z_6_1 = Z_3_1*Z_2_1/(Z_1_1+Z_2_1+Z_3_1)                 // Impedance after star-delta transformation(p.u)
Z_7_1 = X_st_G1_pu+Z_4_1                                // Impedance(p.u)
Z_8_1 = X_st_G1_pu+Z_5_1                                // Impedance(p.u)
```

```
Z_9_1 = Z_7_1*Z_8_1/(Z_7_1+Z_8_1)                   // Impedance in parallel(p.u). Refer Fig E4.14(e) & E4.14(f)
Z_10_1 = Z_9_1+Z_6_1                                 // Impedance(p.u). Refer Fig E4.14(f) & E4.14(g)
Z_11_1 = Z_10_1*X_st_G3_pu/(Z_10_1+X_st_G3_pu)       // Impedance in parallel(p.u). Refer Fig E4.14(g) & E4.14(h)
Z_1 = Z_11_1                                         // Positive sequence impedance(p.u)
Z_2 = Z_1                                            // Negative sequence impedance(p.u)
Z_0 = X_g0_G3_pu+3.0*X_gn_G3_pu                      // Zero sequence impedance(p.u)
E_g = 1.0                                            // Voltage(p.u)
I_f_pu = 3*E_g/(Z_1+Z_2+Z_0)                         // L-G fault current(p.u)
I_f = abs(I_f_pu)*MVA_base*1000/(3**0.5*kV_G)        // Actual fault current(A)
MVA_fault = abs(I_f_pu)*MVA_base                     // Fault MVA

// Results
disp("PART III - EXAMPLE : 4.7 : SOLUTION :-")
printf("\nFault current for a L-G fault at C = %.f A\n", I_f)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 30.8 Fault current when a single phase to earth fault occurs

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.8 :
// Page number 518-519
clear ; clc ; close ; // Clear the work space and console

// Given data
kV_G = 11.0                    // Generator rating(kV)
X_1_G = %i*0.1                 // Positive sequence reactance of generator(p.u)
X_2_G = %i*0.1                 // Negative sequence reactance of generator(p.u)
X_0_G = %i*0.02                // Zero sequence reactance of generator(p.u)
Z = 1.0                        // Earthing resistor(ohm)
X_1_T1 = %i*0.1                // Positive sequence reactance of 2-winding transformer(p.u)
X_2_T1 = %i*0.1                // Negative sequence reactance of 2-winding transformer(p.u)
X_0_T1 = %i*0.1                // Zero sequence reactanc of 2-winding transformere(p.u)
X_1_T2_hv = %i*0.05            // Positive sequence reactance of hv 3-winding transformer(p.u)
X_2_T2_hv = %i*0.05            // Negative sequence reactance of hv 3-winding transformer(p.u)
```

```
X_0_T2_hv = %i*0.05          // Zero sequence reactanc of hv 3-winding transformere(p.u)
X_1_T2_lv_1 = %i*0.02        // Positive sequence reactance of lv 3-winding transformer(p.u)
X_2_T2_lv_1 = %i*0.02        // Negative sequence reactance of lv 3-winding transformer(p.u)
X_0_T2_lv_1 = %i*0.02        // Zero sequence reactanc of lv 3-winding transformere(p.u)
X_1_T2_lv_2 = %i*0.05        // Positive sequence reactance of lv 3-winding transformer(p.u)
X_2_T2_lv_2 = %i*0.05        // Negative sequence reactance of lv 3-winding transformer(p.u)
X_0_T2_lv_2 = %i*0.05        // Zero sequence reactanc of lv 3-winding transformere(p.u)

// Calculations
MVA_b = 10.0                                                    // Base MVA
kV_b = 11.0                                                     // Base voltage(kV)
Z_n = Z*MVA_b/kV_b**2                                           // Impedance(p.u)
Z_1 = X_1_G+X_1_T1+X_1_T2_hv+((X_1_T2_lv_1*X_1_T2_lv_2)/(X_1_T2_lv_1+X_1_T2_lv_2))          // Positive sequence impedance(p.u)
Z_2 = X_2_G+X_2_T1+X_2_T2_hv+((X_2_T2_lv_1*X_2_T2_lv_2)/(X_2_T2_lv_1+X_2_T2_lv_2))          // Negative sequence impedance(p.u)
Z_0 = ((X_0_T1+X_0_T2_hv)*X_0_T2_lv_2/(X_0_T1+X_0_T2_hv+X_0_T2_lv_2))+X_0_T2_lv_1+3*Z_n    // Zero sequence impedance(p.u)
E = 1.0                                                         // Voltage(p.u)
I_f_pu = 3*E/(Z_1+Z_2+Z_0)
```

```
// Fault current(p.u)
I_f = MVA_b*1000*abs(I_f_pu)/(3**0.5*kV_b)                              // Fault current(A)

// Results
disp("PART III - EXAMPLE : 4.8 : SOLUTION :-")
printf("\nFault current, I_f = %.f A\n", I_f)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 30.9 `Fault currents in the lines`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.9 :
// Page number 519
clear ; clc ; close ; // Clear the work space and console

// Given data
MVA_G = 10.0        // Generator rating(MVA)
kV_G = 11.0         // Generator rating(kV)
X_1_G = 27.0        // Positive sequence reactance of generator(p.u)
X_2_G = 9.0         // Negative sequence reactance of generator(p.u)
X_0_G = 4.5         // Zero sequence reactance of
```

```
generator(p.u)
X_1_L = 9.0         // Positive sequence reactance of line upto fault(p.u)
X_2_L = 9.0         // Negative sequence reactance of line upto fault(p.u)
X_0_L = 0           // Zero sequence reactance of line upto fault(p.u)

// Calculations
E_a = kV_G*1000/3**0.5              // Phase voltage(V)
Z_1 = %i*(X_1_G+X_1_L)              // Positive sequence reactance(p.u)
Z_2 = %i*(X_2_G+X_2_L)              // Negative sequence reactance(p.u)
I_b = %i*3**0.5*E_a/(Z_1+Z_2)       // Fault current in line b(p.u)
I_c = -I_b                          // Fault current in line c(p.u)

// Results
disp("PART III - EXAMPLE : 4.9 : SOLUTION :-")
printf("\nFault current in line b, I_b = %.f A", abs(I_b))
printf("\nFault current in line c, I_c = %.f A", real(I_c))
```

Scilab code Exa 30.10 Currents in the faulted phase Current through ground and Vol

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS
```

```
// EXAMPLE : 4.10 :
// Page number 519-520
clear ; clc ; close ; // Clear the work space and console

// Given data
MVA_A = 30.0          // Alternator rating(MVA)
kV_A = 11.0           // Alternator rating(kV)
X_1 = 2.5             // Reactance to positive sequence current(ohm)
X_2 = 0.8*X_1         // Reactance to negative sequence current(ohm)
X_0 = 0.3*X_1         // Reactance to zero sequence current(ohm)

// Calculations
// Case(a)
a = exp(%i*120.0*%pi/180)                    // Operator
Z_1 = %i*X_1                                 // Positive sequence impedance(ohm)
Z_2 = %i*X_2                                 // Negative sequence impedance(ohm)
Z_0 = %i*X_0                                 // Zero sequence impedance(ohm)
Z_02 = Z_0*Z_2/(Z_0+Z_2)                     // Impedance(ohm)
E_a = kV_A*1000/3**0.5                       // Phase voltage(V)
I_a1 = E_a/(Z_1+Z_02)                        // Positive sequence current(A)
I_a2 = -Z_0/(Z_0+Z_2)*I_a1                   // Negative sequence current(A)
I_a0 = -Z_2/(Z_0+Z_2)*I_a1                   // Zero sequence current(A)
I_0 = I_a0                                   // Zero sequence current(A)
```

```
I_a = I_a0+I_a1+I_a2                                // Line current(A)
I_b = I_0+a**2*I_a1+a*I_a2                          // Line current(A)
I_c = I_0+a*I_a1+a**2*I_a2                          // Line current(A)
// Case(b)
I_n = 3*abs(I_0)                                    // Current through ground(A)
// Case(c)
V_a2 = Z_02*I_a1                                    // Negative sequence voltage(V)
V_a = 3*abs(V_a2)                                   // Voltage of healthy phase to neutral(V)

// Results
disp("PART III - EXAMPLE : 4.10 : SOLUTION :-")
printf("\nCase(a): Currents in the faulted phase are ")
printf("\n          I_a = %.f A", abs(I_a))
printf("\n          I_b = %.f %.1f  A", abs(I_b), phasemag(I_b))
printf("\n          I_c = %.f %.1f  A", abs(I_c), phasemag(I_c))
printf("\nCase(b): Current through ground, I_n = %.f A", I_n)
printf("\nCase(c): Voltage of healthy phase to neutral, V_a = %.f V\n", V_a)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 30.11 Fault currents

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
```

```
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.11 :
// Page number 520-521
clear ; clc ; close ; // Clear the work space and console

// Given data
n = 6.0          // Number of alternator
kV_A = 6.6       // Alternator rating(kV)
X_1 = 0.9        // Positive sequence reactance(ohm)
X_2 = 0.72       // Negative sequence reactance(ohm)
X_0 = 0.3        // Zero sequence reactance(ohm)
Z_n = 0.2        // Resistance of grounding resistor(ohm)

// Calculations
E_a = kV_A*1000/3**0.5                   // Phase voltage(V)
// Case(a)
Z_1_a = %i*X_1/n                         // Positive sequence impedance when alternators are in parallel(ohm)
Z_2_a = %i*X_2/n                         // Negative sequence impedance when alternators are in parallel(ohm)
Z_0_a = %i*X_0/n                         // Zero sequence impedance when alternators are in parallel(ohm)
I_a_a = 3*E_a/(Z_1_a+Z_2_a+Z_0_a)        // Fault current assuming 'a' phase to be fault(A)
// Case(b)
Z_0_b = 3*Z_n+%i*X_0                     // Zero sequence impedance(ohm)
```

```
I_a_b = 3*E_a/(Z_1_a+Z_2_a+Z_0_b)      // Fault
    current(A)
// Case(c)
Z_0_c = %i*X_0                          // Zero
    sequence impedance(ohm)
I_a_c = 3*E_a/(Z_1_a+Z_2_a+Z_0_c)      // Fault
    current(A)

// Results
disp("PART III - EXAMPLE : 4.11 : SOLUTION :-")
printf("\nCase(a): Fault current if all alternator
    neutrals are solidly grounded, I_a = %.f A", imag
    (I_a_a))
printf("\nCase(b): Fault current if one alternator
    neutral is grounded & others isolated, I_a = %.1
     f % .1 f   A", abs(I_a_b),phasemag(I_a_b))
printf("\nCase(c): Fault current if one alternator
    neutral is solidly grounded & others isolated,
    I_a = %.2fj A\n", imag(I_a_c))
printf("\nNOTE: ERROR: Calculation mistakes in the
    textbook solution")
```

Scilab code Exa 30.12 Fault current if all 3 phases short circuited If single line

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.12 :
// Page number 521-522
clear ; clc ; close ; // Clear the work space and
```

```
console

// Given data
MVA_A = 30.0          // Alternator rating(MVA)
kV_A = 6.6            // Alternator rating(kV)
X_G = 10.0            // Reactance of alternator(%)
kV_lv_T = 6.6         // Transformer lv side rating(kV)
kV_hv_T = 33.0        // Transformer hv side rating(kV)
X_T = 6.0             // Reactance of transformer(%)
kV_line = 33.0        // Transmission line voltage(kV)
X_line = 4.0          // Transmission line reactance(ohm)
X_g2 = 70.0           // Negative sequence reactance is 70% of +ve sequence reactance of generator(%)

// Calculations
MVA_base = 30.0                           // Base MVA
kV_base = 6.6                             // Base kV
Z_base = kV_base**2/MVA_base              // Base impedance(ohm)
Z_g1 = %i*Z_base*X_G/100                  // Positive sequence impedance of alternator(ohm)
Z_T1 = %i*Z_base*X_T/100                  // Positive sequence impedance of transformer(ohm)
Z_L1 = %i*(kV_base/kV_line)**2*X_line     // Positive sequence impedance of transmission line(ohm)
Z_g2 = X_g2/100*Z_g1                      // Negative sequence impedance of alternator(ohm)
Z_T2 = %i*Z_base*X_T/100                  // Negative sequence impedance of transformer(ohm)
Z_T0 = %i*Z_base*X_T/100                  // Zero sequence impedance of transformer(ohm)
Z_L2 = Z_L1                               // Negative sequence impedance of transmission line(ohm)
Z_1 = Z_g1+Z_T1+Z_L1+Z_T1                 // Positive sequence impedance(ohm)
```

```
Z_2 = Z_g2+Z_T2+Z_L2+Z_T2                          // Negative sequence impedance(ohm)
Z_0 = Z_T0                                         // Zero sequence impedance(ohm)
E_a = kV_base*1000/3**0.5                          // Base voltage(V)
// Case(a)
I_sc = E_a/Z_1                                     // Fault current if all 3 phases short circuited(A)
// Case(b)
I_a = 3*E_a/(Z_1+Z_2+Z_0)                          // Fault current if single line is grounded assuming 'a' to be grounded(A)
// Case(c)
I_b = %i*3**0.5*E_a/(Z_1+Z_2)                      // Fault current for a short circuit between two lines(A)
I_c = -%i*3**0.5*E_a/(Z_1+Z_2)                     // Fault current for a short circuit between two lines(A)

// Results
disp("PART III - EXAMPLE : 4.12 : SOLUTION :-")
printf("\nCase(a): Fault current if all 3 phases short circuited, I_sc = %.f %.f  A", abs(I_sc),phasemag(I_sc))
printf("\nCase(b): Fault current if single line is grounded, I_a = %.fj A", imag(I_a))
printf("\nCase(c): Fault current for a short circuit between two lines, I_b = %.f A", real(I_b))
printf("\n         Fault current for a short circuit between two lines, I_c = %.f A\n", real(I_c))
printf("\nNOTE: ERROR: (1).Calculation mistake in Z_2 in the textbook solution")
printf("\n             (2).Transformer reactance is 6 percent, not 5 percent as in problem statement")
```

Scilab code Exa 30.13 Sub transient current in the faulty phase

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.13 :
// Page number 522
clear ; clc ; close ; // Clear the work space and
   console

// Given data
kV = 6.9            // Alternator rating(kV)
MVA = 10.0          // Alternator rating(MVA)
X_st = 0.15         // Sub-transient reactance(p.u)
X_2 = 0.15          // Negative sequence reactance(p.u
   )
X_0 = 0.05          // Zero sequence reactance(p.u)
X = 0.397           // Grounding reactor(ohm)

// Calculations
MVA_base = 10.0                         // Base MVA
kV_base = 6.9                           // Base kV
Z_base = kV_base**2/MVA_base            // Base
   impedance(ohm)
Z_n = X/Z_base                          // Grounding
    reactor(p.u)
Z_1 = %i*X_st                           // Positive
   sequence impedance(p.u)
Z_2 = %i*X_2                            // Negative
```

```
    sequence impedance(p.u)
Z_0 = %i*(X_0+3*Z_n)                        // Zero
    sequence impedance(p.u)
E_a = 1.0                                   // Phase
    voltage(p.u)
I_a_pu = 3*E_a/(Z_1+Z_2+Z_0)                // Sub−
    transient current in the faulty phase(p.u)
I_base = kV_base*1000/(3**0.5*Z_base)       // Base
    current(A)
I_a = abs(I_a_pu)*I_base                    // Sub−
    transient current in the faulty phase(A)

// Results
disp("PART III − EXAMPLE : 4.13 : SOLUTION :−")
printf("\nSub−transient current in the faulty phase,
     I_a = %.f A\n", I_a)
printf("\nNOTE: Changes in the obtained answer from
    that of textbook is due to more precision here")
```

Scilab code Exa 30.14 Initial symmetrical rms current in all phases of generator

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 4: UNSYMMETRICAL FAULTS IN POWER SYSTEMS

// EXAMPLE : 4.14 :
// Page number 522−523
clear ; clc ; close ; // Clear the work space and
    console

// Given data
```

```
kVA = 10000.0      // Generator rating(kVA)
kV = 13.8          // Generator rating(kV)
X_st = 10.0        // Sub-transient reactance(%)
X_2 = 10.0         // Negative sequence reactance(%)
X_0 = 5.0          // Zero sequence reactance(%)
X = 8.0            // Grounding reactor(%)
X_con = 6.0        // Reactance of reactor connecting generator & transformer(%)

// Calculations
a = exp(%i*120.0*%pi/180)                    // Operator
Z_1 = %i*(X_st+X_con)/100                    // Positive sequence impedance(p.u)
Z_2 = %i*(X_2+X_con)/100                     // Negative sequence impedance(p.u)
Z_0 = %i*X_con/100                           // Zero sequence impedance(p.u)
E_a = 1.0                                    // Phase voltage(p.u)
I_a1 = E_a/(Z_1+Z_2+Z_0)                     // Sub-transient current in the faulty phase(p.u)
I_A1 = %i*I_a1                               // Positive sequence current(p.u)
I_A2 = -%i*I_a1                              // Negative sequence current(p.u)
I_A = I_A1+I_A2                              // Initial symmetrical r.m.s current in phase a(p.u)
I_B1 = a**2*I_A1                             // Positive sequence current(p.u)
I_B2 = a*I_A2                                // Negative sequence current(p.u)
I_B = I_B1+I_B2                              // Initial symmetrical r.m.s current in phase b(p.u)
I_C1 = a*I_A1                                // Positive sequence current(p.u)
I_C2 = a**2*I_A2                             // Negative sequence current(p.u)
I_C = I_C1+I_C2                              // Initial
```

```
symmetrical r.m.s current in phase c(p.u)
I_base = kVA/(3**0.5*kV)                    // Base current(A)
I_A_amp = I_A*I_base                        // Initial symmetrical r.m.s current in phase a(p.u)
I_B_amp = I_B*I_base                        // Initial symmetrical r.m.s current in phase b(p.u)
I_C_amp = I_C*I_base                        // Initial symmetrical r.m.s current in phase c(p.u)

// Results
disp("PART III - EXAMPLE : 4.14 : SOLUTION :-")
printf("\nInitial symmetrical r.m.s current in all phases of generator are,")
printf("\n I_A = %.f A", abs(I_A_amp))
printf("\n I_B = %. f % . f   A", abs(I_B_amp), phasemag(I_B_amp))
printf("\n I_C = %. f % . f   A", abs(I_C_amp), phasemag(I_C_amp))
```

Chapter 32

CIRCUIT BREAKER

Scilab code Exa 32.1 Maximum restriking voltage Frequency of transient oscillation

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 6: CIRCUIT BREAKER

// EXAMPLE : 6.1 :
// Page number 545
clear ; clc ; close ; // Clear the work space and
    console

// Given data
f = 50.0           // Generator frequency(Hz)
kV = 7.5           // emf to neutral rms voltage(kV)
X = 4.0            // Reactance of generator &
    connected system(ohm)
C = 0.01*10**-6    // Distributed capacitance(F)

// Calculations
```

```
// Case(a)
v = 2**0.5*kV                                   // Active
    recovery voltage i.e phase to neutral(kV)
V_max_restrike = v*2                            // Maximum
    restriking voltage i.e phase to neutral(kV)
// Case(b)
L = X/(2.0*%pi*f)                               //
    Inductance(H)
f_n = 1/(2.0*%pi*(L*C)**0.5*1000)               // Frequency
     of transient oscillation(kHZ)
// Case(c)
t = 1.0/(2.0*f_n*1000)                          // Time(sec)
avg_rate = V_max_restrike/t                     // Average
    rate of rise of voltage upto first peak of
    oscillation(kV/s)

// Results
disp("PART III - EXAMPLE : 6.1 : SOLUTION :-")
printf("\nCase(a): Maximum re-striking voltage(phase
    -to-neutral) = %.1f kV", V_max_restrike)
printf("\nCase(b): Frequency of transient
    oscillation , f_n = %.1f kHz", f_n)
printf("\nCase(c): Average rate of rise of voltage
    upto first peak of oscillation = %.f kV/s \n",
    avg_rate)
printf("\nNOTE: Changes in the obtained answer from
    that of textbook is due to more approximation in
    the textbook")
```

Scilab code Exa 32.3 Rate of rise of restriking voltage

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION
```

```
// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 6: CIRCUIT BREAKER

// EXAMPLE : 6.3 :
// Page number 545-546
clear ; clc ; close ; // Clear the work space and console

// Given data
kV = 132.0        // Voltage(kV)
pf = 0.3          // Power factor of the fault
K3 = 0.95         // Recovery voltage was 0.95 of full line value
f_n = 16000.0     // Natural frequency of the restriking transient(Hz)

// Calculations
kV_phase = kV/3**0.5                    // System voltage(kV)
sin_phi = sind(acosd(pf))               // Sin
K2 = 1.0
v = K2*K3*kV/3**0.5*2**0.5*sin_phi      // Active recovery voltage(kV)
V_max_restrike = 2*v                    // Maximum restriking voltage(kV)
t = 1.0/(2.0*f_n)                       // Time(sec)
RRRV = V_max_restrike/(t*10**6)         // Rate of rise of restriking voltage(kV/ -sec)

