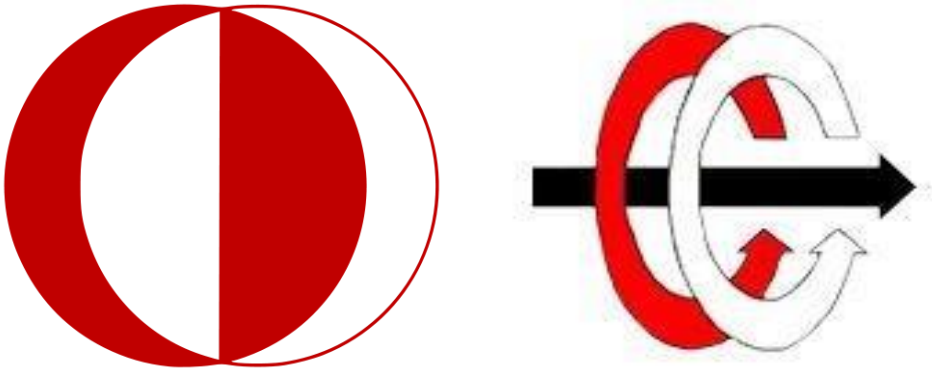
MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING



EE374

ELECTRICAL EQUIPMENT AND APPLICATIONS

Spring 2016 Term Project

Folawiyo Aminu – 2001295

TABLE OF CONTENTS

I. INTRODUCTION

Compensation, the process of reducing the amount of reactive power drawn by a load compared to the active power is of utmost importance in electric power systems. The benefits of compensation includes the usage of thinner cable lines, lower line losses, increased efficiency , reduced voltage drops on the lines, increased transmission capacity, less generator and transformer heating, better voltage regulation and so on. In this project, it is asked to design a tool that can design reactive power compensation units for 3-phase systems based on the some specifications in MATLAB environment.

II. SPECIFICATIONS AND DESCRIPTIONS

The design tool is to be able to evaluate 2 cases and find compensation units at minimal costs to satisfy power factor higher than 0.98 for inductive loads and higher than 0.95 for capacitive loads. These cases are

- Case 1 – On-site compensation
- Case 2 – Substation compensation

Other specifications are, “Graphical User Interface” is to be designed.

Information to be provided are

- Network connectivity and parameters
- Load curves

III. DESIGN METHOLOGY

a. CASE 1

For the design of the on-site compensation. A couple of assumptions were made. These assumptions include

- Balanced 3 phase systems – The system is assumed to be a balanced 3 phase system since the contrary was not stated. This assumption eases calculation significantly since with this assumption, the per-phase equivalent circuit is used to design the tool.
- Constant Voltage at load – For the design, the voltage at the loads and buses are assumed to be constant that is, unvarying with the power drawn by the load. While this is not the most accurate exact model of the system, the variation of the voltage level of the load with power as well as the voltage drop on the lines is incomparable to the nominal voltage levels. Consequently, the voltage is assumed to be constant at the nominal level.

Solution procedure – The solution for the 1st case was implemented in a series of steps shown below

- Step 1: Design of compensation unit to unity pf – First, a function was designed and implemented to calculate the required KVAR to compensate load to unity pf at every hour.
- Step 2: Maximum values for inductance and capacitor compensation units are found.
- Step 3: These values are then divided into appropriate number of smaller units to satisfy the power factor requirements at all times.

b. CASE 2

For the design of the substation compensation, assumptions made during the design in the Case 1 are also made in this case including balanced 3 phase systems and constant voltage at load.

Solution procedure – The solution for the 2nd case was implemented in a series of steps which are very similar to the case 1. The steps are shown below

- Step 1: Calculation of all load and line active power and reactive power.
- Step 2: Design of compensation unit to unity pf – First, a function was designed and implemented to calculate the required KVAR to compensate all loads to unity pf at every hour.
- Step 3: Maximum values for inductance and capacitor compensation units are found.
- Step 4: These values are then divided into appropriate number of smaller units to satisfy the power factor requirements at all times.

This solution method has an added advantage of compensation to unity pf at the period of peak load.

IV. RESULTS FROM SAMPLE CASE

a. COMPENSATION UNITS

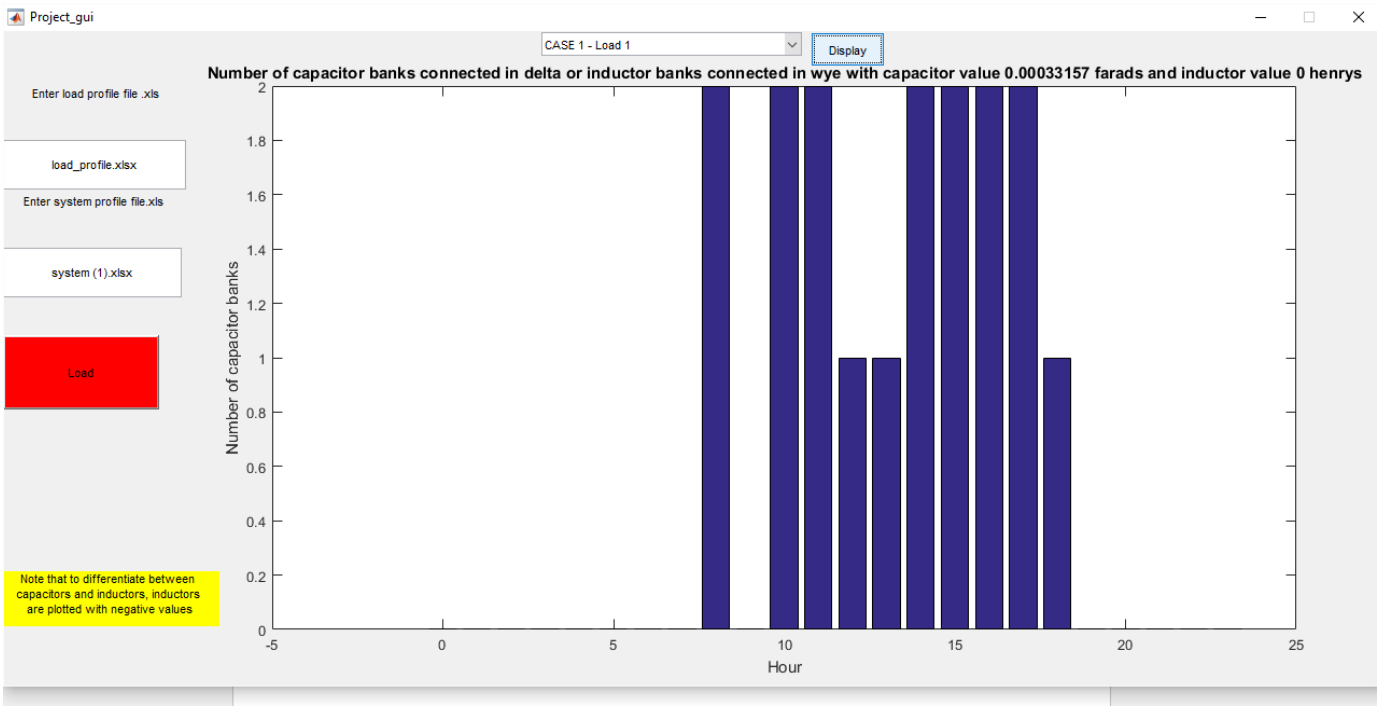


Figure 1: Compensation units, CASE 1 – LOAD 1