// Results
disp("PART III - EXAMPLE : 6.3 : SOLUTION :-")
printf("\nRate of rise of restriking voltage, R.R.R.V = %.2f kV/ -sec", RRRV)
```

Scilab code Exa 32.5 Voltage across the pole of a CB and Resistance to be used acr

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 6: CIRCUIT BREAKER

// EXAMPLE : 6.5 :
// Page number 565
clear ; clc ; close ; // Clear the work space and
    console

// Given data
kV = 132.0          // Voltage(kV)
C = 0.01*10**-6     // Phase to ground capacitance(F)
L = 6.0             // Inductance(H)
i = 5.0             // Magnetizing current(A)

// Calculations
V_pros = i*(L/C)**0.5/1000      // Prospective value
    of voltage(kV)
R = 1.0/2*(L/C)**0.5/1000       // Resistance to be
    used across the contacts to eliminate the
    restriking voltage(k-ohm)

// Results
disp("PART III - EXAMPLE : 6.5 : SOLUTION :-")
printf("\nVoltage across the pole of a CB = %.1f kV"
    , V_pros)
printf("\nResistance to be used across the contacts
    to eliminate the restriking voltage, R = %.2f k-
    ohm\n", R)
printf("\nNOTE: ERROR: Unit of final answer R is k-
    ohm, not ohm as in the textbook solution")
```

Scilab code Exa 32.6 `Rated normal current Breaking current Making current and Shor`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 6: CIRCUIT BREAKER

// EXAMPLE : 6.6 :
// Page number 567
clear ; clc ; close ; // Clear the work space and console

// Given data
I = 1200.0       // Rated normal current(A)
MVA = 1500.0     // Rated MVA
kV = 33.0        // Voltage(kV)

// Calculations
I_breaking = MVA/(3**0.5*kV)    // Rated symmetrical breaking current(kA)
I_making = I_breaking*2.55      // Rated making current(kA)
I_short = I_breaking            // Short-time rating(kA)

// Results
disp("PART III - EXAMPLE : 6.6 : SOLUTION :-")
printf("\nRated normal current = %.f A", I)
printf("\nBreaking current = %.2f kA (rms)", I_breaking)
printf("\nMaking current = %.f kA", I_making)
```

```
printf("\nShort-time rating = %.2f kA for 3 secs",I_short)
```

Scilab code Exa 32.8 Sustained short circuit Initial symmetrical rms current Maxim

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 6: CIRCUIT BREAKER

// EXAMPLE : 6.8 :
// Page number 569
clear ; clc ; close ; // Clear the work space and console

// Given data
kVA = 7500.0      // Rated kVA
X_st = 9.0        // Sub-transient reactance(%)
X_t = 15.0        // Transient reactance(%)
X_d = 100.0       // Direct-axis reactance(%)
kV = 13.8         // Voltage(kV). Assumption

// Calculations
kVA_base = 7500.0                          // Base kVA
kVA_sc_sustained = kVA_base/X_d*100        // Sustained S.C kVA
I_sc_sustained = kVA_base/(3**0.5*kV)      // Sustained S.C current(A). rms
I_st = kVA*100/(X_st*3**0.5*kV)            // Initial symmetrical rms current in the breaker(A)
I_max_dc = 2**0.5*I_st                     // Maximum possible dc component of the short-circuit(A)
```

```
I_moment = 1.6*I_st                     // Momentary current rating of the breaker(A)
I_interrupt = 1.1*I_st                  // Current to be interrupted by the breaker(A)
I_kVA = 3**0.5*I_interrupt*kV           // Interrupting kVA

// Results
disp("PART III - EXAMPLE : 6.8 : SOLUTION :-")
printf("\nCase(a): Sustained short circuit KVA in the breaker = %.f kVA", kVA_sc_sustained)
printf("\n         Sustained short circuit current in the breaker = %.1f A (rms)", I_sc_sustained)
printf("\nCase(b): Initial symmetrical rms current in the breaker = %.f A (rms)", I_st)
printf("\nCase(c): Maximum possible dc component of the short-circuit in the breaker = %.f A", I_max_dc)
printf("\nCase(d): Momentary current rating of the breaker = %.f A (rms)", I_moment)
printf("\nCase(e): Current to be interrupted by the breaker = %.f A (rms)", I_interrupt)
printf("\nCase(f): Interrupting kVA = %.f kVA \n", I_kVA)
printf("\nNOTE: Changes in the obtained answer from that of textbook due to more approximation in textbook")
```

Chapter 33

PROTECTIVE RELAYS

Scilab code Exa 33.1 Time of operation of the relay

```
// A Textbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 7: PROTECTIVE RELAYS

// EXAMPLE : 7.1 :
// Page number 595-596
clear ; clc ; close ; // Clear the work space and
   console

// Given data
I_setting = 150.0    // Current setting of IDMT(%)
t_mult = 0.5         // Time multiplier setting
ratio_CT = 500.0/5   // CT ratio
CT_sec = 5.0         // Secondary turn
I_f = 6000.0         // Fault current

// Calculations
```

```
I_sec_fault = I_f/ratio_CT                      //
    Secondary fault current(A)
PSM = I_sec_fault/(CT_sec*I_setting/100)   // Plug
    setting multiplier
t = 3.15                                        // Time
    against this PSM(sec). From graph E7.1 in
    textbook page no 595
time_oper = t*t_mult                            //
    Operating time(sec)

// Results
disp("PART III - EXAMPLE : 7.1 : SOLUTION :-")
printf("\nTime of operation of the relay = %.3f sec"
    , time_oper)
```

Scilab code Exa 33.2 `Time of operation of the relay`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 7: PROTECTIVE RELAYS

// EXAMPLE : 7.2 :
// Page number 596
clear ; clc ; close ; // Clear the work space and
    console

// Given data
ratio = 525.0/1      // CT ratio
CT_sec = 1.0         // Secondary turn
t_mult = 0.3         // Time multiplier setting
I_f = 5250.0         // Fault current(A)
```

```

// Calculations
I_sec_fault = I_f/ratio               // Secondary fault current(A)
PSM = I_sec_fault/(1.25*CT_sec)       // Plug setting multiplier
t = 3.15                              // Time against this PSM(sec). From graph E7.1 in textbook page no 595
time_oper = t*t_mult                  // Operating time (sec)

// Results
disp("PART III - EXAMPLE : 7.2 : SOLUTION :-")
printf("\nTime of operation of the relay = %.3f sec", time_oper)
```

Scilab code Exa 33.3 Operating time of feeder relay Minimum plug setting of transf

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 7: PROTECTIVE RELAYS

// EXAMPLE : 7.3 :
// Page number 596
clear ; clc ; close ; // Clear the work space and console

// Given data
MVA = 20.0              // Transformer MVA
overload = 30.0         // Overload of transformer(%)
```

```
kV = 11.0                // Bus bar rating(kV)
CT_trans = 1000.0/5      // Transformer CT
CT_cb = 400.0/5          // Circuit breaker CT
ps = 125.0               // Plug setting(%)
ts = 0.3                 // Time setting
I_f = 5000.0             // Fault current(A)
t_margin = 0.5           // Discriminative time margin(sec)

// Calculations
I_sec_fault = I_f/CT_cb                              // Secondary fault current(A)
CT_cb_sec = 5.0                                      // Secondary turn
PSM = I_sec_fault/(ps/100*CT_cb_sec)                 // Plug setting multiplier
t = 2.8                                              // Time against this PSM(sec). From graph E7.1 in textbook page no 595
time_oper = t*ts                                     // Operating time of feeder relay(sec)
I_ol = (1+(overload/100))*MVA*1000/(3**0.5*kV)       // Overload current(A)
I_sec_T = I_ol/CT_trans                              // Secondary current(A)
CT_T_sec = 5.0                                       // Secondary turn of transformer
PSM_T = I_sec_T/CT_T_sec                             // Minimum plug setting multiplier of transformer
I_sec_T1 = I_f/CT_trans                              // Secondary fault current(A)
ps_T1 = 1.5                                          // Plug setting as per standard value
PSM_T1 = I_sec_T1/(CT_T_sec*ps)                      // Plug setting multiplier of transformer
t_T1 = 7.0                                           // Time against this PSM(sec). From graph E7.1 in textbook page no 595
```

```
time_setting = (time_oper+t_margin)/t_T1          //
   Time setting of transformer

// Results
disp("PART III - EXAMPLE : 7.3 : SOLUTION :-")
printf("\nOperating time of feeder relay = %.2f sec"
   , time_oper)
printf("\nMinimum plug setting of transformer relay,
    P.S > %.2f ", PSM_T)
printf("\nTime setting of transformer = %.3f ",
   time_setting)
```

Scilab code Exa 33.4 Time of operation of the two relays

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 7: PROTECTIVE RELAYS

// EXAMPLE : 7.4 :
// Page number 596-597
clear ; clc ; close ; // Clear the work space and
   console

// Given data
I_f = 2000.0            // Fault current(A)
ratio_CT = 200.0/1      // CT ratio
R_1 = 100.0             // Relay 1 set on(%)
R_2 = 125.0             // Relay 2 set on(%)
t_margin = 0.5          // Discriminative time margin(
   sec)
TSM_1 = 0.2             // Time setting multiplier of
```

```
relay 1

// Calculations
CT_sec = 200.0                           // CT secondary
PSM_1 = I_f*100/(CT_sec*R_1)             // PSM of relay 1
t_1 = 2.8                                // Time against this PSM(sec). From graph E7.1 in textbook page no 595
time_oper_1 = TSM_1*t_1                  // Operating time of relay with TSM of 0.2(Sec)
PSM_2 = I_f*100/(CT_sec*R_2)             // PSM of relay 2
t_2 = 3.15                               // Time against this PSM(sec). From graph E7.1 in textbook page no 595
actual_time_2 = time_oper_1+t_margin     // Actual time of operation of relay 2(sec)
TSM_2 = actual_time_2/t_2                // Time setting multiplier of relay 2

// Results
disp("PART III - EXAMPLE : 7.4 : SOLUTION :-")
printf("\nTime of operation of relay 1 = %.2f sec", time_oper_1)
printf("\nActual time of operation of relay 2 = %.2f sec", actual_time_2)
printf("\nT.S.M of relay 2 = %.4f", TSM_2)
```

Scilab code Exa 33.6 Will the relay operate the trip of the breaker

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
```

```
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 7: PROTECTIVE RELAYS

// EXAMPLE : 7.6 :
// Page number 611
clear ; clc ; close ; // Clear the work space and console

// Given data
I_min = 0.1            // Relay minimum pick up current(A)
slope = 10.0           // Slope characteristic(%)
CT_ratio = 400.0/5     // CT ratio
I_1 = 320.0            // Current(A)
I_2 = 304.0            // Current(A)

// Calculations
I_op_coil = (I_1-I_2)/CT_ratio              // Current in operating coil(A)
I_re_coil = 1.0*(I_1+I_2)/(2*CT_ratio)      // Current in restraining coil(A)
I_re_coil_slope = I_re_coil*slope/100       // Current in restraining coil with slope(A)

// Results
disp("PART III - EXAMPLE : 7.6 : SOLUTION :-")
if(I_op_coil<I_re_coil_slope) then
    printf("\nRelay will not trip the circuit breaker")
else then
    print("\nRelay will trip the circuit breaker")
end
```

Chapter 34

PROTECTION OF ALTERNATORS AND AC MOTORS

Scilab code Exa 34.1 Neutral earthing reactance

```
// A Textbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 8: PROTECTION OF ALTERNATORS AND AC
   MOTORS

// EXAMPLE : 8.1 :
// Page number 624
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 6600.0              // Alternator Voltage(V)
P = 2000.0*10**3        // Rating of alternator(W)
```

```
PF = 0.8                // Power factor of alternator
X = 12.5                // Alternator reactance(%)
I = 200.0               // Current protection(A)
per = 10.0              // Percentage of winding unprotected(%)

// Calculations
I_fl = P/(3**0.5*V*PF)              // Full load current of alternator(A)
x = X*V/(3**0.5*100*I_fl)           // Reactance per phase of alternator(ohm)
x_per = per/100*x                   // Reactance of 10% of the winding(ohm)
NA = V/(3**0.5*per)                 // Voltage induced in winding(V)
r = ((NA/I)**2-x_per**2)**0.5       // Neutral earthing reactance(ohm)

// Results
disp("PART III - EXAMPLE : 8.1 : SOLUTION :-")
printf("\nNeutral earthing reactance, r = %.2f ohm", r)
```

Scilab code Exa 34.2 Unprotected portion of each phase of the stator winding again

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 8: PROTECTION OF ALTERNATORS AND AC MOTORS

// EXAMPLE : 8.2 :
```

```
// Page number 624-625
clear ; clc ; close ; // Clear the work space and console

// Given data
MVA = 20.0              // Generator rating (MVA)
V = 11.0*10**3          // Generator voltage (V)
ratio_CT = 1200.0/5     // Ratio of current transformer
I_min_op = 0.75         // Minimum operating current of relay (A)
R = 6.0                 // Neutral point earthing resistance (ohm)

// Calculations
I_max_fault = ratio_CT*I_min_op         // Maximum fault current to operate relay (A)
x = I_max_fault*3**0.5*100*R/V          // Unprotected portion for R = 6 ohm (%)
R_1 = 3.0                               // Neutral point earthing resistance (ohm)
x_1 = I_max_fault*3**0.5*100*R_1/V      // Unprotected portion for R = 3 ohm (%)
R_3 = 12.0                              // Neutral point earthing resistance (ohm)
x_3 = I_max_fault*3**0.5*100*R_3/V      // Unprotected portion for R = 12 ohm (%)

// Results
disp("PART III - EXAMPLE : 8.2 : SOLUTION :-")
printf("\nUnprotected portion of each phase of the stator winding against earth fault , x = %.f percent", x)
printf("\nEffect of varying neutral earthing resistance keeping relay operating current the same :")
printf("\n (i)    R = 3 ohms")
printf("\n        Unprotected portion = %.1f percent"
```

```
, x_1)
printf("\n        Protected portion = %.1f percent", (100-x_1))
printf("\n (ii)  R = 6 ohms")
printf("\n        Unprotected portion = %.f percent", x)
printf("\n        Protected portion = %.f percent", (100-x))
printf("\n (iii) R = 12 ohms")
printf("\n        Unprotected portion = %.f percent", x_3)
printf("\n        Protected portion = %.f percent", (100-x_3))
```

Scilab code Exa 34.3 Portion of alternator winding unprotected

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 8: PROTECTION OF ALTERNATORS AND AC MOTORS

// EXAMPLE : 8.3 :
// Page number 625
clear ; clc ; close ; // Clear the work space and console

// Given data
kVA = 5000.0       // Alternator rating(kVA)
V = 6600.0         // Alternator voltage(V)
X = 2.0            // Synchronous reactance per phase (ohm)
```

```
R = 0.5              // Resistance(ohm)
ofb = 30.0           // Out-of-balance current(%)
R_n = 6.5            // Resistance of resistor earthed
    to star point(ohm)

// Calculations
I_fl = kVA*1000/(3**0.5*V)                   // Full
    load current(A)
I_ofb = ofb/100*I_fl                         // Out-of
    -balance current(A)
x = R_n/((V/(3**0.5*100*I_ofb))-(R/100))     //
    Portion of winding unprotected(%)

// Results
disp("PART III - EXAMPLE : 8.3 : SOLUTION :-")
printf("\nPortion of alternator winding unprotected,
     x = %.1f percent", x)
```

Scilab code Exa 34.4 `Will the relay trip the generator CB`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 8: PROTECTION OF ALTERNATORS AND AC
    MOTORS

// EXAMPLE : 8.4 :
// Page number 625
clear ; clc ; close ; // Clear the work space and
    console

// Given data
```

```
I_min = 0.15          // Minimum pick up current of relay(A)
slope = 12.0          // Slope(%)
CT_ratio = 400.0/5    // CT ratio
I_1 = 360.0           // Current(A)
I_2 = 300.0           // Current(A)

// Calculations
i_1 = I_1/CT_ratio                           // Current(A)
i_2 = I_2/CT_ratio                           // Current(A)
percentage = (i_1-i_2)/((i_1+i_2)/2)*100     // Percentage(%)

// Results
disp("PART III - EXAMPLE : 8.4 : SOLUTION :-")
if(percentage>slope) then
    printf("\nRelay would trip the circuit breaker, since the point lie on +ve torque regime")
else then
    printf("\nRelay would not trip the circuit breaker, since the point do not lie on +ve torque regime")
end
```

Scilab code Exa 34.5 Winding of each phase unprotected against earth when machine

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 8: PROTECTION OF ALTERNATORS AND AC
```

```
    MOTORS

// EXAMPLE : 8.5 :
// Page number 625-626
clear ; clc ; close ; // Clear the work space and
    console

// Given data
MVA = 50.0              // Alternator rating(MVA)
V = 33.0*10**3          // Alternator voltage(V)
CT_ratio = 2000.0/5     // CT ratio
R = 7.5                 // Resistor earthed generator
    neutral(ohm)
I = 0.5                 // Current above which pick up
    current(A)

// Calculations
I_min = CT_ratio*I              // Minimum current
    required to operate relay(A)
x = I_min*R/(V/3**0.5)*100     // Winding unprotected
    during normal operation(%)

// Results
disp("PART III - EXAMPLE : 8.5 : SOLUTION :-")
printf("\nWinding of each phase unprotected against
    earth when machine operates at nominal voltage, x
     = %.2f percent", x)
```

Scilab code Exa 34.6 `Portion of winding unprotected`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 8: PROTECTION OF ALTERNATORS AND AC MOTORS

// EXAMPLE : 8.6 :
// Page number 626
clear ; clc ; close ; // Clear the work space and console

// Given data
MVA = 50.0          // Alternator rating(MVA)
kV = 11.0           // Alternator voltage(kV)
X = 2.0             // Synchronous reactance per phase(ohm)
R = 0.7             // Resistance per phase(ohm)
R_n = 5.0           // Resistance through which alternator is earthed(ohm)
ofb = 25.0          // Out-of-balance current(%)

// Calculations
I_fl = MVA*1000/(3**0.5*kV)                          // Full load current(A)
I_ofb = ofb/100*I_fl                                 // Out-of-balance current(A)
x = R_n/((kV*1000/(3**0.5*100*I_ofb))-(R/100))       // Portion of winding unprotected(%)

// Results
disp("PART III - EXAMPLE : 8.6 : SOLUTION :-")
printf("\nPortion of winding unprotected, x = %.f percent", x)
```

Scilab code Exa 34.7 Percentage of winding that is protected against earth faults

```
// A Texbook on POWER SYSTEM ENGINEERING
```

```
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 8: PROTECTION OF ALTERNATORS AND AC
   MOTORS

// EXAMPLE : 8.7 :
// Page number 626-627
clear ; clc ; close ; // Clear the work space and
   console

// Given data
kV = 11.0          // Alternator voltage(kV)
MVA = 5.0          // Alternator rating(MVA)
X = 2.0            // Reactance per phase(ohm)
ofb = 35.0         // Out-of-balance current(%)
R_n = 5.0          // Resistance through which star
   point is earthed(ohm)

// Calculations
I_fl = MVA*1000/(3**0.5*kV)                // Full
   load current(A)
I_ofb = ofb/100*I_fl                       // Out-of-
   balance current(A)
x = I_ofb*R_n*100/(kV*1000/3**0.5)         // Portion
   of winding unprotected(%)
protected = 100.0-x                        // Winding
   that is protected against earth faults(%)

// Results
disp("PART III - EXAMPLE : 8.7 : SOLUTION :-")
printf("\nPercentage of winding that is protected
   against earth faults = %.2f percent", protected)
```

Scilab code Exa 34.8 Magnitude of neutral earthing resistance

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 8: PROTECTION OF ALTERNATORS AND AC MOTORS

// EXAMPLE : 8.8 :
// Page number 627
clear ; clc ; close ; // Clear the work space and console

// Given data
kV = 11.0          // Alternator voltage(kV)
P = 100.0          // Alternator maximum rating(MW)
PF = 0.8           // Power factor
X = 0.1            // Reactance of alternator(pu)
i = 500.0          // Current(A)
per = 10.0         // Windings unprotected(%)

// Calculations
I = P*1000/(3**0.5*kV*PF)        // Rated current of alternator(A)
a = i/I                          // Relay setting
I_n = a*I*100/per                // Current through neutral(A)
R = kV*1000/(3**0.5*I_n)         // Magnitude of neutral earthing resistance(ohm)

// Results
```

```
disp("PART III - EXAMPLE : 8.8 : SOLUTION :-")
printf("\nMagnitude of neutral earthing resistance ,
    R = %.2f ohm\n", R)
printf("\nNOTE: ERROR: Unit of resistance is not
    mentioned in textbook solution")
```

Chapter 35

PROTECTION OF TRANSFORMERS

Scilab code Exa 35.2 Ratio of CTs

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 9: PROTECTION OF TRANSFORMERS

// EXAMPLE : 9.2 :
// Page number 635-636
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V_lv = 220.0                    // LV side voltage of
    transformer(V)
V_hv = 11000.0                  // HV side voltage of
    transformer(V)
ratio_CT = 600.0/(5/3**0.5)   // CT ratio on LV side
```

```
    of transformer

// Calculations
CT_pri = 600.0              // Primary CT
CT_sec = 5.0/3**0.5         // Secondary CT
I_1 = V_lv/V_hv*CT_pri      // Line current in
    secondary of transformer corresponding to primary
     winding(A)
I_2 = CT_sec*3**0.5         // Current in secondary of
   CT(A)

// Results
disp("PART III - EXAMPLE : 9.2 : SOLUTION :-")
printf("\nRatio of CTs on 11000 V side = %.f : %.f \
   n", I_1,I_2)
printf("\nNOTE: ERROR: Mistake in representing the
    final answer in textbook solution")
```

Scilab code Exa 35.3 Ratio of CTs on high voltage side

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 9: PROTECTION OF TRANSFORMERS

// EXAMPLE : 9.3 :
// Page number 636
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V_lv = 11.0*10**3          // LV side voltage of
```

```
    transformer(V)
V_hv = 66.0*10**3         // HV side voltage of
    transformer(V)
ratio_CT = 250.0/5        // CT ratio on LV side of
    transformer

// Calculations
V_hv_phase = V_hv/3**0.5           // HV side phase
    voltage(V)
ratio_main_T = V_hv_phase/V_lv     // Ratio of main
    transformer
I_2 = 250.0                        // Primary CT
I_1 = I_2/(ratio_main_T*3**0.5)    // Primary line
    current(A)
CT_sec = 5.0                       // Secondary CT
secondary_side = CT_sec/3**0.5     // HV side CT
    secondary

// Results
disp("PART III - EXAMPLE : 9.3 : SOLUTION :-")
printf("\nRatio of CTs on high voltage side = %.1f :
    %.1f = (%.f/%.2f 3 ) : (%.f/ 3 ) ", I_1,
    secondary_side,I_2,ratio_main_T,CT_sec)
```

Scilab code Exa 35.4 Ratio of protective CTs

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 9: PROTECTION OF TRANSFORMERS

// EXAMPLE : 9.4 :
```

```
// Page number 636
clear ; clc ; close ; // Clear the work space and console

// Given data
V_hv = 33.0             // HV side voltage of transformer(kV)
V_lv = 6.6              // LV side voltage of transformer(kV)
ratio_CT = 100.0/1      // CT ratio on LV side of transformer

// Calculations
CT_pri = 100.0                  // Primary CT
CT_sec = 1.0                    // Secondary CT
I_hv = V_lv/V_hv*CT_pri         // Line current on HV side(A)
I_lv = CT_sec/3**0.5            // Line current on LV side(A)

// Results
disp("PART III - EXAMPLE : 9.4 : SOLUTION :-")
printf("\nRatio of protective CTs on 33 kV side = %.f : %.f/ 3 = %.f : %.f ", I_hv,CT_sec,3**0.5*I_hv,I_lv*3**0.5)
```

Scilab code Exa 35.5 CT ratios on high voltage side

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 9: PROTECTION OF TRANSFORMERS
```

```
// EXAMPLE : 9.5 :
// Page number 636-637
clear ; clc ; close ; // Clear the work space and
    console

// Given data
kVA = 200.0              // Transformer rating(kVA)
E_1 = 11000.0            // HV side voltage of
    transformer(kV)
E_2 = 400.0              // LV side voltage of
    transformer(kV)
ratio_CT = 500.0/5       // CT ratio on LV side of
    transformer
I_f = 750.0              // Fault current(A)

// Calculations
I_2 = 500.0                        // Primary CT
I_1 = 5.0                          // Secondary CT
I_1_T = E_2*I_2/(3**0.5*E_1)       // Primary current in
    transformer(A)
I_hv_T = I_1_T*3**0.5              // Equivalent line
    current on HV side(A)
I_pilot_lv = I_1*3**0.5            // Pilot current on LV
     side(A)

// Results
disp("PART III - EXAMPLE : 9.5 : SOLUTION :-")
printf("\nCT ratios on high voltage side = %.2f : %
    .2f \n", I_hv_T,I_pilot_lv)
printf("\nNOTE: Circulating current is not
    calculated")
```

Scilab code Exa 35.6 Suitable CT ratios

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 9: PROTECTION OF TRANSFORMERS

// EXAMPLE : 9.6 :
// Page number 640
clear ; clc ; close ; // Clear the work space and
    console

// Given data
MVA = 50.0          // Transformer rating(MVA)
V_hv = 132.0        // HV side voltage of transformer(
    kV)
V_lv = 33.0         // LV side voltage of transformer(
    kV)
CT_sec = 1.0        // Secondary CT rating

// Calculations
I_FL = MVA*1000/(3**0.5*V_lv)
    // Full-load current(A)
CT_ratio_33kV = I_FL/CT_sec
    // CT ratio on 33 kV side
CT_ratio_132kV = (I_FL*V_lv/V_hv)/(CT_sec/3**0.5)
    // CT ratio on 132 kV side

// Results
disp("PART III - EXAMPLE : 9.6 : SOLUTION :-")
printf("\nCT ratio on 33 kV side = %.f : 1 ",
    CT_ratio_33kV)
printf("\nCT ratio on 132 kV side = %.f : 1 = %.
    f 3  : 1 ", CT_ratio_132kV,CT_ratio_132kV
    /3**0.5)
```

Chapter 36

PROTECTION OF TRANSMISSION LINE SHUNT INDUCTORS AND CAPACITORS

Scilab code Exa 36.1 `First Second and Third zone relay setting Without infeed and`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 10: PROTECTION OF TRANSMISSION LINE, SHUNT INDUCTORS AND CAPACITORS

// EXAMPLE : 10.1 :
// Page number 647-648
clear ; clc ; close ; // Clear the work space and console

// Given data
```

```
G2_per = 70.0                  // G2 is fed at 70% distance from A in section AB(%)
X_T = 10.0                     // Transformer reactance(%)
zone_1_per = 80.0              // Setting for first zone(%)
zone_2_per = 50.0              // Setting for second zone(%)
CT_ratio = 400.0/5             // CT ratio
PT_ratio = 166000.0/110        // PT ratio
Z_AB = complex(20.0,60.0)      // Section AB impedance(ohm)
Z_BC = complex(10.0,25.0)      // Section BC impedance(ohm)
MVA = 10.0                     // Transformer rating(MVA)
kV_hv = 166.0                  // HV side voltage(kV)
kV_lv = 33.0                   // LV side voltage(kV)

// Calculations
// Case(i) Without infeed
Z_sec_1 = zone_1_per/100*Z_AB*CT_ratio/PT_ratio    // First zone setting(ohm)
Z_BC_hv = Z_BC*(kV_hv/kV_lv)**2    // Z_BC on 166 kV base(ohm)
Z_T = %i*10*X_T*kV_hv**2/(MVA*1000)    // Transformer impedance(ohm)
Z_sec_2 = (Z_AB+zone_2_per/100*Z_BC_hv+Z_T)*CT_ratio/PT_ratio    // Second zone setting(ohm)
Z_sec_3 = (Z_AB+Z_BC_hv+Z_T)*CT_ratio/PT_ratio    // Third zone setting(ohm)
// Case(ii) With infeed
I_AB = 2.0    // Current ratio
Z_zone_1 = (G2_per/100*Z_AB)+I_AB*(zone_1_per-G2_per
```

```
)/100*Z_AB                                    // First zone impedance(ohm)
Z_1 = Z_zone_1*CT_ratio/PT_ratio              // First zone setting(ohm)
Z_zone_2 = (G2_per/100*Z_AB)+I_AB*(((zone_1_per-zone_2_per)/100*Z_AB)+(zone_2_per/100*Z_BC_hv)+Z_T)   // Second zone impedance(ohm)
Z_2 = Z_zone_2*CT_ratio/PT_ratio              // Second zone setting(ohm)
under_reach = Z_zone_2-(Z_AB+zone_2_per/100*Z_BC_hv+Z_T)                // Under-reach due to infeed(ohm)
Z_zone_3 = (G2_per/100*Z_AB)+I_AB*(((zone_1_per-zone_2_per)/100*Z_AB)+Z_BC_hv+Z_T)                  // Third zone impedance(ohm)
Z_3 = Z_zone_3*CT_ratio/PT_ratio              // Third zone setting(ohm)

// Results
disp("PART III - EXAMPLE : 10.1 : SOLUTION :-")
printf("\nCase(i)  Without infeed:")
printf("\n          First zone relay setting = (%.2f + %.2fj) ohm", real(Z_sec_1),imag(Z_sec_1))
printf("\n          Second zone relay setting = (%.1f + %.1fj) ohm", real(Z_sec_2),imag(Z_sec_2))
printf("\n          Third zone relay setting = (%.1f + %.1fj) ohm", real(Z_sec_3),imag(Z_sec_3))
printf("\nCase(ii) With infeed:")
printf("\n          First zone relay setting = (%.3f + %.2fj) ohm", real(Z_1),imag(Z_1))
printf("\n          Second zone relay setting = (%.1f + %.1fj) ohm", real(Z_2),imag(Z_2))
printf("\n          Third zone relay setting = (%.1f + %.fj) ohm\n", real(Z_3),imag(Z_3))
printf("\nNOTE: ERROR: Calculation mistake in Z_BC.
```

```
Hence, changes in the obtained answer from that
of textbook")
```

Scilab code Exa 36.2 Impedance seen by relay and Relay setting for high speed back

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART III : SWITCHGEAR AND PROTECTION
// CHAPTER 10: PROTECTION OF TRANSMISSION LINE,
   SHUNT INDUCTORS AND CAPACITORS

// EXAMPLE : 10.2 :
// Page number 648
clear ; clc ; close ; // Clear the work space and
   console

// Given data
CT_ratio = 300.0/5            // CT ratio
PT_ratio = 166000.0/110       // PT ratio
Z_AB = complex(40.0,160.0)    // Section AB impedance(
   ohm)
Z_BC = complex(7.5,15.0)      // Section BC impedance(
   ohm)
kV_hv = 166.0                 // HV side voltage(kV)
kV_lv = 33.0                  // LV side voltage(kV)
MVA = 5.0                     // Transformer rating(
   MVA)
X_T = 6.04                    // Transformer reactance
   (%)

// Calculations
Z_T = %i*10*X_T*kV_hv**2/(MVA*1000)    // Tranformer
```

```
impedance(ohm)
Z_fault = Z_AB+Z_T                       // Fault impedance(ohm)
Z_sec = Z_fault*CT_ratio/PT_ratio        // Relay setting for primary protection(ohm)
Z_BC_hv = Z_BC*(kV_hv/kV_lv)**2          // Z_BC on 166 kV base(ohm)
Z = Z_AB+Z_T+Z_BC_hv                     // For backup protection of line BC(ohm)
Z_sec_set = Z*CT_ratio/PT_ratio          // Relay setting(ohm)

// Results
disp("PART III - EXAMPLE : 10.2 : SOLUTION :-")
printf("\nImpedance seen by relay = (%.f + %.fj) ohm", real(Z_fault),imag(Z_fault))
printf("\nRelay setting for high speed & backup protection = (%.1f + %.2fj) ohm", real(Z_sec_set),imag(Z_sec_set))
```

Chapter 39

INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

Scilab code Exa 39.1 Total annual cost of group drive and Individual drive

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
   MOTORS

// EXAMPLE : 1.1 :
// Page number 676
clear ; clc ; close ; // Clear the work space and
   console

// Given data
capital_cost_group = 8000.0        // Capital cost of
    group drive(Rs)
```

```
n_single = 5.0                          // Number of individual drive
capital_cost_single = 2500.0            // Capital cost of  individual drive(Rs)
energy_cons_group = 40000.0             // Annual energy consumption of group drive(kWh)
energy_cons_single = 30000.0            // Annual energy consumption of group drive(kWh)
cost_energy = 8.0/100                   // Cost of energy per kWh(Rs)
dmo_group = 12.0                        // Depreciation, maintenance & other fixed charges for group drive(%)
dmo_single = 18.0                       // Depreciation, maintenance & other fixed charges for individual drive(%)

// Calculations
// Case(a)
annual_cost_energy_a = energy_cons_group*cost_energy      // Annual cost of energy(Rs)
dmo_cost_a = capital_cost_group*dmo_group/100             // Depreciation, maintenance & other fixed charges per year for group drive(Rs)
yearly_cost_a = annual_cost_energy_a+dmo_cost_a           // Total yearly cost(Rs)
// Case(b)
total_cost = capital_cost_single*n_single                 // Capital cost of individual drive(Rs)
annual_cost_energy_b = energy_cons_single*cost_energy  // Annual cost of energy(Rs)
dmo_cost_b = total_cost*dmo_single/100                    // Depreciation, maintenance & other fixed charges per year for individual drive(Rs)
yearly_cost_b = annual_cost_energy_b+dmo_cost_b           // Total yearly cost(Rs)
```

```
// Results
disp("PART IV - EXAMPLE : 1.1 : SOLUTION :-")
printf("\nTotal annual cost of group drive = Rs. %.f
    ", yearly_cost_a)
printf("\nTotal annual cost of individual drive = Rs
    . %.f ", yearly_cost_b)
```

Scilab code Exa 39.2 Starting torque in terms of full load torque with star delta

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
    MOTORS

// EXAMPLE : 1.2 :
// Page number 680
clear ; clc ; close ; // Clear the work space and
    console

// Given data
I_sc = 6.0      // Short circuit current = 6 times
    full load current
s_fl = 5.0      // Full load slip(%)
tap = 60.0      // Auto-tranformer tapping(%)

// Calculations
// Case(a)
I_s_fl_a = I_sc/3.0                    // I_s/I_fl
T_s_fl_a = I_s_fl_a**2*s_fl/100        // Starting
    torque in terms of full-load torque with star-
```

```
    delta starter
// Case(b)
I_s_fl_b = tap/100*I_sc              // I_s/I_fl
T_s_fl_b = I_s_fl_b**2*s_fl/100      // Starting torque in terms of full-load torque with auto-transformer starter

// Results
disp("PART IV - EXAMPLE : 1.2 : SOLUTION :-")
printf("\nCase(a): Starting torque in terms of full-load torque with star-delta starter, I_s/I_fl = %.1f ", T_s_fl_a)
printf("\nCase(b): Starting torque in terms of full-load torque with auto-transformer starter, I_s/I_fl = %.3f ", T_s_fl_b)
```

Scilab code Exa 39.3 Tapping to be provided on an auto transformer Starting torque

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.3 :
// Page number 680-681
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 400.0       // IM voltage(V)
s_fl = 5.0      // Full-load slip(%)
```

```
I_fl = 20.0     // Full load current drawn from supply by IM(A)
Z = 2.5         // Impedance per phase(ohm)
I_max = 50.0    // Maximum current drawn(A)

// Calculations
V_phase = V/3**0.5                       // Normal phase voltage(V)
P = (100**2*I_max*Z/V_phase)**0.5        // Tapping to be provided to auto-transformer(%)
I_s = I_max/(P/100)                      // Starting current taken by motor(A)
T_s_fl = (I_s/I_fl)**2*s_fl/100          // Starting torque in terms of full-load torque
T_s_fl_R = (I_max/I_fl)**2*s_fl/100  // Starting torque in terms of full-load torque when a resistor is used

// Results
disp("PART IV - EXAMPLE : 1.3 : SOLUTION :-")
printf("\nTapping to be provided on an auto-transformer, P = %.1f percent", P)
printf("\nStarting torque in terms of full-load torque, T_s = %.3f*T_fl ", T_s_fl)
printf("\nStarting torque in terms of full-load torque if a resistor were used in series, T_s = %.4f*T_fl ", T_s_fl_R)
```

Scilab code Exa 39.4 Starting torque and Starting current if motor started by Dire

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

```

```
// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
   MOTORS

// EXAMPLE : 1.4 :
// Page number 681-682
clear ; clc ; close ; // Clear the work space and
   console

// Given data
hp = 30.0        // Power of cage IM(hp)
V = 500.0        // Cage IM voltage(V)
P = 4.0          // Number of poles
f = 50.0         // Frequency(Hz)
I_fl = 33.0      // Full load current(A)
s = 4.0/100      // Slip
Z = 3.5          // Impedance per phase(ohm)
tap = 60.0       // Auto-transformer tap setting(%)

// Calculations
// Case(1)
I_s_1 = 3**0.5*(V/Z)                          //
   Starting current taken from line(A)
N_s = 120*f/P                                 // Speed(
   rpm)
N_fl = N_s-N_s*s                              // Full
   load speed of motor(rpm)
T_fl = hp*746*60/(2*%pi*N_fl)                 // Full
   load torque(N-m)
T_s_1 = (I_s_1/I_fl)**2*s*T_fl                //
   Starting torque(N-m)
// Case(2)
V_ph = V/3**0.5                               // Phase
   voltage in star(V)
I_s_2 = V_ph/Z                                //
   Starting current(A/phase)
T_s_2 = (I_s_2/(I_fl/3**0.5))**2*s*T_fl       //
   Starting torque(N-m)
```

```
// Case(3)
V_ph_at = V*tap/(3**0.5*100)                    // Phase voltage of auto-transformer secondary(V)
V_impressed = V_ph_at*3**0.5                    // Volatage impressed on delta-connected stator(V)
I_s_3 = V_impressed/Z                           // Starting current(A/phase)
I_s_line = 3**0.5*I_s_3                         // Motor starting line current from auto-transformer secondary(A)
I_s_line_3 = tap/100*I_s_line                   // Starting current taken from supply(A)
T_s_3 = (I_s_3/(I_fl/3**0.5))**2*s*T_fl         // Starting torque(N-m)
// Case(4)
I_s_4 = 3**0.5*V/Z                              // Starting current from line(A)
T_s_4 = T_fl*s*(I_s_4/I_fl)**2                  // Starting torque(N-m)

// Results
disp("PART IV - EXAMPLE : 1.4 : SOLUTION :-")
printf("\nCase(1): Starting torque for direct switching, T_s = %.f N-m", T_s_1)
printf("\n         Starting current taken from supply line for direct switching, I_s = %.f A", I_s_1)
printf("\nCase(2): Starting torque for star-delta starting, T_s = %.f N-m", T_s_2)
printf("\n         Starting current taken from supply line for star-delta starting, I_s = %.1f A per phase", I_s_2)
printf("\nCase(3): Starting torque for auto-transformer starting, T_s = %.f N-m", T_s_3)
printf("\n         Starting current taken from supply line for auto-transformer starting, I_s = %.f A", I_s_line_3)
printf("\nCase(4): Starting torque for series-
```

```
parallel switch , T_s = %.f N-m", T_s_4)
printf("\n          Starting current taken from
   supply line for series-parallel switch , I_s = %.f
    A\n", I_s_4)
printf("\nNOTE: ERROR: Calculation mistakes and more
    approximation in textbook solution")
```

Scilab code Exa 39.5 Motor current per phase Current from the supply Starting torq

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
   MOTORS

// EXAMPLE : 1.5 :
// Page number 682
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 400.0        // IM voltage(V)
f = 50.0         // Frequency(Hz)
I_s = 5.0        // Full voltage starting current in
   terms of full load current
T_s = 2.0        // Full voltage starting torque in
   terms of full load torque
tap = 65.0       // Auto-tranformer tapping(%)

// Calculations
V_ph = V/3**0.5                 // Phase voltage(V)
V_ph_motor = tap/100*V_ph       // Motor phase voltage
```

```
when auto-transformer is used(V)
I_ph_motor = tap/100*I_s          // Motor phase current in terms of full load current
I_1 = tap/100*I_ph_motor          // Line current from supply in terms of full load current
T = (tap/100)**2*T_s              // Starting torque in terms of full load current
V_applied = V_ph/2**0.5           // Voltage to be applied to develop full-load torque(V)
I_line = V_applied/V_ph*I_s       // Line current in terms of full load current

// Results
disp("PART IV - EXAMPLE : 1.5 : SOLUTION :-")
printf("\nCase(i): Motor current per phase = %.2f*I_fl ", I_ph_motor)
printf("\nCase(ii): Current from the supply, I_1 = %.2f*I_fl ", I_1)
printf("\nCase(iii): Starting torque with auto-transformer starter, T = %.3f*T_fl ", T)
printf("\nVoltage to be applied if motor has to develop full-load torque at starting, V = %.f V", V_applied)
printf("\nLine current from the supply to develop full-load torque at starting = %.2f*I_fl ", I_line)
```

Scilab code Exa 39.6 Ratio of starting current to full load current

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
```

```
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.6 :
// Page number 682
clear ; clc ; close ; // Clear the work space and console

// Given data
hp = 10.0         // IM rating(hp)
V = 400.0         // IM voltage(V)
pf = 0.8          // Lagging power factor
n = 0.9           // Efficiency of IM
I_sc = 7.2        // Short-circuit current at 160V(A)
V_sc = 160.0      // Voltage at short-circuit(V)

// Calculations
I_fl = hp*746/(3**0.5*V*pf*n)     // Full-load line current(A)
I_sc_fv = V/V_sc*I_sc             // Short-circuit current at full voltage(A)
I_s = I_sc_fv/3.0                 // Starting current with star-delta starter(A)
I_s_fl = I_s/I_fl                 // Ratio of starting current to full load current

// Results
disp("PART IV - EXAMPLE : 1.6 : SOLUTION :-")
printf("\nRatio of starting current to full-load current, I_s/I_fl = %.1f \n", I_s_fl)
printf("\nNOTE: ERROR: Calculation mistake in final answer in textbook solution")
```

Scilab code Exa 39.7 Resistance to be placed in series with shunt field

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
   MOTORS

// EXAMPLE : 1.7 :
// Page number 685-686
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 230.0        // Voltage of DC shunt motor(V)
N_1 = 1000.0     // No load speed(rpm)
R_sh = 40.0      // Shunt resistance(ohm)
N_2 = 1200.0     // Speed with series resistance(rpm)

// Calculations
phi_2 = N_1/N_2          // Flux_2 in terms flux_1
I_N1 = V/R_sh            // Exciting current at 1000
   rpm(A)
phi_1 = 11.9             // Flux corresponding to I_N1
   (mWb)
phi_N2 = phi_1*phi_2     // Flux at 1200 rpm(mWb)
I_phi_N2 = 3.25          // Exciting current
   corresponding to phi_N2(A)
R = V/I_phi_N2           // Resistance in field
   circuit(ohm)
R_extra = R-R_sh         // Resistance to be placed in
    series with shunt field(ohm)

// Results
disp("PART IV - EXAMPLE : 1.7 : SOLUTION :-")
printf("\nResistance to be placed in series with
   shunt field = %.1f ohm", R_extra)
```

Scilab code Exa 39.9 `Speed and Current when field winding is shunted by a diverter`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
   MOTORS

// EXAMPLE : 1.9 :
// Page number 686-687
clear ; clc ; close ; // Clear the work space and
   console

// Given data
I_f1 = 25.0     // Current without diverter(A)
N_1 = 500.0     // Speed of dc series motor without
   diverter(rpm)

// Calculations
I_a2 = ((3.0/2)**0.5*I_f1**2*3/2)**0.5    // Field
   current with diverter(A)
N_2 = I_f1*N_1*3/(2*I_a2)                  // Speed
   with diverter(rpm)

// Results
disp("PART IV - EXAMPLE : 1.9 : SOLUTION :-")
printf("\nSpeed when field winding is shunted by a
   diverter, N_2 = %.f rpm", N_2)
printf("\nCurrent when field winding is shunted by a
    diverter, I_a2 = %.1f A", I_a2)
```

Scilab code Exa 39.10 `Additional resistance to be inserted in the field circuit to`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.10 :
// Page number 687
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 220.0        // DC shunt motor voltage(V)
I_a1 = 50.0      // Armature current at 800rpm(A)
N_1 = 800.0      // Speed of dc shunt motor(rpm)
N_2 = 1000.0     // Speed of dc shunt motor with additional resistance(rpm)
I_a2 = 75.0      // Armature current with additional resistance(A)
R_a = 0.15       // Armature resistance(ohm)
R_f = 250.0      // Field resistance(ohm)

// Calculations
E_b1 = V-R_a*I_a1                  // Back emf at 800 rpm(V)
I_f1 = V/R_f                       // Shunt field current(A)
E_b2 = V-R_a*I_a2                  // Back emf at 1000 rpm(V)
```

```
I_f2 = E_b2*N_1*I_f1/(E_b1*N_2)    // Shunt field current at 1000 rpm(A)
R_f2 = V/I_f2                      // Field resistance at 1000 rpm(ohm)
R_add = R_f2-R_f                   // Additional resistance required(ohm)

// Results
disp("PART IV - EXAMPLE : 1.10 : SOLUTION :-")
printf("\nAdditional resistance to be inserted in the field circuit to raise the speed = %.1f ohm\n", R_add)
printf("\nNOTE: ERROR: Calculation mistake in E_b2 in the textbook solution")
```

Scilab code Exa 39.11 Speed of motor with a diverter connected in parallel with se

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.11 :
// Page number 687
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 220.0        // DC series motor voltage(V)
I_1 = 20.0       // Armature current at 800rpm(A)
N_1 = 800.0      // Speed of dc series motor(rpm)
```

```
R_div = 0.4       // Diverter resistance(ohm)
R_a = 0.5         // Armature resistance(ohm)
R_f = 0.2         // Series field resistance(ohm)

// Calculations
E_b1 = V-(R_a+R_f)*I_1            // Back emf at 800 rpm(V)
I_2 = I_1*R_div/(R_div+R_f)       // Series field current at new speed(A)
E_b2 = V-(R_a*I_1+R_f*I_2)        // Back emf at new speed(V)
N_2 = I_1*N_1*E_b2/(I_2*E_b1)     // New speed with diverter(rpm)

// Results
disp("PART IV - EXAMPLE : 1.11 : SOLUTION :-")
printf("\nSpeed of motor with a diverter connected in parallel with series field , N_2 = %.f rpm", N_2)
```

Scilab code Exa 39.12 Diverter resistance as a percentage of field resistance

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.12 :
// Page number 687-688
clear ; clc ; close ; // Clear the work space and console
```

```

// Given data
speed_per = 15.0    // Motor speed increased by(%)

// Calculations
N_2 = (100+speed_per)/100       // New speed N_2(rpm)
phi_2 = 1/N_2*100               // Flux_2 in terms of full load flux
I_sc1 = 0.75                    // New series field current in terms of I_a1
I_a2 = N_2                      // Armature current in terms of I_a1
R_d = I_sc1/(I_a2-I_sc1)*100    // Diverter resistance in terms of series field resistance(%)

// Results
disp("PART IV - EXAMPLE : 1.12 : SOLUTION :-")
printf("\nDiverter resistance, R_d = %.1f percent of field resistance", R_d)
```

Scilab code Exa 39.13 Additional resistance to be placed in the armature circuit

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.13 :
// Page number 689
clear ; clc ; close ; // Clear the work space and
```

```
console

// Given data
V = 250.0        // Voltage of DC shunt motor(V)
N_1 = 400.0      // No load speed(rpm)
R_a = 0.5        // Armature resistance(ohm)
N_2 = 200.0      // Speed with additional resistance(rpm)
I_a = 20.0       // Armature current(A)

// Calculations
k_phi = (V-I_a*R_a)/N_1      // k
R = (V-k_phi*N_2)/I_a        // Resistance(ohm)
R_add = R-R_a                // Additional resistance to be placed in armature circuit(ohm)

// Results
disp("PART IV - EXAMPLE : 1.13 : SOLUTION :-")
printf("\nResistance to be placed in the armature circuit = %.f ohm\n", R_add)
printf("\nNOTE: ERROR: The given data doesnt match with example 1.7 as mentioned in problem statement")
```

Scilab code Exa 39.14 Resistance to be connected in series with armature to reduce

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

```

```
// EXAMPLE : 1.14 :
// Page number 689
clear ; clc ; close ; // Clear the work space and
    console

// Given data
V = 400.0          // Voltage of DC shunt motor(V)
hp = 20.0          // Power of DC shunt motor(hp)
I = 44.0           // Current drawn by motor(A)
N_1 = 1000.0       // Speed(rpm)
N_2 = 800.0        // Speed with additional resistance(
    rpm)
R_sh = 200.0       // Shunt field resistance(ohm)

// Calculations
output = hp*746              // Motor output(W)
I_f1 = V/R_sh                // Shunt field current(A)
I_a1 = I-I_f1                // Armature current(A)
E_b1 = output/I_a1           // Back emf(V)
R_a = (V-E_b1)/I_a1          // Armature resistance(
    ohm)
I_a2 = I_a1*(N_2/N_1)**2     // Armature current at N2
    (A)
E_b2 = N_2/N_1*E_b1          // Back emf at N2(V)
r = ((V-E_b2)/I_a2)-R_a      // Resistance connected
    in series with armature(ohm)

// Results
disp("PART IV - EXAMPLE : 1.14 : SOLUTION :-")
printf("\nResistance to be connected in series with
    armature to reduce speed, r = %.2f ohm", r)
```

Scilab code Exa 39.15 Ohmic value of resistor connected in the armature circuit

```
// A Texbook on POWER SYSTEM ENGINEERING
```

```
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
   MOTORS

// EXAMPLE : 1.15 :
// Page number 690
clear ; clc ; close ; // Clear the work space and
   console

// Given data
hp = 15.0          // Power of DC shunt motor(hp)
V = 400.0          // Voltage of DC shunt motor(V)
N_reduce = 20.0    // Speed is to be reduced by(%)
I_f = 3.0          // Field current(A)
R_a = 0.5          // Armature resistance(ohm)
n = 0.85           // Efficiency of motor

// Calculations
motor_input = hp*746/n           // Motor input(W)
I = motor_input/V                // Motor current(A)
I_a1 = I-I_f                     // Armature current(
   A)
I_a2 = I_a1                      // Armature current
   at new speed(A)
E_b1 = V-I_a1*R_a                // Back emf(V)
E_b2 = E_b1*(100-N_reduce)/100   // Back emf at new
   speed(V)
r = ((V-E_b2)/I_a2)-R_a          // Ohmic value of
   resistor connected in the armature circuit(ohm)

// Results
disp("PART IV - EXAMPLE : 1.15 : SOLUTION :-")
printf("\nOhmic value of resistor connected in the
   armature circuit, r = %.2f ohm", r)
```

Scilab code Exa 39.16 `External resistance per phase added in rotor circuit to redu`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
   MOTORS

// EXAMPLE : 1.16 :
// Page number 697-698
clear ; clc ; close ; // Clear the work space and
   console

// Given data
p = 6.0              // Number of poles
f = 50.0             // Frequency(Hz)
R_2 = 0.3            // Rotor resistance per phase(ohm)
N_1 = 960.0          // Rotor speed(rpm)
N_2 = 800.0          // New rotor speed with external
   resistance(rpm)

// Calculations
N_s = 120*f/p              // Synchronous speed(rpm)
S_1 = (N_s-N_1)/N_s        // Slip at full load
S_2 = (N_s-N_2)/N_s        // New slip
R = (S_2/S_1*R_2)-R_2      // External resistance per
   phase added in rotor circuit to reduce speed(ohm)

// Results
disp("PART IV - EXAMPLE : 1.16 : SOLUTION :-")
printf("\nExternal resistance per phase added in
```

```
rotor circuit to reduce speed , R = %.1 f ohm" , R)
```

Scilab code Exa 39.17 Braking torque and Torque when motor speed has fallen

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
   MOTORS

// EXAMPLE : 1.17 :
// Page number 699
clear ; clc ; close ; // Clear the work space and
   console

// Given data
hp = 50.0                // DC shunt motor rating(hp)
V = 440.0                // Voltage(V)
I_b = 150.0              // Breaking current(A)
N_reduce = 40.0          // Speed of motor fallen by(%)
R_a = 0.1                // Armature resistance(ohm)
I_a_fl = 100.0           // Full-load armature current(A)
N_fl = 600.0             // Full-load speed(rpm)

// Calculations
E_b = V-I_a_fl*R_a                          // Back emf of
   motor(V)
V_a = V+E_b                                 // Voltage
   across armature when braking starts(V)
R_b = V_a/I_b                               // Resistance
   required(ohm)
R_extra = R_b-R_a                           // Extra
```

```
resistance required(ohm)
T_fl = hp*746*60/(2*%pi*N_fl)         // Full-load torque(N-m)
T_initial_b = T_fl*I_b/I_a_fl         // Initial breaking torque(N-m)
E_b2 = E_b*(100-N_reduce)/100         // Back emf at new speed(V)
I = (V+E_b2)/R_b                      // Current(A)
EBT = T_fl*I/I_a_fl                   // Torque when motor speed reduced by 40%(N-m)

// Results
disp("PART IV - EXAMPLE : 1.17 : SOLUTION :-")
printf("\nBraking torque = %.1f N-m", T_initial_b)
printf("\nTorque when motor speed has fallen, E.B.T = %.1f N-m\n", EBT)
printf("\nNOTE: ERROR: Calculation mistakes in the textbook solution")
```

Scilab code Exa 39.18 Initial plugging torque and Torque at standstill

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.18 :
// Page number 699-700
clear ; clc ; close ; // Clear the work space and console

```

```
// Given data
V = 400.0             // Voltage of IM(V)
p = 4.0               // Number of poles
f = 50.0              // Frequency(Hz)
hp = 25.0             // Power developed(hp)
S = 0.04              // Slip
R_X_2 = 1.0/4         // Ratio of rotor resistance to
    standstill reactance i.e R2/X2

// Calculations
N_s = 120*f/p

    // Synchronous speed(rpm)
N_fl = N_s*(1-S)

    // Full load speed(rpm)
T_fl = hp*735.5*60/(2*%pi*N_fl*9.81)
                                                   // Full-
    load torque(kg-m)
S_1 = 1.0

    // Slip at standstill
X_R_2 = 1.0/R_X_2

    // Ratio of standstill reactance to rotor
    resistance
T_s_fl = S_1/S*((1+(S*X_R_2)**2)/(1+(S_1*X_R_2)**2))
                                // T_standstill/T_fl
T_standstill = T_s_fl*T_fl

    // Standstill torque(kg-m)
S_instant = (N_s+N_fl)/N_s

    // Slip at instant of plugging
T_initial = (S_instant/S)*((1+(S*X_R_2)**2)/(1+(
    S_instant*X_R_2)**2))*T_fl    // Initial plugging
    torque(kg-m)

```

```
// Results
disp("PART IV - EXAMPLE : 1.18 : SOLUTION :-")
printf("\nInitial plugging torque = %.1f kg-m", T_initial)
printf("\nTorque at standstill = %.f kg-m\n", T_standstill)
printf("\nNOTE: ERROR: Calculation mistake from full-load torque onwards. Hence, change in obtained answer from that of textbook")
```

Scilab code Exa 39.19 Value of resistance to be connected in motor circuit

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.19 :
// Page number 701
clear ; clc ; close ; // Clear the work space and console

// Given data
T = 312.5          // Load torque(N-m)
N = 500.0          // Speed limit(rpm)
R_total = 1.0      // Total resistance of armature & field(ohm)

// Calculations
input_load = 2*%pi*N*T/60            // Input from load(W)
```

```
E = 345.0                                // Voltage from
    magnetization curve(V). From Fig E1.5 page no 701
I = 47.5                                 // Current from
    magnetization curve(A). From Fig E1.5 page no 701
R = E/I                                  // Resistance(ohm
    )
R_add = R-R_total                        // Additional
    resistance required(ohm)

// Results
disp("PART IV - EXAMPLE : 1.19 : SOLUTION :-")
printf("\nValue of resistance to be connected in
    motor circuit = %.2f ohm", R_add)
```

Scilab code Exa 39.20 Current drawn by the motor from supply and Resistance requir

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
   MOTORS

// EXAMPLE : 1.20 :
// Page number 702
clear ; clc ; close ; // Clear the work space and
    console
funcprot(0)

// Given data
V = 500.0           // Shunt motor voltage(V)
load = 400.0        // Hoist load(kg)
speed = 2.5         // Hoist raised speed(m/sec)
```

```
n_motor = 0.85     // Efficiency of motor
n_hoist = 0.75     // Efficiency of hoist

// Calculations
P_output = load*speed*9.81                          // Power output from motor(W)
P_input = P_output/(n_motor*n_hoist)                // Motor input(W)
I = P_input/V                                       // Current drawn from supply(A)
output_G = load*speed*9.81*n_motor*n_hoist          // Generator output(W)
R = V**2/output_G                                   // Resistance required in the armature circuit for rheostatic braking(ohm)

// Results
disp("PART IV - EXAMPLE : 1.20 : SOLUTION :-")
printf("\nCurrent drawn by the motor from supply = %.1f A", I)
printf("\nResistance required in the armature circuit for rheostatic braking, R = %.f ohm", R)
```

Scilab code Exa 39.21 One hour rating of motor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.21 :
```

```
// Page number 705
clear ; clc ; close ; // Clear the work space and console

// Given data
t = 1.0            // Time(hour)
hp = 15.0          // Motor rating(hp)
T = 2.0            // Time constant(hour)
theta_f = 40.0     // Temperature rise( C )

// Calculations
P = (1.0/(1-exp(-t/T)))**0.5*hp      // One−hour rating of motor(hp)

// Results
disp("PART IV − EXAMPLE : 1.21 : SOLUTION :−")
printf("\nOne−hour rating of motor, P = %.f hp\n", P)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more approximation in the textbook solution")
```

Scilab code Exa 39.22 Final temperature rise and Thermal time constant of the moto

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.22 :
// Page number 706
```

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

// Given data
hp = 10.0                    // Motor rating(hp)
d = 0.7                      // Diameter of cylinder(m)
l = 1.0                      // Length of cylinder(m)
w = 380.0                    // Weight of motor(kgm)
heat_specific = 700.0        // Specific heat(J/kg/1 C )
heat_dissipation = 15.0      // Outer surface heat dissipation rate(W/sq.cm/ C )
n = 0.88                     // Efficiency

// Calculations
output = hp*735.5                                          // Output of motor(W)
loss = (1-n)/n*output                                   // Losses(W)
area_cooling = %pi*d*l                                 // Cooling surface area(sq.m)
theta_m = loss/(area_cooling*heat_dissipation)            // Final temperature rise( C )
T_sec = w*heat_specific/(area_cooling*heat_dissipation)  // Thermal time constant(sec)
T_hour = T_sec/3600                                        // Thermal time constant(hours)

// Results
disp("PART IV - EXAMPLE : 1.22 : SOLUTION :-")
printf("\nFinal temperature rise, _m = %.1f C", theta_m)
printf("\nThermal time constant of the motor = %.2f hours\n", T_hour)
```

```
printf("\nNOTE: ERROR: Mistake in calculating
    thermal time constant in the textbook solution")
```

Scilab code Exa 39.23 `Half hour rating of motor`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC
    MOTORS

// EXAMPLE : 1.23 :
// Page number 706
clear ; clc ; close ; // Clear the work space and
    console

// Given data
hp = 25.0           // Motor rating(hp)
T = 100.0/60        // Heating time constant(hour)
theta = 40.0        // Temperature rise( C )
t = 0.5             // Time(hour)
n = 0.85            // Motor maximum efficiency

// Calculations
output = hp*735.5/1000                              //
    Output of motor(kW)
output_max = output*n                               //
    Power at maximum efficiency(kW)
theta_f2 = theta/(1-exp(-t/T))                      //
      _f2 ( C )
loss = 1+(output/output_max)**2                     //
    Losses at 18.4 kW output in terms of W
```

```
P = ((theta_f2/theta*loss)-1)**0.5*output_max    // Half-hour rating of motor(kW)
P_hp = P*1000/735.5                              // Half-hour rating of motor(hp)

// Results
disp("PART IV - EXAMPLE : 1.23 : SOLUTION :-")
printf("\nHalf-hour rating of motor, P = %.f kW = %.1f hp (metric)\n", P,P_hp)
printf("\nNOTE: ERROR: Calculation mistake from final temperature rise onwards in textbook")
```

Scilab code Exa 39.24 Time for which the motor can run at twice the continuously r

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.24 :
// Page number 706
clear ; clc ; close ; // Clear the work space and console

// Given data
theta_f1 = 40.0      // Temperature rise( C )
T = 100.0            // Heating time constant(min)
rated_2 = 2.0        // Motor at twice the continuously rating

// Calculations
```

```
loss_cu = 2.0**2                        // Copper loss at twice full load in terms of W
loss_total = loss_cu+1                  // Total losses at full load in terms of W
theta_f2 = theta_f1*loss_total/rated_2  //  _f2 ( C )
t = log(1-(theta_f1/theta_f2))*(-T)     // Time for which motor can run at twice the continuously rated output without overheating(min)

// Results
disp("PART IV - EXAMPLE : 1.24 : SOLUTION :-")
printf("\nMotor can run at twice the continuously rated output without overheating for time, t = %.f min", t)
```

Scilab code Exa 39.25 Maximum overload that can be carried by the motor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.25 :
// Page number 706-707
clear ; clc ; close ; // Clear the work space and console

// Given data
kW = 20.0          // Motor output(kW)
theta_1 = 50.0     // Temperature rise not to be
```

```
    exceeded on overload( C )
t_1 = 1.0           // Time on overload(hour)
theta_2 = 30.0      // Temperature rise on full-load( C )
t_2 = 1.0           // Time on full-load(hour)
theta_3 = 40.0      // Temperature rise on full-load( C )
t_3 = 2.0           // Time on full-load(hour)

// Calculations
e_lambda = 1.0/3                    // Obtained directly from textbook
theta_f = theta_2/(1-e_lambda)      //  _f ( C )
theta_f1 = theta_1/(1-e_lambda)     //  '_f( C )
P = (theta_f1/theta_f)**0.5*kW      // Maximum overload that can be carried by the motor(kW)

// Results
disp("PART IV - EXAMPLE : 1.25 : SOLUTION :-")
printf("\nMaximum overload that can be carried by the motor, P = %.1f kW", P)
```

Scilab code Exa 39.26 `Required size of continuously rated motor`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.26 :
// Page number 707-708
```

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

// Given data
hp_1 = 100.0         // Motor load(hp)
t_1 = 10.0           // Time of operation(min)
hp_2 = 0             // Motor load(hp)
t_2 = 5.0            // Time of operation(min)
hp_3 = 60.0          // Motor load(hp)
t_3 = 8.0            // Time of operation(min)
hp_4 = 0             // Motor load(hp)
t_4 = 4.0            // Time of operation(min)

// Calculations
t_total = t_1+t_2+t_3+t_4                                                    // Total time of operation(min)
rms = ((hp_1**2*t_1+hp_2**2*t_2+hp_3**2*t_3+hp_4**2*t_4)/t_total)**0.5  // rms horsepower

// Results
disp("PART IV - EXAMPLE : 1.26 : SOLUTION :-")
printf("\nRequired size of continuously rated motor = %.f H.P\n", rms)
printf("\nNOTE: ERROR: Calculation mistake in the textbook")
printf("\n      Actual value is written here instead of standard values")
```

Scilab code Exa 39.27 Suitable size of the motor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION
```

```

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.27 :
// Page number 708
clear ; clc ; close ; // Clear the work space and console

// Given data
hp_1 = 200.0      // Motor load(hp)
t_1 = 5.0         // Time of operation(min)
hp_2 = 100.0      // Motor load(hp)
t_2 = 10.0        // Time of operation(min)
hp_3 = 0          // Motor load(hp)
t_3 = 3.0         // Time of operation(min)

// Calculations
m = hp_1/t_1                                                     // Slope of uniform rise power
t_total = t_1+t_2+t_3                                            // Total time of operation(min)
ans = integrate('(m*x)**2','x', 0, t_1)                          // Integarted uniform area upto 5 min
rms = ((ans+hp_2**2*t_2+hp_3**2*t_3)/t_total)**0.5               // rms horsepower

// Results
disp("PART IV - EXAMPLE : 1.27 : SOLUTION :-")
printf("\nrms horsepower = %.1f HP. Therefore, a motor of %.f H.P should be selected", rms,rms+4)
```

Scilab code Exa 39.28 Time taken to accelerate the motor to rated speed against fu

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.28 :
// Page number 710
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 440.0        // DC shunt motor voltage(V)
hp = 50.0        // Motor rating(hp)
N = 600.0        // Speed(rpm)
I = 80.0         // Current at full-load(A)
I_1 = 1.1        // Lower current limit in terms of full current
I_2 = 1.5        // Upper current limit in terms of full current
J = 20.0         // Moment of inertia(kg-m^2)

// Calculations
T = hp*746*60/(2*%pi*N)        // Full load torque of motor(N-m)
T_avg_start = (I_1+I_2)/2*T    // Average starting torque(N-m)
T_g = ((I_1+I_2)/2-1)*T        // Torque available for acceleration(N-m)
```

```
alpha = T_g/J                    // Angular acceleration(rad/sec^2)
t = 2*%pi*N/(60*alpha)           // Time taken to accelerate the motor(sec)

// Results
disp("PART IV - EXAMPLE : 1.28 : SOLUTION :-")
printf("\nTime taken to accelerate the motor to rated speed against full load torque, t = %.2f sec\n", t)
printf("\nNOTE: ERROR: Calculation mistake in the textbook solution")
```

Scilab code Exa 39.29 Time taken to accelerate the motor to rated speed

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.29 :
// Page number 710
clear ; clc ; close ; // Clear the work space and console

// Given data
hp = 50.0          // Motor rating(hp)
N = 600.0          // Speed(rpm)
energy = 276.0     // Stored energy(kg-m/hp)

// Calculations
```

```
g = 9.81
T = hp*746*60/(2*%pi*N*g)              // Full load torque of motor(kg-m)
J = hp*energy*2*g/(2*%pi*N/60)**2      // Moment of inertia(kg-m^2)
alpha = T*g/J                          // Angular acceleration(rad/sec^2)
t = 2*%pi*N/(60*alpha)                 // Time taken to accelerate the motor to rated speed(sec)

// Results
disp("PART IV - EXAMPLE : 1.29 : SOLUTION :-")
printf("\nTime taken to accelerate the motor to rated speed, t = %.2f sec", t)
```

Scilab code Exa 39.30 Time taken to accelerate a fly wheel

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.30 :
// Page number 710
clear ; clc ; close ; // Clear the work space and console

// Given data
J = 1270.0       // Moment of inertia of fly-wheel(kg-m^2)
N = 500.0        // Speed(rpm)
```

```
hp = 50.0        // Motor rating(hp)

// Calculations
g = 9.81
T = hp*746*60/(2*%pi*N*g)          // Full load torque of motor(kg-m)
T_m = 2*T                          // Accelerating torque(kg-m)
alpha = T_m*g/J                    // Angular acceleration(rad/sec^2)
t = 2*%pi*N/(60*alpha)             // Time taken to accelerate a fly-wheel(sec)

// Results
disp("PART IV - EXAMPLE : 1.30 : SOLUTION :-")
printf("\nTime taken to accelerate a fly-wheel, t = %.1f sec", t)
```

Scilab code Exa 39.31 `Time taken for dc shunt motor to fall in speed with constant`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.31 :
// Page number 710-711
clear ; clc ; close ; // Clear the work space and console

// Given data
```

```
N_1 = 1000.0    // Speed of dc shunt motor(rpm)
N_2 = 400.0     // Speed of dc shunt motor(rpm)
R = 14.0        // Resistance connected across armature(ohm)
E_1 = 210.0     // EMF induced in armature at 1000 rpm(V)
J = 17.0        // Moment of inertia(kg-m^2)
T_F = 1.0       // Frictional torque(kg-m)

// Calculations
g = 9.81
output = E_1**2/R                          // Motor output(W)
T_E = output*60/(2*%pi*N_1*g)              // Electric braking torque(kg-m)
w_1 = 2*%pi*N_1/60                         //  _1 (rad/sec)
k = T_E/w_1
t = J/(g*k)*log(N_1/N_2)                   // Time taken for dc shunt motor to fall in speed with constant excitation(sec)
kw = T_E*N_2/N_1                           // k
t_F = J/(g*k)*log((1+T_E)/(1+kw))          // Time for the same fall if frictional torque exists(sec)

// Results
disp("PART IV - EXAMPLE : 1.31 : SOLUTION :-")
printf("\nTime taken for dc shunt motor to fall in speed with constant excitation , t = %.1f sec", t)
printf("\nTime for the same fall if frictional torque exists , t = %.1f sec", t_F)
```

Scilab code Exa 39.32 Time taken and Number of revolutions made to come to standst

```
// A Texbook on POWER SYSTEM ENGINEERING
```

```
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.32 :
// Page number 711
clear ; clc ; close ; // Clear the work space and console

// Given data
V = 400.0        // Voltage of synchronous motor(V)
p = 8.0          // Number of poles
J = 630.0        // Moment of inertia(kg-m^2)
T_E = 165.0      // Braking torque(kg-m)
kw_1 = 690.0     // Electric braking torque(kg-m)
T_F = 1.4        // Frictional torque(kg-m)
f = 50.0         // Frequency(Hz). Assumed normal supply frequency

// Calculations
g = 9.81
// Case(a) Plugging
T_B = T_E+T_F                                  // Torque(kg-m)
beta = T_B*g/J                                 // Retardation(rad/sec^2)
N_s = 120*f/p                                  // Synchronous speed(rad/sec)
w = 2*%pi*N_s/60                               //  (rad/sec)
```

```
t_a = integrate('-1.0/beta','w', w, 0)                                        // Time taken to stop the motor(sec)
n_a = integrate('-w/(2*%pi*beta)','w', w, 0)                              // Number of revolutions
// Case(b) Rheostatic braking
k = kw_1/w
t_b = J/(g*k)*log((T_F+kw_1)/T_F)                                              // Time taken to stop the motor(sec)
n_b = 1.0/(2*%pi*k)*(J/(g*k)*(T_F+kw_1)*(1-exp(-k*g*t_b/J))-T_F*t_b)  // Number of revolutions

// Results
disp("PART IV - EXAMPLE : 1.32 : SOLUTION :-")
printf("\nCase(a): Time taken to come to standstill by plugging, t = %.1f sec", t_a)
printf("\n         Number of revolutions made to come to standstill by plugging, n = %.f revolutions", n_a)
printf("\nCase(b): Time taken to come to standstill by rheostatic braking, t = %.1f sec", t_b)
printf("\n         Number of revolutions made to come to standstill by rheostatic braking, n = %.f revolutions\n", n_b)
printf("\nNOTE: ERROR: Calculation mistake in finding number of revolution in case(a) in textbook solution")
```

Scilab code Exa 39.33 `Inertia of flywheel required`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION
```

```

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.33 :
// Page number 712-713
clear ; clc ; close ; // Clear the work space and console

// Given data
hp = 500.0        // Rating of IM(hp)
N_nl = 40.0       // No-load speed(rpm)
S_fl = 0.12       // Slip at full-load
T_l = 41500.0     // Load torque(kg-m)
t = 10.0          // Duration of each rolling period(sec)

// Calculations
g = 9.81
T_fl = hp*746*60/(2*%pi*N_nl*g*(1-S_fl))      // Torque at full-load(kg-m)
T_m = 2.0*T_fl                                // Motor torque at any instant(kg-m)
slip = S_fl*N_nl                              // Slip (rpm)
slip_rad = slip*2*%pi/60                      // Slip (rad/sec)
k = slip_rad/T_fl
J = -g*t/(k*log(1-(T_m/T_l)))                 // Inertia of flywheel(kg-m^2)

// Results
disp("PART IV - EXAMPLE : 1.33 : SOLUTION :-")
printf("\nInertia of flywheel required, J = %.3e kg-m^2\n", J)
printf("\nNOTE: ERROR : J = 2.93*10^6 kg-m^2 and not 2.93*10^5 as mentioned in the textbook solution"
```

```
)
```

Scilab code Exa 39.34 Moment of inertia of the flywheel

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 1: INDUSTRIAL APPLICATIONS OF ELECTRIC MOTORS

// EXAMPLE : 1.34 :
// Page number 713
clear ; clc ; close ; // Clear the work space and console

// Given data
T_l = 150.0       // Load torque(kg-m)
t = 15.0          // Duration of load torque(sec)
T_m = 85.0        // Motor torque(kg-m)
N = 500.0         // Speed(rpm)
s_fl = 0.1        // Full-load slip

// Calculations
g = 9.81
slip = N*s_fl*2*%pi/60                    // Slip(rad/sec)
k = slip/T_m
T_0 = 0                                   // No-load torque(kg-m)
J = -g*t/(k*log((T_l-T_m)/(T_l-T_0)))    // Moment of inertia of flywheel(kg-m^2)

```

```
// Results
disp("PART IV - EXAMPLE : 1.34 : SOLUTION :-")
printf("\nInertia of flywheel required , J = %.f kg-m
    ^2\n", J)
printf("\nNOTE: ERROR : Calculation mistake in the
    textbook solution")
```

Chapter 40

HEATING AND WELDING

Scilab code Exa 40.1 Diameter Length and Temperature of the wire

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 2: HEATING AND WELDING

// EXAMPLE : 2.1 :
// Page number 724-725
clear ; clc ; close ; // Clear the work space and
    console

// Given data
P = 15.0*10**3      // Power supplied(W)
V = 220.0           // Voltage(V)
T_w = 1000.0        // Temperature of wire( C )
T_c = 600.0         // Temperature of charges( C )
k = 0.6             // Radiatting efficiency
e = 0.9             // Emissivity

```

```
// Calculations
rho = 1.016/10**6                                   // Specific resistance(ohm-m)
d_square = 4*rho*P/(%pi*V**2)                       // d^2 in terms of l
T_1 = T_w+273                                       // Absolute temperature( C )
T_2 = T_c+273                                       // Absolute temperature( C )
H = 5.72*10**4*k*e*((T_1/1000)**4-(T_2/1000)**4)    // Heat produced(watts/sq.m)
dl = P/(%pi*H)
l = (dl**2/d_square)**(1.0/3)                       // Length of wire(m)
d = dl/l                                            // Diameter of wire(m)
T_2_cold = 20.0+273                                 // Absolute temperature at the 20 C normal temperature( C )
T_1_cold = (H/(5.72*10**4*k*e)+(T_2_cold/1000)**4)**(1.0/4)*1000    // Absolute temperature when charge is cold( C )
T_1_c = T_1_cold-273                                // Temperature when charge is cold( C )

// Results
disp("PART IV - EXAMPLE : 2.1 : SOLUTION :-")
printf("\nDiameter of the wire, d = %.3f cm", d*100)
printf("\nLength of the wire, l = %.2f m", l)
printf("\nTemperature of the wire when charge is
```

```
    cold , T_1 = %.f C absolute = %.f C \n",
    T_1_cold ,T_1_c)
printf("\nNOTE: Slight changes in the obtained
    answer from that of textbook is due to more
    precision here")
```

Scilab code Exa 40.2 Width and Length of nickel chrome strip

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 2: HEATING AND WELDING

// EXAMPLE : 2.2 :
// Page number 725
clear ; clc ; close ; // Clear the work space and
    console

// Given data
P = 15.0*10**3           // Power supplied(W)
V = 220.0                // Voltage(V)
T_w = 1000.0             // Temperature of wire( C )
T_c = 600.0              // Temperature of charges( C
    )
k = 0.6                  // Radiatting efficiency
e = 0.9                  // Emissivity
thick = 0.25/1000        // Thickness of nickel-chrome
     strip(m)

// Calculations
rho = 1.016/10**6
                                        // Specific
```

```
resistance(ohm-m)
R = V**2/P                                                    //
Resistance(ohm)
l_w = R*thick/rho                                     // Length of
strip in terms of w
T_1 = T_w+273                                               //
Absolute temperature( C )
T_2 = T_c+273                                               //
Absolute temperature( C )
H = 5.72*10**4*k*e*((T_1/1000)**4-(T_2/1000)**4)
        // Heat produced(watts/sq.m)
wl = P/(2*H)
w = (wl/l_w)**0.5                                     // Width of
nickel-chrome strip(m)
l = w*l_w                                                     //
Length of nickel-chrome strip(m)

// Results
disp("PART IV - EXAMPLE : 2.2 : SOLUTION :-")
printf("\nWidth of nickel-chrome strip, w = %.3f cm"
    , w*100)
printf("\nLength of nickel-chrome strip, l = %.1f m"
    , l)
```

Scilab code Exa 40.3 Power drawn under various connections

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
```

```
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 2: HEATING AND WELDING

// EXAMPLE : 2.3 :
// Page number 726-727
clear ; clc ; close ; // Clear the work space and
    console

// Given data
R = 50.0     // Resistance of each resistor in oven(
    ohm)
n = 6.0      // Number of resistance
V = 400.0    // Supply voltage(V)
tap = 50.0   // Auto-transformer tapping(%)

// Calculations
// Case(a)(i)
P_a_i = n*V**2/R*10**-3                          //
    Power consumption for 6 elements in parallel(kW)
// Case(a)(ii)
P_each_a_ii = V**2/(R+R)*10**-3                  //
    Power consumption in each group of 2 resistances
    in series(kW)
P_a_ii = n/2*P_each_a_ii                         //
    Power consumption for 3 groups(kW)
// Case(b)(i)
V_b_i = V/3**0.5                                 //
    Supply voltage against each resistance(V)
P_each_b_i = 2*V_b_i**2/R*10**-3                 //
    Power consumption in each branch(kW)
P_b_i = n/2*P_each_b_i                           //
    Power consumption for 2 elements in parallel in
    each phase(kW)
// Case(b)(ii)
V_b_ii = V/3**0.5                                //
    Supply voltage to any branch(V)
```

```
P_each_b_ii = V_b_ii**2/(R+R)*10**-3          //
    Power consumption in each branch(kW)
P_b_ii = n/2*P_each_b_ii                      //
    Power consumption for 2 elements in series in
    each phase(kW)
// Case(c)(i)
P_each_c_i = V**2/(R+R)*10**-3                //
    Power consumption by each branch(kW)
P_c_i = n/2*P_each_c_i                        //
    Power consumption for 2 elements in series in
    each branch(kW)
// Case(c)(ii)
P_each_c_ii = 2*V**2/R*10**-3                 //
    Power consumption by each branch(kW)
P_c_ii = n/2*P_each_c_ii                      //
    Power consumption for 2 elements in parallel in
    each branch(kW)
// Case(d)
V_d = V*tap/100                               //
    Voltage under tapping(V)
ratio_V = V_d/V                               //
    Ratio of normal voltage to tapped voltage
loss = ratio_V**2                             //
    Power loss in terms of normal power

// Results
disp("PART IV - EXAMPLE : 2.3 : SOLUTION :-")
printf("\nCase(a): AC Single phase 400 V supply")
printf("\n          Case(i) :  Power consumption for
    6 elements in parallel = %.1f kW", P_a_i)
printf("\n          Case(ii):  Power consumption for
    3 groups in parallel with 2 element in series = %
    .1f kW", P_a_ii)
printf("\nCase(b): AC Three phase 400 V supply with
    star combination")
printf("\n          Case(i) :  Power consumption for
    2 elements in parallel in each phase = %.1f kW",
    P_b_i)
```

```
printf("\n          Case(ii):  Power consumption for
   2 elements in series in each phase = %.1f kW",
   P_b_ii)
printf("\nCase(c): AC Three phase 400 V supply with
   delta combination")
printf("\n          Case(i) :  Power consumption for
   2 elements in series in each branch = %.1f kW",
   P_c_i)
printf("\n          Case(ii):  Power consumption for
   2 elements in parallel in each branch = %.1f kW",
    P_c_ii)
printf("\nCase(d): Power loss will be %.2f of the
   values obtained as above with auto-transformer
   tapping", loss)
```

Scilab code Exa 40.4 Amount of energy required to melt brass

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 2: HEATING AND WELDING

// EXAMPLE : 2.4 :
// Page number 728
clear ; clc ; close ; // Clear the work space and
   console

// Given data
w_brass = 1000.0          // Weight of brass(kg)
time = 1.0                // Time(hour)
heat_sp = 0.094           // Specific heat
fusion = 40.0             // Latent heat of fusion(
```

```
kcal/kg)
T_initial = 24.0          // Initial temperature( C )
melt_point = 920.0        // Melting point of brass( C )
n = 0.65                  // Efficiency

// Calculations
heat_req = w_brass*heat_sp*(melt_point-T_initial)       // Heat required to raise the temperature(kcal)
heat_mel = w_brass*fusion                               // Heat required for melting(kcal)
heat_total = heat_req+heat_mel                          // Total heat required(kcal)
energy = heat_total*1000*4.18/(10**3*3600*n)            // Energy input(kWh)
power = energy/time                                     // Power(kW)

// Results
disp("PART IV - EXAMPLE : 2.4 : SOLUTION :-")
printf("\nAmount of energy required to melt brass = %.f kWh", energy)
```

Scilab code Exa 40.5 Height up to which the crucible should be filled to obtain ma

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 2: HEATING AND WELDING
```

```
// EXAMPLE : 2.5 :
// Page number 728-729
clear ; clc ; close ; // Clear the work space and console

// Given data
V_2 = 12.0          // Secondary voltage(V)
P = 30.0*10**3      // Power(W)
PF = 0.5            // Power factor

// Calculations
I_2 = P/(V_2*PF)                 // Secondary current(A)
Z_2 = V_2/I_2                    // Secondary impedance(ohm)
R_2 = Z_2*PF                     // Secondary resistance(ohm)
sin_phi = (1-PF**2)**0.5
X_2 = Z_2*sin_phi                // Secondary reactance(ohm)
h = R_2/X_2
H_m = h                          // Height up to which the crucible should be filled to obtain maximum heating effect in terms of H_c

// Results
disp("PART IV - EXAMPLE : 2.5 : SOLUTION :-")
printf("\nHeight up to which the crucible should be filled to obtain maximum heating effect, H_m = %.3f*H_c \n", H_m)
printf("\nNOTE: ERROR: Calculation mistake in textbook solution and P is 30 kW not 300 kW")
```

Scilab code Exa 40.6 Voltage necessary for heating and Current flowing in the mate

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 2: HEATING AND WELDING

// EXAMPLE : 2.6 :
// Page number 732
clear ; clc ; close ; // Clear the work space and console

// Given data
l = 10.0            // Length of material(cm)
b = 10.0            // Breadth of material(cm)
t = 3.0             // Thickness of material(cm)
f = 20.0*10**6      // Frequency(Hz)
P = 400.0           // Power absorbed(W)
e_r = 5.0           // Relative permittivity
PF = 0.05           // Power factor

// Calculations
e_0 = 8.854*10**-12           // Absolute permittivity
A = l*b*10**-4                // Area(Sq.m)
C = e_0*e_r*A/(t/100)         // Capacitace of parallel plate condenser(F)
X_c = 1.0/(2*%pi*f*C)         // Reactance of condenser(ohm)
phi = acosd(PF)               //   (  )
R = X_c*tand(phi)             // Resistance of condenser(ohm)
V = (P*R)**0.5                // Voltage necessary for heating(V)
I_c = V/X_c                   // Current flowing in the material(A)

```

```
// Results
disp("PART IV - EXAMPLE : 2.6 : SOLUTION :-")
printf("\nVoltage necessary for heating , V = %.f V", V)
printf("\nCurrent flowing in the material , I_c = %.2f A\n", I_c)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here & approximation in textbook")
```

Scilab code Exa 40.7 `Voltage applied across electrodes and Current through the mat`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 2: HEATING AND WELDING

// EXAMPLE : 2.7 :
// Page number 732-733
clear ; clc ; close ; // Clear the work space and console

// Given data
l = 4.0             // Length of material(cm)
b = 2.0             // Breadth of material(cm)
t = 1.0             // Thickness of material(cm)
l_e = 20.0          // Length of area(cm)
b_e = 2.0           // Breadth of area(cm)
dis = 1.6           // Distance of separation of electrode(cm)
f = 20.0*10**6      // Frequency(Hz)
P = 80.0            // Power absorbed(W)
```

```
e_r1 = 5.0        // Relative permittivity
e_r2 = 1.0        // Relative permittivity of air
PF = 0.05         // Power factor

// Calculations
e_0 = 8.854*10**-12                                      // Absolute permittivity
A_1 = (l_e-l)*b_e*10**-4                                 // Area of one electrode(sq.m)
A_2 = l*b*10**-4                                         // Area of material under electrode(sq.m)
d = dis*10**-2                                           // Distance of separation of electrode(m)
d_1 = t*10**-2                                           // (m)
d_2 = (d-d_1)                                            // (m)
C = e_0*((A_1*e_r2/d)+(A_2/((d_1/e_r1)+(d_2/e_r2))))     // Capacitance(F)
X_c = 1.0/(2*%pi*f*C)                                    // Reactance(ohm)
phi = acosd(PF)                                          //    (    )
R = X_c*tand(phi)                                        // Resistance(ohm)
V = (P*R)**0.5                                           // Voltage applied across electrodes(V)
I_c = V/X_c                                              // Current through the material(A)

```

```
// Results
disp("PART IV - EXAMPLE : 2.7 : SOLUTION :-")
printf("\nVoltage applied across electrodes , V = %.f
    V", V)
printf("\nCurrent through the material , I_c = %.1f A
   \n", I_c)
printf("\nNOTE: ERROR: Calculation mistake in the
   textbook solution")
```

Scilab code Exa 40.8 Time taken to melt Power factor and Electrical efficiency of

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti , M.L.Soni , P.V.Gupta , U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 2: HEATING AND WELDING

// EXAMPLE : 2.8 :
// Page number 736-737
clear ; clc ; close ; // Clear the work space and
   console

// Given data
weight = 3000.0        // Weight of steel(kg)
I = 5000.0             // Current(A)
V_arc = 60.0           // Arc voltage(V)
R_t = 0.003            // Resistance of transformer(
   ohm)
X_t = 0.005            // Reactance of transformer(
   ohm)
heat_sp = 0.12         // Specific heat of steel
heat_latent = 8.89     // Latent heat of steel(kilo-
   cal/kg)
```

```
t_2 = 1370.0            // Melting point of steel( C )
t_1 = 18.0              // Initial temperature of steel( C )
n = 0.6                 // Overall efficiency

// Calculations
R_arc_phase = V_arc/I                                   // Arc resistance per phase(ohm)
IR_t = I*R_t                                            // Voltage drop across resistance(V)
IX_t = I*X_t                                            // Voltage drop across reactance(V)
V = ((V_arc+IR_t)**2+IX_t**2)**0.5                      // Voltage(V)
PF = (V_arc+IR_t)/V                                     // Power factor
heat_kg = (t_2-t_1)*heat_sp+heat_latent                 // Amount of heat required per kg of steel(kcal)
heat_total = weight*heat_kg                             // Heat for 3 tonnes(kcal)
heat_actual_kcal = heat_total/n                         // Actual heat required(kcal)
heat_actual = heat_actual_kcal*1.162*10**-3             // Actual heat required(kWh)
P_input = 3*V*I*PF*10**-3                               // Power input(kW)
time = heat_actual/P_input*60                           // Time required(min)
n_elect = 3*V_arc*I/(P_input*1000)*100                  // Electrical efficiency(%)

// Results
disp("PART IV - EXAMPLE : 2.8 : SOLUTION :-")
printf("\nTime taken to melt 3 metric tonnes of steel = %.f minutes", time)
printf("\nPower factor of the furnace = %.2f ", PF)
printf("\nElectrical efficiency of the furnace = %.f percent\n", n_elect)
```

```
printf("\nNOTE: ERROR: Calculation and substitution
    mistake in the textbook solution")
```

Chapter 41

ELECTROLYTIC AND ELECTRO METALLURGICAL PROCESSES

Scilab code Exa 41.1 Quantity of electricity and Time taken for the process

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 3: ELECTROLYTIC AND ELECTRO-METALLURGICAL
    PROCESSES

// EXAMPLE : 3.1 :
// Page number 747-748
clear ; clc ; close ; // Clear the work space and
    console

// Given data
```

```
l = 20.0         // Length of shaft(cm)
d = 10.0         // Diameter of shaft(cm)
thick = 1.5      // Layer of nickel(mm)
J = 195.0        // Current density(A/sq.m)
n_I = 0.92       // Current efficiency
g = 8.9          // Specific gravity of nickel

// Calculations
Wt = %pi*l*d*thick/10*g*10**-3          // Weight of nickel to be deposited(kg)
ece_nickel = 1.0954                     // Electro-chemical equivalent of nickel(kg/1000 Ah)
Q_I = Wt*1000/(ece_nickel*n_I)          // Quantity of electricity required(Ah)
time = Q_I/(%pi*l*d*10**-4*J)           // Time taken(hours)

// Results
disp("PART IV - EXAMPLE : 3.1 : SOLUTION :-")
printf("\nQuantity of electricity = %.f Ah", Q_I)
printf("\nTime taken for the process = %.f hours", time)
```

Scilab code Exa 41.2 Annual output of refined copper and Energy consumption

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 3: ELECTROLYTIC AND ELECTRO-METALLURGICAL PROCESSES

// EXAMPLE : 3.2 :
```

```
// Page number 748
clear ; clc ; close ; // Clear the work space and console

// Given data
no_cells = 600.0     // Number of cells employed for copper refining
I = 4000.0           // Current(A)
V = 0.3              // Voltage per cell(V)
hour = 90.0          // Time of plant operation(hours)
ece_cu = 1.1844      // Electro-chemical equivalent of copper(kg/1000 Ah)

// Calculations
Ah_week  = I*hour                              // Ah per week per cell
Ah_year = Ah_week*52                           // Ah per year per cell
Wt  = no_cells*ece_cu*Ah_year/(1000*10**3)    // Weight of copper refined per year(tonnes)
energy = V*I*no_cells*hour*52/1000            // Energy consumed(kWh)
consumption = energy/Wt                        // Consumption(kWh/tonne)

// Results
disp("PART IV - EXAMPLE : 3.2 : SOLUTION :-")
printf("\nAnnual output of refined copper = %.f tonnes", Wt)
printf("\nEnergy consumption = %.1f kWh/tonne\n", consumption)
printf("\nNOTE: ERROR: Substitution & calculation mistake in the textbook solution")
```

Scilab code Exa 41.3 `Weight of aluminium produced from aluminium oxide`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 3: ELECTROLYTIC AND ELECTRO-METALLURGICAL PROCESSES

// EXAMPLE : 3.3 :
// Page number 748
clear ; clc ; close ; // Clear the work space and console

// Given data
hour = 24.0           // Time(hour)
I = 3500.0            // Average current(A)
n = 0.9               // Current efficiency
valency = 3.0         // Aluminium valency
w = 27.0              // Atomic weight of aluminium
ece_Ag = 107.98       // Electro-chemical equivalent of silver
Wt_dep = 0.00111      // Silver deposition by one coulomb(gm)

// Calculations
chemical_eq_Al = w/valency                     // Chemical equivalent of aluminium
eme_Al  = Wt_dep/ece_Ag*chemical_eq_Al         // Electro-chemical equivalent of aluminium(gm/coulomb)
Wt_Al_liberated = I*hour*3600*n*eme_Al/1000    // Weight of aluminium liberated(Kg)

// Results
disp("PART IV - EXAMPLE : 3.3 : SOLUTION :-")
```

```
printf("\nWeight of aluminium produced from
   aluminium oxide = %.1f kg", Wt_Al_liberated)
```

Chapter 42

ILLUMINATION

Scilab code Exa 42.2 `mscp of lamp Illumination on the surface when it is normal In`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 4: ILLUMINATION

// EXAMPLE : 4.2 :
// Page number 753
clear ; clc ; close ; // Clear the work space and console

// Given data
lumens = 800.0          // Flux emitted by a lamp(lumens)
cp = 100.0              // cp of a lamp
d = 2.0                 // Distance b/w plane surface & lamp(m)
theta_ii = 45.0         // Inclined surface(  )
theta_iii = 90.0        // Parallel rays(  )
```

```

// Calculations
// Case(a)
mscp = lumens/(4.0*%pi)              // mscp of lamp
// Case(b)
I_i = cp/d**2                        // Illumination
    on the surface when it is normal(lux)
I_ii = cp/d**2*cosd(theta_ii)        // Illumination
    on the surface when it is inclined to 45  (lux)
I_iii = cp/d**2*cosd(theta_iii)      // Illumination
    on the surface when it is parallel to rays(lux)

// Results
disp("PART IV - EXAMPLE : 4.2 : SOLUTION :-")
printf("\nCase(a): mscp of the lamp, mscp = %.f ",
    mscp)
printf("\nCase(b): Case(i)  : Illumination on the
    surface when it is normal, I = %.f lux", I_i)
printf("\n         Case(ii) : Illumination on the
    surface when it is inclined to 45  , I = %.3f lux
    ", I_ii)
printf("\n         Case(iii): Illumination on the
    surface when it is parallel to rays, I = %.f lux\
    n", abs(I_iii))
printf("\nNOTE: ERROR: Calculation mistake in case(a
    ) in textbook solution")
```

Scilab code Exa 42.3 Illumination at the centre Edge of surface with and Without r

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
```

```
// CHAPTER 4: ILLUMINATION

// EXAMPLE : 4.3 :
// Page number 753-754
clear ; clc ; close ; // Clear the work space and console

// Given data
cp = 200.0          // cp of a lamp
per = 0.6           // Reflector directing light
D = 10.0            // Diameter(m)
h = 6.0             // Height at which lamp is hung(m)

// Calculations
flux = cp*4*%pi                              // Flux(lumens)
I_i = cp/h**2                                // Illumination at the centre without reflector(lux)
d = (h**2+(D/2)**2)**0.5                     // (m)
I_without = (cp/h**2)*(h/d)                  // Illumination at the edge without reflector(lux)
I_with = cp*4*%pi*per/(25*%pi)               // Illumination at the edge with reflector(lux)
theta = acosd(h/d)                           //  ()
w = 2.0*%pi*(1-cosd(theta/2))                //  (steradian)
phi = cp*w                                   //  (lumens)
I_avg = phi/(25*%pi)                         // Average illumination over the area without reflector(lux)

// Results
disp("PART IV - EXAMPLE : 4.3 : SOLUTION :-")
printf("\nCase(i) : Illumination at the centre without reflector = %.2f lux", I_i)
printf("\n          Illumination at the centre with
```

```
reflector = %.1f lux", I_with)
printf("\nCase(ii): Illumination at the edge of the surface without reflector = %.2f lux", I_without)
printf("\n           Illumination at the edge of the surface with reflector = %.1f lux", I_with)
printf("\nAverage illumination over the area without the reflector, I  = %.3f lux\n", I_avg)
printf("\nNOTE: ERROR: Slight calculation mistake & more approximation in textbook solution")
```

Scilab code Exa 42.5 cp of the globe and Percentage of light emitted by lamp that

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 4: ILLUMINATION

// EXAMPLE : 4.5 :
// Page number 754
clear ; clc ; close ; // Clear the work space and console

// Given data
flux = 900.0        // Lamp emitting light(lumens)
D = 30.5            // Diameter of globe(cm)
B = 250.0*10**-3    // Uniform brightness(Ambert)

// Calculations
cp = %pi/4*D**2*(B/%pi)               // Candle power
flux_emit = cp*4*%pi                  // Flux emitted by globe(lumens)
flux_abs = flux-flux_emit             // Flux absorbed
```

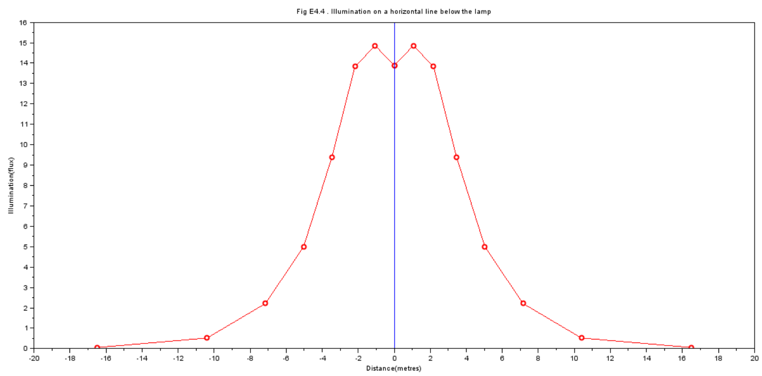

Figure 42.1: Curve showing illumination on a horizontal line below lamp

```
    by globe(lumens)
light_abs_per = flux_abs/flux*100  // Light absorbed
    (%)

// Results
disp("PART IV - EXAMPLE : 4.5 : SOLUTION :-")
printf("\ncp of the globe = %.f ", cp)
printf("\nPercentage of light emitted by lamp that
    is absorbed by the globe = %.1f percent\n",
    light_abs_per)
printf("\nNOTE: Changes in the obtained answer from
    that of textbook is due to more precision here &
    approximation in textbook solution")
```

Scilab code Exa 42.6 `Curve showing illumination on a horizontal line below lamp`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
```

```
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 4: ILLUMINATION

// EXAMPLE : 4.6 :
// Page number 754-755
clear ; clc ; close ; // Clear the work space and console

// Given data
cp_0 = 500.0        // Candle power
theta_0 = 0.0       //   (  )
cp_1 = 560.0        // Candle power
theta_1 = 10.0      //   (  )
cp_2 = 600.0        // Candle power
theta_2 = 20.0      //   (  )
cp_3 = 520.0        // Candle power
theta_3 = 30.0      //   (  )
cp_4 = 400.0        // Candle power
theta_4 = 40.0      //   (  )
cp_5 = 300.0        // Candle power
theta_5 = 50.0      //   (  )
cp_6 = 150.0        // Candle power
theta_6 = 60.0      //   (  )
cp_7 = 50.0         // Candle power
theta_7 = 70.0      //   (  )
h = 6.0             // Height of lamp(m)

// Calculations
I_0 = cp_0/h**2*(cosd(theta_0))**3     // Illumination(lux)
l_0 = h*tand(theta_0)                  // Distance(m)
I_1 = cp_1/h**2*(cosd(theta_1))**3     // Illumination(lux)
l_1 = h*tand(theta_1)                  // Distance(m)
I_2 = cp_2/h**2*(cosd(theta_2))**3     // Illumination(lux)
```

```
l_2 = h*tand(theta_2)                    // Distance(m)
I_3 = cp_3/h**2*(cosd(theta_3))**3       // Illumination(lux)
l_3 = h*tand(theta_3)                    // Distance(m)
I_4 = cp_4/h**2*(cosd(theta_4))**3       // Illumination(lux)
l_4 = h*tand(theta_4)                    // Distance(m)
I_5 = cp_5/h**2*(cosd(theta_5))**3       // Illumination(lux)
l_5 = h*tand(theta_5)                    // Distance(m)
I_6 = cp_6/h**2*(cosd(theta_6))**3       // Illumination(lux)
l_6 = h*tand(theta_6)                    // Distance(m)
I_7 = cp_7/h**2*(cosd(theta_7))**3       // Illumination(lux)
l_7 = h*tand(theta_7)                    // Distance(m)
l = [-l_7,-l_6,-l_5,-l_4,-l_3,-l_2,-l_1,l_0,l_0,l_1,l_2,l_3,l_4,l_5,l_6,l_7]
I = [I_7,I_6,I_5,I_4,I_3,I_2,I_1,I_0,I_0,I_1,I_2,I_3,I_4,I_5,I_6,I_7]
a = gca() ;
a.thickness = 2                                          // sets thickness of plot
plot(l,I,'ro-')                                          // Plot of illumination curve
x = [0,0,0,0,0,0]
y = [0,5,10,11,14,16]
plot(x,y)                                                // Plot of straight line
a.x_label.text = 'Distance(metres)'                      // labels x-axis
a.y_label.text = 'Illumination(flux)'                    // labels y-axis
xtitle("Fig E4.4 . Illumination on a horizontal line below the lamp")
```

```
xset('thickness',2)                                          // sets thickness of axes

// Results
disp("PART IV - EXAMPLE : 4.6 : SOLUTION :-")
printf("\nThe curve showing illumination on a horizontal line below lamp is represented in Figure E4.4")
```

Scilab code Exa 42.7 Maximum and Minimum illumination on the floor along the centr

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 4: ILLUMINATION

// EXAMPLE : 4.7 :
// Page number 755
clear ; clc ; close ; // Clear the work space and console

// Given data
d = 9.15          // Lamp space(m)
h = 4.575         // Height(m)
P = 100.0         // Power(candle)

// Calculations
theta_3_max = 0

   //   (   )
cos_theta_3_max_cubic = cosd(theta_3_max)**3
```

```
theta_4_max = atand(2)     // ( )
cos_theta_4_max_cubic = cosd(theta_4_max)**3
theta_5_max = atand(4)     // ( )
cos_theta_5_max_cubic = cosd(theta_5_max)**3
theta_6_max = atand(6)     // ( )
cos_theta_6_max_cubic = cosd(theta_6_max)**3
I_max = P/h**2*(cos_theta_3_max_cubic+2*cos_theta_4_max_cubic+2*cos_theta_5_max_cubic+2*cos_theta_6_max_cubic)   // Max illumination(lux)
theta_4_min = atand(1)     // ( )
cos_theta_4_min_cubic = cosd(theta_4_min)**3
theta_5_min = atand(3)     // ( )
cos_theta_5_min_cubic = cosd(theta_5_min)**3
theta_6_min = atand(5)     // ( )
cos_theta_6_min_cubic = cosd(theta_6_min)**3
I_min = P/h**2*2*(cos_theta_4_min_cubic+cos_theta_5_min_cubic+cos_theta_6_min_cubic)   // Minimum illumination(lux)

// Results
disp("PART IV - EXAMPLE : 4.7 : SOLUTION :-")
printf("\nMaximum illumination on the floor along the centre line = %.2f lux", I_max)
printf("\nMinimum illumination on the floor along the centre line = %.2f lux", I_min)
```

Scilab code Exa 42.8 Illumination on the working plane

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 4: ILLUMINATION

// EXAMPLE : 4.8 :
// Page number 758
clear ; clc ; close ; // Clear the work space and console

// Given data
b = 15.25      // Breadth of workshop(m)
l = 36.6       // Length of workshop(m)
no = 20.0      // Number of lamps
P = 500.0      // Power of each lamp(W)
n = 15.0       // Luminous efficiency of each lamp(lumens/watt)
df = 0.7       // Depreciation factor
cou = 0.5      // Co-efficient of utilization

// Calculations
lumen_lamp = no*P*n                  // Lamp lumens
lumen_plane = lumen_lamp*df*cou      // Lumens on the working plane
I = lumen_plane/(l*b)                // Illumination(lm/sq.m)

// Results
disp("PART IV - EXAMPLE : 4.8 : SOLUTION :-")
```

```
printf("\nIllumination on the working plane = %.1f lm per sq.m\n", I)
printf("\nNOTE: ERROR: The breadth should be 15.25m but mentioned as 5.25m in textbook statement")
```

Scilab code Exa 42.9 Suitable scheme of illumination and Saving in power consumpti

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 4: ILLUMINATION

// EXAMPLE : 4.9 :
// Page number 758-759
clear ; clc ; close ; // Clear the work space and console

// Given data
b = 27.45         // Breadth of hall(m)
l = 45.75         // Length of hall(m)
I_avg = 108.0     // Average illumination(lumens/sq.m)
h = 0.75          // Height(m)
cou = 0.35        // Co-efficient of utilization
pf = 0.9          // Pereciation factor
P_fl = 80.0       // Fluorescent lamp power(W)
n_100 = 13.4      // Luminous efficiency for 100W filament lamp(lumens/watt)
n_200 = 14.4      // Luminous efficiency for 200W filament lamp(lumens/watt)
n_80 = 30.0       // Luminous efficiency for 80W fluorescent lamp(lumens/watt)

```

```
// Calculations
area = b*l                                                  // Area to be illuminated(Sq.m)
I_total = area*I_avg                                        // Total illumination on working plane(lumens)
gross_lumen = I_total/(cou*pf)                              // Gross lumens required
P_required = gross_lumen/n_200                              // Power required for illumination(W)
P_required_kW = P_required/1000                             // Power required for illumination(kW)
no_lamp = P_required/200                                    // Number of lamps
P_required_new = gross_lumen/n_80                           // Power required when fluorescent lamp used(W)
P_required_new_kW = P_required_new/1000                     // Power required when fluorescent lamp used(kW)
P_saving = P_required_kW-P_required_new_kW                  // Saving in power(kW)

// Results
disp("PART IV - EXAMPLE : 4.9 : SOLUTION :-")
printf("\nSuitable scheme: Whole area divided into %.f rectangles & 200-watt fitting is suspended at centre of each rectangle", no_lamp)
printf("\nSaving in power consumption = %.1f kW", P_saving)
```

Chapter 43

ELECTRIC TRACTION SPEED TIME CURVES AND MECHANICS OF TRAIN MOVEMENT

Scilab code Exa 43.1 Maximum speed over the run

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES
   AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.1 :
// Page number 778
clear ; clc ; close ; // Clear the work space and
   console

// Given data
```

```
speed = 45.0        // Scheduled speed(kmph)
D = 1.5             // Distance between 2 stops(km)
t = 20.0            // Time of stop(sec)
alpha = 2.4         // Acceleration(km phps)
beta = 3.2          // Retardation(km phps)

// Calculations
t_total = D*3600/speed                      // Total time(sec)
T = t_total-t                               // Actual time for run(sec)
k = (alpha+beta)/(alpha*beta)               // Constant
V_m = (T/k)-((T/k)**2-(7200*D/k))**0.5     // Maximum speed over the run(kmph)

// Results
disp("PART IV - EXAMPLE : 5.1 : SOLUTION :-")
printf("\nMaximum speed over the run, V_m = %.f kmph", V_m)
```

Scilab code Exa 43.2 Value of retardation

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.2 :
// Page number 778
clear ; clc ; close ; // Clear the work space and console
```

```

// Given data
V_m = 65.0          // Maximum speed(kmph)
t = 30.0            // Time of stop(sec)
speed = 43.5        // Scheduled speed(kmph)
alpha = 1.3         // Acceleration(km phps)
D = 3.0             // Distance between 2 stops(km)

// Calculations
t_total = D*3600/speed                                  // Total time of run including stop(sec)
T = t_total-t                                                    // Actual time for run(sec)
V_a = D/T*3600                                                  // Average speed(kmph)
beta = 1/((7200.0*D/V_m**2*((V_m/V_a)-1))-(1/alpha))   // Value of retardation(km phps)

// Results
disp("PART IV - EXAMPLE : 5.2 : SOLUTION :-")
printf("\nValue of retardation ,    = %.3f km phps\n", beta)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
printf("\n       ERROR:     unit is km phps & not km phps as mentioned in textbook solution")
```

Scilab code Exa 43.3 Rate of acceleration required to operate service

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
```

```
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION−SPEED TIME CURVES AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.3 :
// Page number 778−779
clear ; clc ; close ; // Clear the work space and console

// Given data
speed = 25.0          // Scheduled speed(kmph)
D = 800.0/1000        // Distance between 2 stations(km)
t = 20.0              // Time of stop(sec)
V_m_per = 20.0        // Maximum speed higher than(%)
beta = 3.0            // Retardation(km phps)

// Calculations
t_total = D*3600/speed                                    // Total time of run including stop(sec)
T = t_total-t                                                     // Actual time for run(sec)
V_a = D/T*3600                                                    // Average speed(kmph)
V_m = (100+V_m_per)*V_a/100                          // Maximum speed(kmph)
alpha = 1/((7200.0*D/V_m**2*((V_m/V_a)-1))-(1/beta))   // Value of acceleration(km phps)

// Results
disp("PART IV − EXAMPLE : 5.3 : SOLUTION :−")
printf("\nRate of acceleration required to operate this service,    = %.2f km phps", alpha)
```

Scilab code Exa 43.4 Duration of acceleration Coasting and Braking periods

```
// A Textbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES
   AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.4 :
// Page number 779
clear ; clc ; close ; // Clear the work space and
   console

// Given data
D = 2.0              // Distance between 2 stations(km)
V_a = 40.0           // Average speed(kmph)
V_1 = 60.0           // Maximum speed limitation(kph)
alpha = 2.0          // Acceleration(km phps)
beta_c = 0.15        // Coasting retardation(km phps)
beta = 3.0           // Braking retardation(km phps)

// Calculations
t_1 = V_1/alpha
                                                    // Time
   for acceleration(sec)
T = 3600*D/V_a
                                                    // Actual
    time of run(sec)
V_2 = (T-t_1-(V_1/beta_c))*beta*beta_c/(beta_c-beta)
      // Speed at the end of coasting period(kmph)
t_2 = (V_1-V_2)/beta_c
```

```
                                                    // Coasting
   period(sec)
t_3 = V_2/beta
                                                            //
   Braking period(sec)

// Results
disp("PART IV - EXAMPLE : 5.4 : SOLUTION :-")
printf("\nDuration of acceleration, t_1 = %.f sec",
   t_1)
printf("\nDuration of coasting, t_2 = %.f sec", t_2)
printf("\nDuration of braking, t_3 = %.f sec", t_3)
```

Scilab code Exa 43.5 Tractive resistance

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES
   AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.5 :
// Page number 781-782
clear ; clc ; close ; // Clear the work space and
   console

// Given data
r = 1.0     // Tractive resistance(N/tonne)

// Calculations
tractive_res_i = 0.278*r     // Tractive resistance(
   N/tonne) = Energy consumption(Wh/tonne-km)
```

```
beta = 1/277.8                    // Tractive resistance(
   N/tonne) = Retardation(km kmps/tonne)
energy = 98.1*1000/3600           // 1% gradient = energy
   (Wh per tonne km)

// Results
disp("PART IV - EXAMPLE : 5.5 : SOLUTION :-")
printf("\nCase(i)   : Tractive resistance of 1 N per
   tonne = %.3f Wh per tonne-km", tractive_res_i)
printf("\nCase(ii)  : Tractive resistance of 1 N per
   tonne = %.5f km phps per tonne", beta)
printf("\nCase(iii): 1 percent gradient = %.2f Wh
   per tonne km\n", energy)
printf("\nNOTE: Slight change in the obtained answer
    from that of textbook is due to more precision
   here")
```

Scilab code Exa 43.6 Torque developed by each motor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES
   AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.6 :
// Page number 782
clear ; clc ; close ; // Clear the work space and
   console

// Given data
W = 254.0          // Weight of motor-coach train(tonne)
```

```
no = 4.0          // Number of motor
t_1 = 20.0        // Time(sec)
V_m = 40.25       // Maximum speed(kmph)
G = 1.0           // Gradient(%)
gamma = 3.5       // Gear ratio
n = 0.95          // Gear efficiency
D = 91.5/100      // Wheel diameter(m)
r = 44.0          // Train resistance(N/tonne)
I = 10.0          // Rotational inertia(%)

// Calculations
W_e = W*(100+I)/100                    // Accelerating weight of train(tonne)
alpha = V_m/t_1                        // Acceleration (km phps)
F_t = 277.8*W_e*alpha+W*r+98.1*W*G     // Tractive effort(N)
T = F_t*D/(2*n*gamma)                  // Torque developed(N-m)
T_each = T/no                          // Torque developed by each motor(N-m)

// Results
disp("PART IV - EXAMPLE : 5.6 : SOLUTION :-")
printf("\nTorque developed by each motor = %.f N-m\n", T_each)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here & more approximation in textbook")
printf("\n      ERROR: W = 254 tonne, not 256 tonne as mentioned in textbook problem statement")
```

Scilab code Exa 43.7 `Time taken by train to attain speed`

```
// A Texbook on POWER SYSTEM ENGINEERING
```

```
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.7 :
// Page number 782
clear ; clc ; close ; // Clear the work space and console

// Given data
W = 203.0        // Weight of motor-coach train(tonne)
no = 4.0         // Number of motors
T = 5130.0       // Shaft torque(N-m)
V_m = 42.0       // Maximum speed(kmph)
G = 100.0/250    // Gradient
gamma = 3.5      // Gear ratio
n = 0.93         // Gear efficiency
D = 91.5/100     // Wheel diameter(m)
r = 45.0         // Train resistance(N/tonne)
I = 10.0         // Rotational inertia(%)

// Calculations
W_e = W*(100+I)/100                      // Accelerating weight of train(tonne)
F_t = n*4*T*2*gamma/D                    // Tractive effort(N)
alpha = (F_t-W*r-98.1*W*G)/(277.8*W_e)   // Acceleration(km phps)
t_1 = V_m/alpha                          // Time taken by train to attain speed(sec)

// Results
disp("PART IV - EXAMPLE : 5.7 : SOLUTION :-")
printf("\nTime taken by train to attain speed, t_1 =
```

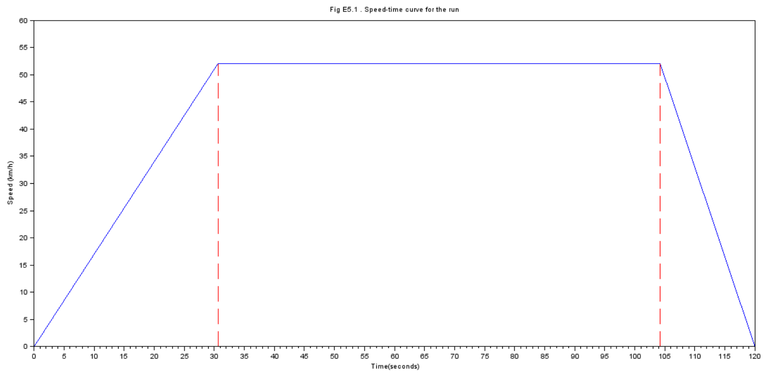

Figure 43.1: Speed Time curve for the run and Energy consumption at the axles of train

```
    %.1f sec", t_1)
```

Scilab code Exa 43.8 Speed Time curve for the run and Energy consumption at the ax

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES
   AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.8 :
// Page number 782-783
clear ; clc ; close ; // Clear the work space and
   console

```

```
// Given data
V_a = 42.0            // Average speed of train(kmph)
D = 1400.0/1000       // Distance(km)
alpha = 1.7           // Acceleration(km phps)
beta = 3.3            // Retardation(km phps)
r = 50.0              // Tractive resistance(N/tonne)
I = 10.0              // Rotational inertia(%)

// Calculations
T = D*3600/V_a                                                    // Time for run(sec)
k = (alpha+beta)/(alpha*beta)                         // Constant
V_m = (T/k)-((T/k)**2-(7200*D/k))**0.5           // Maximum speed over the run(kmph)
t_1 = V_m/alpha                                                  // Time of acceleration(sec)
t_3 = V_m/beta                                                    // Time(sec)
t_2 = T-(t_1+t_3)                                             // Time(sec)
D_1 = D-(V_a*t_1/(2*3600))                          // Distance(km)
We_W = (100+I)/100                                          // W_e/W
energy = (0.0107*V_m**2*We_W/D)+(0.278*r*D_1/D)          // Energy consumption(Wh per tonne-km)
a = gca() ;
a.thickness = 2                                                  // sets thickness of plot
plot([0,t_1,t_1,(t_1+t_2),(t_1+t_2),(t_1+t_2+t_3)],[0,V_m,V_m,V_m,V_m,0])        // Plotting speed-time curve
plot([t_1,t_1],[0,V_m],'r--')
```

```
plot([t_1+t_2,t_1+t_2],[0,V_m],'r--')
a.x_label.text = 'Time(seconds)'
                      // labels x-axis
a.y_label.text = 'Speed (km/h)'
                       // labels y-axis
xtitle("Fig E5.1 . Speed-time curve for the run")
xset('thickness',2)
                                    // sets
   thickness of axes

// Results
disp("PART IV - EXAMPLE : 5.8 : SOLUTION :-")
printf("\nSpeed-time curve for the run is shown in
   Figure E5.1")
printf("\nEnergy consumption at the axles of train =
   %.1f Wh per tonne-km", energy)
```

Scilab code Exa 43.9 `Acceleration Coasting retardation and Scheduled speed`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES
   AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.9 :
// Page number 783
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V_A = 48.0          // Speed(kmph)
```

```
t_1 = 24.0        // Time taken to accelerate from rest to speed(sec)
t_2 = 69.0        // Coasting time(sec)
r = 58.0          // Constant resistance(N/tonne)
beta = 3.3        // Retardation(km phps)
t_3 = 11.0        // Retardation time(sec)
t_iii_a = 20.0    // Station stop time(sec)
t_iii_b = 15.0    // Station stop time(sec)
I = 10.0          // Rotational inertia(%)

// Calculations
alpha = V_A/t_1                                                   // Acceleration(km phps)
V_B = beta*t_3                                                    // Speed at B(km phps)
beta_c = (V_A-V_B)/t_2                                            // Retardation during coasting(km phps)
distance_acc = 1.0/2*t_1*V_A/3600                                 // Distance covered during acceleration(km)
distance_coasting = (V_A**2-V_B**2)/(2*beta_c*3600)               // Distance covered during coasting(km)
distance_braking = t_3*V_B/(3600*2)                               // Distance covered during braking(km)
distance_total = distance_acc+distance_coasting+distance_braking  // Total distance(km)
speed_iii_a = distance_total*3600/(t_1+t_2+t_3+t_iii_a)           // Scheduled speed with a stop of 20 sec(kmph)
speed_iii_b = distance_total*3600/(t_1+t_2+t_3+t_iii_b)           // Scheduled speed with a stop of 15 sec(kmph)

```

```
// Results
disp("PART IV - EXAMPLE : 5.9 : SOLUTION :-")
printf("\nCase(i)  : Acceleration ,     = %.f km phps", alpha)
printf("\nCase(ii) : Coasting retardation ,  _c  = %.2f km phps", beta_c)
printf("\nCase(iii): Scheduled speed with a stop of 20 seconds = %.2f kmph", speed_iii_a)
printf("\n           Scheduled speed with a stop of 15 seconds = %.2f kmph\n", speed_iii_b)
printf("\nNOTE: ERROR: Calculation mistakes in the textbook solution")
```

Scilab code Exa 43.10 `Minimum adhesive weight of the locomotive`

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.10 :
// Page number 784
clear ; clc ; close ; // Clear the work space and console

// Given data
W = 350.0        // Weight of train(tonne)
G = 1.0          // Gradient
alpha = 0.8      // Acceleration(km phps)
u = 0.25         // Co-efficient of adhesion
r = 44.5         // Train resistance(N/tonne)
```

```
I = 10.0          // Rotational inertia(%)

// Calculations
W_e = W*(100+I)/100                        // Accelerating weight of train(tonne)
F_t = 277.8*W_e*alpha+W*r+98.1*W*G    // Tractive effort(N)
adhesive_weight = F_t/(u*9.81*1000)   // Adhesive weight(tonnes)

// Results
disp("PART IV - EXAMPLE : 5.10 : SOLUTION :-")
printf("\nMinimum adhesive weight of the locomotive = %.1f tonnes\n", adhesive_weight)
printf("\nNOTE: ERROR: Train resistance is 44.5 N per tonne & not 45 N per tonne as mentioned in textbook problem statement")
```

Scilab code Exa 43.11 Energy usefully employed in attaining speed and Specific ene

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.11 :
// Page number 784
clear ; clc ; close ; // Clear the work space and console

// Given data
```

```
W = 400.0          // Weight of train(tonne)
G = 100.0/75       // Gradient
alpha = 1.6        // Acceleration(km phps)
r = 66.75          // Train resistance(N/tonne)
I = 10.0           // Rotational inertia(%)
V = 48.0           // Speed(kmph)
n = 0.7            // Overall efficiency of equipment

// Calculations
W_e = W*(100+I)/100                      // Accelerating
    weight of train(tonne)
F_t = 277.8*W_e*alpha+W*r+98.1*W*G    // Tractive
   effort(N)
t = V/alpha                              // Time(sec)
energy_a = F_t*V*t/(2*3600**2)           // Energy
   usefully employed(kWh)
G_r = 98.1*G+r                           // Force(N)
work_tonne_km = G_r*1000                 // Work done
   per tonne per km(Nw-m)
energy_b = work_tonne_km/(n*3600)        // Energy
   consumption(Wh per tonne-km)

// Results
disp("PART IV - EXAMPLE : 5.11 : SOLUTION :-")
printf("\nCase(a): Energy usefully employed in
   attaining speed = %.2f kWh", energy_a)
printf("\nCase(b): Specific energy consumption at
   steady state speed = %.1f Wh per tonne-km",
   energy_b)
```

Scilab code Exa 43.12 Minimum adhesive weight of a locomotive

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
```

```
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 5: ELECTRIC TRACTION-SPEED TIME CURVES AND MECHANICS OF TRAIN MOVEMENT

// EXAMPLE : 5.12 :
// Page number 784-785
clear ; clc ; close ; // Clear the work space and console

// Given data
W = 200.0        // Trailing weight(tonne)
G = 1.0          // Gradient(%)
alpha = 1.0      // Acceleration(km phps)
u = 0.2          // Co-efficient of adhesion
r = 50.0         // Train resistance(N/tonne)
I = 10.0         // Rotational inertia(%)

// Calculations
W_L = ((277.8*(100+I)/100*alpha)+98.1*G+r)*W/(u*9.81*1000-((277.8*(100+I)/100*alpha)+98.1*G+r))  // Weight of locomotive(tonnes)

// Results
disp("PART IV - EXAMPLE : 5.12 : SOLUTION :-")
printf("\nMinimum adhesive weight of a locomotive, W_L = %.1f tonnes\n", W_L)
printf("\nNOTE: ERROR: Calculation mistake in textbook solution in calculating W_L")
```

Chapter 44

MOTORS FOR ELECTRIC TRACTION

Scilab code Exa 44.1 Speed current of the motor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 6: MOTORS FOR ELECTRIC TRACTION

// EXAMPLE : 6.1 :
// Page number 788
clear ; clc ; close ; // Clear the work space and
    console

// Given data
I_1 = 10.0      // Current(A)
T_1 = 54.0      // Torque(N-m)
I_2 = 20.0      // Current(A)
T_2 = 142.0     // Torque(N-m)
I_3 = 30.0      // Current(A)
```

```
T_3 = 250.0    // Torque(N-m)
I_4 = 40.0     // Current(A)
T_4 = 365.0    // Torque(N-m)
I_5 = 50.0     // Current(A)
T_5 = 480.0    // Torque(N-m)
I_6 = 60.0     // Current(A)
T_6 = 620.0    // Torque(N-m)
I_7 = 70.0     // Current(A)
T_7 = 810.0    // Torque(N-m)
E = 500.0      // Operating voltage(V)
R_a = 0.6      // Armature resistance(ohm)

// Calculations
N_1 = 9.55*(E-I_1*R_a)*I_1/T_1    // Speed(rpm)
N_2 = 9.55*(E-I_2*R_a)*I_2/T_2    // Speed(rpm)
N_3 = 9.55*(E-I_3*R_a)*I_3/T_3    // Speed(rpm)
N_4 = 9.55*(E-I_4*R_a)*I_4/T_4    // Speed(rpm)
N_5 = 9.55*(E-I_5*R_a)*I_5/T_5    // Speed(rpm)
N_6 = 9.55*(E-I_6*R_a)*I_6/T_6    // Speed(rpm)
N_7 = 9.55*(E-I_7*R_a)*I_7/T_7    // Speed(rpm)

// Results
disp("PART IV - EXAMPLE : 6.1 : SOLUTION :-")
printf("\nSpeed-current of the motor")
printf("\n________________________________________")
printf("\n Current(A)          :       Speed(rpm)   ")
printf("\n________________________________________")
printf("\n   %.f                :          %.f ", I_1,
   N_1)
printf("\n   %.f                :          %.f ", I_2,
   N_2)
printf("\n   %.f                :          %.f ", I_3,
   N_3)
printf("\n   %.f                :          %.f ", I_4,
   N_4)
printf("\n   %.f                :          %.f ", I_5,
   N_5)
printf("\n   %.f                :          %.f ", I_6,
```

```
    N_6)
printf("\n    %.f                    :          %.f ", I_7,
    N_7)
printf("\n---------------------------------------\n"
    )
printf("\nNOTE: ERROR: Calculation mistakes in the
    textbook solution")
```

Scilab code Exa 44.2 Speed torque for motor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 6: MOTORS FOR ELECTRIC TRACTION

// EXAMPLE : 6.2 :
// Page number 788-789
clear ; clc ; close ; // Clear the work space and
    console

// Given data
N_1 = 500.0      // Speed(rpm)
I_1 = 50.0       // Current(A)
E_1 = 220.0      // Armature voltage(V)
I_2 = 100.0      // Current(A)
E_2 = 350.0      // Armature voltage(V)
I_3 = 150.0      // Current(A)
E_3 = 440.0      // Armature voltage(V)
I_4 = 200.0      // Current(A)
E_4 = 500.0      // Armature voltage(V)
I_5 = 250.0      // Current(A)
E_5 = 540.0      // Armature voltage(V)
```

```
I_6 = 300.0      // Current(A)
E_6 = 570.0      // Armature voltage(V)
R_wb = 0.08      // Armature and brush resistance(ohm)
R_f = 0.05       // Resistance of series field(ohm)
V = 600.0        // Operating voltage(V)

// Calculations
R_a = R_wb+R_f                   // Armature resistance(ohm)
N_11 = N_1/E_1*(V-I_1*R_a)      // Speed(rpm)
T_1 = 9.55*E_1*I_1/N_1          // Torque(N-m)
N_2 = N_1/E_2*(V-I_2*R_a)       // Speed(rpm)
T_2 = 9.55*E_2*I_2/N_1          // Torque(N-m)
N_3 = N_1/E_3*(V-I_3*R_a)       // Speed(rpm)
T_3 = 9.55*E_3*I_3/N_1          // Torque(N-m)
N_4 = N_1/E_4*(V-I_4*R_a)       // Speed(rpm)
T_4 = 9.55*E_4*I_4/N_1          // Torque(N-m)
N_5 = N_1/E_5*(V-I_5*R_a)       // Speed(rpm)
T_5 = 9.55*E_5*I_5/N_1          // Torque(N-m)
N_6 = N_1/E_6*(V-I_6*R_a)       // Speed(rpm)
T_6 = 9.55*E_6*I_6/N_1          // Torque(N-m)

// Results
disp("PART IV - EXAMPLE : 6.2 : SOLUTION :-")
printf("\nSpeed-torque curve for motor")
printf("\n_______________________________________")
printf("\n Speed(rpm)              :       Torque(N-m)   ")
printf("\n_______________________________________")
printf("\n    %.f                  :          %.f ", N_11, T_1)
printf("\n    %.f                   :          %.f ", N_2, T_2)
printf("\n    %.f                   :          %.f ", N_3, T_3)
printf("\n    %.f                   :          %.f ", N_4, T_4)
printf("\n    %.f                   :          %.f ", N_5, T_5)
```

```
printf("\n    %.f                    :          %.f ", N_6,
   T_6)
printf("\n------------------------------------\n"
   )
printf("\nNOTE: ERROR: Calculation mistakes in the
   textbook solution")
```

Scilab code Exa 44.3 Speed of motors when connected in series

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 6: MOTORS FOR ELECTRIC TRACTION

// EXAMPLE : 6.3 :
// Page number 790
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 650.0       // Voltage supply(V)
r_A = 45.0      // Radius of driving wheel(cm)
r_B = 43.0      // Radius of driving wheel(cm)
N_A = 400.0     // Speed(rpm)
drop = 10.0     // Voltage drop(%)

// Calculations
rho = r_B/r_A
IR = drop*V/100                   // Voltage drop(V)
V_A = (rho*(V-IR)+IR)/(1+rho)     // Voltage(V)
V_B = V-V_A                       // Voltage(V)
N_A_A = N_A*(V_A-IR)/(V-IR)       // N"_A(rpm)
```

```
N_B_B = N_A_A*r_A/r_B            // N"_B(rpm)

// Results
disp("PART IV - EXAMPLE : 6.3 : SOLUTION :-")
printf("\nSpeed of first motor when connected in series, N_A = %.f rpm", N_A_A)
printf("\nSpeed of second motor when connected in series, N_B = %.f rpm\n", N_B_B)
printf("\nNOTE: Changes in the obtained answer from that of textbook is due to more precision here")
```

Scilab code Exa 44.4 HP delivered by the locomotive when dc series motor and Induc

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 6: MOTORS FOR ELECTRIC TRACTION

// EXAMPLE : 6.4 :
// Page number 791
clear ; clc ; close ; // Clear the work space and console

// Given data
F_t = 33800.0    // Tractive effort(N)
V = 48.3         // Velocity(kmph)
T = 53400.0      // Tractive effort(N)

// Calculations
HP = F_t*V*1000/(60*60*746)     // HP on level track(hp)
HP_i = HP*(T/F_t)**0.5          // hp delivered by
```

```
    locomotive for dc series motor(hp)
HP_ii = HP*T/F_t                    // hp delivered by
    locomotive for induction motor(hp)

// Results
disp("PART IV - EXAMPLE : 6.4 : SOLUTION :-")
printf("\nhp delivered by the locomotive when dc
    series motor is used = %.f HP", HP_i)
printf("\nhp delivered by the locomotive when
    induction motor is used = %.f HP", HP_ii)
```

Scilab code Exa 44.5 New characteristics of motor

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 6: MOTORS FOR ELECTRIC TRACTION

// EXAMPLE : 6.5 :
// Page number 792-793
clear ; clc ; close ; // Clear the work space and
    console

// Given data
I_1 = 100.0                    // Current(A)
N_1 = 71.0                     // Speed(kmph)
F_t1 = 2225.0                  // Tractive effort(N)
I_2 = 150.0                    // Current(A)
N_2 = 57.0                     // Speed(kmph)
F_t2 = 6675.0                  // Tractive effort(N)
I_3 = 200.0                    // Current(A)
N_3 = 50.0                     // Speed(kmph)
```

```
F_t3 = 11600.0              // Tractive effort(N)
I_4 = 250.0                 // Current(A)
N_4 = 45.0                  // Speed(kmph)
F_t4 = 17350.0              // Tractive effort(N)
I_5 = 300.0                 // Current(A)
N_5 = 42.0                  // Speed(kmph)
F_t5 = 23200.0              // Tractive effort(N)
D_A = 101.6                 // Size of wheels(cm)
ratio_gear = 72.0/23        // Gear ratio
D_B = 106.7                 // Size of wheels(cm)
ratio_gear_new = 75.0/20    // Gear ratio

// Calculations
N_B = ratio_gear*D_B/(ratio_gear_new*D_A)      //
   Speed in terms of V(kmph)
F_tB = D_A*ratio_gear_new/(ratio_gear*D_B)     //
   Tractive effort in terms of F_tA(N)
N_B1 = N_B*N_1                                 //
   Speed(kmph)
F_tB1 = F_tB*F_t1                              //
   Tractive effort(N)
N_B2 = N_B*N_2                                 //
   Speed(kmph)
F_tB2 = F_tB*F_t2                              //
   Tractive effort(N)
N_B3 = N_B*N_3                                 //
   Speed(kmph)
F_tB3 = F_tB*F_t3                              //
   Tractive effort(N)
N_B4 = N_B*N_4                                 //
   Speed(kmph)
F_tB4 = F_tB*F_t4                              //
   Tractive effort(N)
N_B5 = N_B*N_5                                 //
   Speed(kmph)
F_tB5 = F_tB*F_t5                              //
   Tractive effort(N)

```

```
// Results
disp("PART IV - EXAMPLE : 6.5 : SOLUTION :-")
printf("\nNew characteristics of motor")
printf("\n_______________________________________")
printf("\n Current(A)  :  Speed(kmph)  :  F_t(N)")
printf("\n_______________________________________")
printf("\n  %.f          :    %.1f           :    %.f  ",
   I_1,N_B1,F_tB1)
printf("\n  %.f          :    %.1f           :    %.f  ",
   I_2,N_B2,F_tB2)
printf("\n  %.f          :    %.1f           :    %.f  ",
   I_3,N_B3,F_tB3)
printf("\n  %.f          :    %.1f           :    %.f  ",
   I_4,N_B4,F_tB4)
printf("\n  %.f          :    %.1f           :    %.f  ",
   I_5,N_B5,F_tB5)
printf("\n_______________________________________\n"
   )
printf("\nNOTE: Changes in the obtained answer from
   that of textbook is due to more precision here")
```

Chapter 45

CONTROL OF MOTORS

Scilab code Exa 45.1 Approximate loss of energy in starting rheostats

```
// A Textbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 7: CONTROL OF MOTORS

// EXAMPLE : 7.1 :
// Page number 798
clear ; clc ; close ; // Clear the work space and console

// Given data
no = 2.0            // Number of motors
V_m = 48.0          // Uniform speed(kmph)
t = 30.0            // Time(sec)
F_t_m = 13350.0     // Average tractive effort per motor(N)

// Calculations
```

```
F_t = no*F_t_m                  // Average tractive effort(N)
energy = t*F_t*V_m/(2*3600**2)  // Useful energy for acceleration(kWh)
energy_loss = energy/no         // Approximate loss of energy in starting rheostats(kWh)

// Results
disp("PART IV - EXAMPLE : 7.1 : SOLUTION :-")
printf("\nApproximate loss of energy in starting rheostats = %.3f kWh", energy_loss)
```

Scilab code Exa 45.2 Energy supplied during the starting period Energy lost in the

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 7: CONTROL OF MOTORS

// EXAMPLE : 7.2 :
// Page number 798
clear ; clc ; close ; // Clear the work space and console

// Given data
W = 175.0         // Weight of multiple unit train(tonnes)
no = 6.0          // Number of motors
F_t = 69000.0     // Total tractive effort(N)
V = 600.0         // Line voltage(V)
I = 200.0         // Average current(A)
V_m = 38.6        // Speed(kmph)
```

```
R = 0.15           // Resistance of each motor(ohm)

// Calculations
alpha = F_t/(277.8*W)                              // Acceleration(km phps)
T = V_m/alpha                                      // Time for acceleration(sec)
t_s = (V-2*I*R)*T/(2*(V-I*R))                      // Duration of starting period(sec)
t_p = T-t_s                                        // (sec)
energy_total_series = no/2*V*I*t_s                 // Total energy supplied in series position(watt-sec)
energy_total_parallel = no*V*I*t_p                 // Total energy supplied in parallel position(watt-sec)
total_energy = (energy_total_series+energy_total_parallel)/(1000*3600)          // Energy supplied during starting period(kWh)
energy_waste_series = (no/2)/2*(V-2*I*R)*I*t_s     // Energy wasted in starting resistance in series position(watt-sec)
energy_waste_parallel = no*(V/2)/2*I*t_p           // Energy wasted in starting resistance in parallel position(watt-sec)
total_energy_waste = (energy_waste_series+energy_waste_parallel)/(1000*3600)    // Total energy wasted in starting resistance(kWh)
energy_lost = (no*I**2*R*T)/(1000*3600)            // Energy lost in motor resistance(kWh)
useful_energy = T*F_t*V_m/(2*3600**2)              // Useful energy supplied to train(kWh)

// Results
disp("PART IV - EXAMPLE : 7.2 : SOLUTION :-")
printf("\nEnergy supplied during the starting period = %.2f kWh", total_energy)
```

```
printf("\nEnergy lost in the starting resistance = %.1f kWh", total_energy_waste)
printf("\nUseful energy supplied to the train = %.1f kWh", useful_energy)
```

Scilab code Exa 45.3 Duration of starting period Speed of train at transition Rheo

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 7: CONTROL OF MOTORS

// EXAMPLE : 7.3 :
// Page number 799
clear ; clc ; close ; // Clear the work space and console

// Given data
W = 132.0            // Weight of electric train(tonnes)
no = 4.0             // Number of motors
V = 600.0            // Voltage of motor(V)
I = 400.0            // Current per motor(A)
F_t_m = 19270.0      // Tractive effort per motor at 400A & 600V(N)
V_m = 39.0           // Train speed(kmph)
G = 1.0              // Gradient
r = 44.5             // Resistance to traction(N/tonne)
inertia = 10.0       // Rotational inertia(%)
R = 0.1              // Resistance of each motor(ohm)

```

```
// Calculations
W_e = W*(100+inertia)/100                                    // Accelerating weight of train(tonne)
F_t = F_t_m*no                                               // Total tractive effort at 400A & 600V(N)
alpha = (F_t-W*r-98.1*W*G)/(277.8*W_e)                       // Acceleration(km phps)
T = V_m/alpha                                                // Time for acceleration(sec)
t_s = (V-2*I*R)*T/(2*(V-I*R))                                // Duration of starting period(sec)
V_transition = alpha*t_s                                     // Speed at transition(km phps)
t_p = T-t_s                                                  // (sec)
loss_series = (no/2*((V-2*I*R)/2)*I*t_s)/(1000*3600)         // Energy lost during series period(kWh)
loss_parallel = (no*(V/2)/2*I*t_p)/(1000*3600)               // Energy lost during parallel period(kWh)

// Results
disp("PART IV - EXAMPLE : 7.3 : SOLUTION :-")
printf("\nCase(i)  : Duration of starting period, t_s = %.1f sec", t_s)
printf("\nCase(ii) : Speed of train at transition, t  = %.1f sec", V_transition)
printf("\nCase(iii): Case(a): Rheostatic losses during series starting = %.2f kWh", loss_series)
printf("\n           Case(b): Rheostatic losses during parallel starting = %.2f kWh\n", loss_parallel)
```

```
printf("\nNOTE: ERROR: Calculation mistakes in the
   textbook solution")
```

Chapter 46

BRAKING

Scilab code Exa 46.1 Braking torque

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 8: BRAKING

// EXAMPLE : 8.1 :
// Page number 806
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 525.0     // Voltage of motor(V)
I_1 = 50.0    // Current(A)
T_1 = 216.0   // Torque(N-m)
I_2 = 70.0    // Current(A)
T_2 = 344.0   // Torque(N-m)
I_3 = 80.0    // Current(A)
T_3 = 422.0   // Torque(N-m)
```

```
I_4 = 90.0    // Current(A)
T_4 = 500.0   // Torque(N-m)
V_m = 26.0    // Speed(kmph)
R_b = 5.5     // Resistance of braking rheostat(ohm)
R_m = 0.5     // Resistance of motor(ohm)

// Calculations
I = 75.0                  // Current drawn at 26 kmph(
   A)
back_emf = V-I*R_m        // Back emf of the motor(V)
R_t = R_b+R_m             // Total resistance(ohm)
I_del = back_emf/R_t      // Current delivered(A)
T_b = T_3*I_del/I_3       // Braking torque(N-m)

// Results
disp("PART IV - EXAMPLE : 8.1 : SOLUTION :-")
printf("\nBraking torque = %.f N-m", T_b)
```

Scilab code Exa 46.2 Current delivered when motor works as generator

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 8: BRAKING

// EXAMPLE : 8.2 :
// Page number 806
clear ; clc ; close ; // Clear the work space and
   console

// Given data
V = 525.0       // Voltage of motor(V)
```

```
I_1 = 50.0      // Current(A)
N_1 = 1200.0    // Speed(rpm)
I_2 = 100.0     // Current(A)
N_2 = 950.0     // Speed(rpm)
I_3 = 150.0     // Current(A)
N_3 = 840.0     // Speed(rpm)
I_4 = 200.0     // Current(A)
N_4 = 745.0     // Speed(rpm)
N = 1000.0      // Speed opearting(rpm)
R = 3.0         // Resistance(ohm)
R_m = 0.5       // Resistance of motor(ohm)

// Calculations
I = 85.0                 // Current drawn at 1000 rpm (A)
back_emf = V-I*R_m       // Back emf of the motor(V)
R_t = R+R_m              // Total resistance(ohm)
I_del = back_emf/R_t     // Current delivered(A)

// Results
disp("PART IV - EXAMPLE : 8.2 : SOLUTION :-")
printf("\nCurrent delivered when motor works as generator = %.f A", I_del)
```

Scilab code Exa 46.3 Energy returned to lines

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 8: BRAKING

// EXAMPLE : 8.3 :
```

```
// Page number 810
clear ; clc ; close ; // Clear the work space and
    console

// Given data
W = 400.0          // Weight of train(tonne)
G = 100.0/70       // Gradient(%)
t = 120.0          // Time(sec)
V_1 = 80.0         // Speed(km/hr)
V_2 = 50.0         // Speed(km/hr)
r_kg = 5.0         // Tractive resistance(kg/tonne)
I = 7.5            // Rotational inertia(%)
n = 0.75           // Overall efficiency

// Calculations
W_e = W*(100+I)/100                                        //
    Accelerating weight of train(tonne)
r = r_kg*9.81                                                    //
    Tractive resistance(N-m/tonne)
energy_recuperation = 0.01072*W_e*(V_1**2-V_2**2)
    /1000    // Energy available for recuperation(kWh)
F_t = W*(r-98.1*G)                                          // Tractive
     effort during retardation(N)
distance = (V_1+V_2)*1000*t/(2*3600)
                              // Distance travelled by
    train during retardation period(m)
energy_train = abs(F_t)*distance/(3600*1000)
                      // Energy available during train
    movement(kWh)
net_energy = n*(energy_recuperation+energy_train)
                 // Net energy returned to supply system(
    kWh)

// Results
disp("PART IV - EXAMPLE : 8.3 : SOLUTION :-")
```

```
printf("\nEnergy returned to lines = %.2f kWh\n",
   net_energy)
printf("\nNOTE: ERROR: Calculation mistakes & more
   approximation in textbook solution")
```

Scilab code Exa 46.4 Energy returned to the line

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 8: BRAKING

// EXAMPLE : 8.4 :
// Page number 810
clear ; clc ; close ; // Clear the work space and
   console

// Given data
W = 355.0           // Weight of train(tonne)
V_1 = 80.5          // Speed(km/hr)
V_2 = 48.3          // Speed(km/hr)
D = 1.525           // Distance(km)
G = 100.0/90        // Gradient(%)
I = 10.0            // Rotational inertia(%)
r = 53.0            // Tractive resistance(N/tonne)
n = 0.8             // Overall efficiency

// Calculations
beta = (V_1**2-V_2**2)/(2*D*3600)    // Braking
   retardation(km phps)
W_e = W*(100+I)/100                  // Accelerating
   weight of train(tonne)
```

```
F_t = 277.8*W_e*beta+98.1*W*G-W*r    // Tractive effort(N)
work_done = F_t*D*1000               // Work done by this effort(N-m)
energy = work_done*n/(1000*3600)     // Energy returned to line(kWh)

// Results
disp("PART IV - EXAMPLE : 8.4 : SOLUTION :-")
printf("\nEnergy returned to the line = %.1f kWh", energy)
```

Scilab code Exa 46.5 Braking effect and Rate of retardation produced by this braki

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 8: BRAKING

// EXAMPLE : 8.5 :
// Page number 811-812
clear ; clc ; close ; // Clear the work space and console
funcprot(0)

// Given data
area = 16.13        // Area of brakes(sq.cm/pole face)
phi = 2.5*10**-3    // Flux(Wb)
u = 0.2             // Co-efficient of friction
W = 10.0            // Weight of car(tonnes)

```

```
// Calculations
a = area*10**-4                         // Area of brakes(sq.m/pole face)
F = phi**2/(2*%pi*10**-7*a)             // Force(N)
force = F*u                             // Braking effect considering flux and coefficient of friction(N)
beta = u*F/(W*1000)*100                 // Rate of retardation produced by braking effect(cm/sec^2)

// Results
disp("PART IV - EXAMPLE : 8.5 : SOLUTION :-")
printf("\nBraking effect , F = %.f N", force)
printf("\nRate of retardation produced by this braking effect ,     = %.2f cm/sec^2", beta)
```

Chapter 47

ELECTRIC TRACTION SYSTEMS AND POWER SUPPLY

Scilab code Exa 47.1 Maximum potential difference between any two points of the ra

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 9: ELECTRIC TRACTION SYSTEMS AND POWER
   SUPPLY

// EXAMPLE : 9.1 :
// Page number 817-818
clear ; clc ; close ; // Clear the work space and
   console

// Given data
L = 3.0             // Length of section ACB of rail(
   km)
```

```
L_B_A = 2.0         // Distance of B from A(km)
I_load = 350.0      // Loading(A/km)
r_rail = 0.035      // Resistance of rail(ohm/km)
r_feed = 0.03       // Resistance of negative feeder(
    ohm/km)

// Calculations
x_val = integrate('I_load*(L-x)','x',0,L_B_A)
I = x_val/(L_B_A-0)                                    //
    Current in negative feeder(A)
x = L-(I/I_load)                                       //
    Distance from feeding point(km)
C = integrate('r_rail*I_load*x','x',0,x)
V = r_feed*L_B_A*I                                     //
    Voltage produced by negative booster(V)
rating = V*I/1000                                      //
    Rating of the booster(kW)

// Results
disp("PART IV - EXAMPLE : 9.1 : SOLUTION :-")
printf("\nMaximum potential difference between any
    two points of the rails, C = %.2f V", C)
printf("\nRating of the booster = %.1f kW", rating)
```

Scilab code Exa 47.2 Maximum sag and Length of wire required

```
// A Texbook on POWER SYSTEM ENGINEERING
// A.Chakrabarti, M.L.Soni, P.V.Gupta, U.S.Bhatnagar
// DHANPAT RAI & Co.
// SECOND EDITION

// PART IV : UTILIZATION AND TRACTION
// CHAPTER 9: ELECTRIC TRACTION SYSTEMS AND POWER
    SUPPLY

```

```
// EXAMPLE : 9.2 :
// Page number 820
clear ; clc ; close ; // Clear the work space and console

// Given data
D = 50.0     // Distance between poles(m)
w = 0.5      // Weight of trolley wire per metre(kg)
T = 520.0    // Maximum tension(kg)

// Calculations
l = D/2                                  // Half distance b/w poles(m)
d = w*l**2/(2*T)                         // Sag(m)
wire_length = 2*(l+(2*d**2/(3*l)))       // Length of wire required(m)

// Results
disp("PART IV - EXAMPLE : 9.2 : SOLUTION :-")
printf("\nMaximum sag, d = %.4f metres", d)
printf("\nLength of wire required = %.f metres", wire_length)
```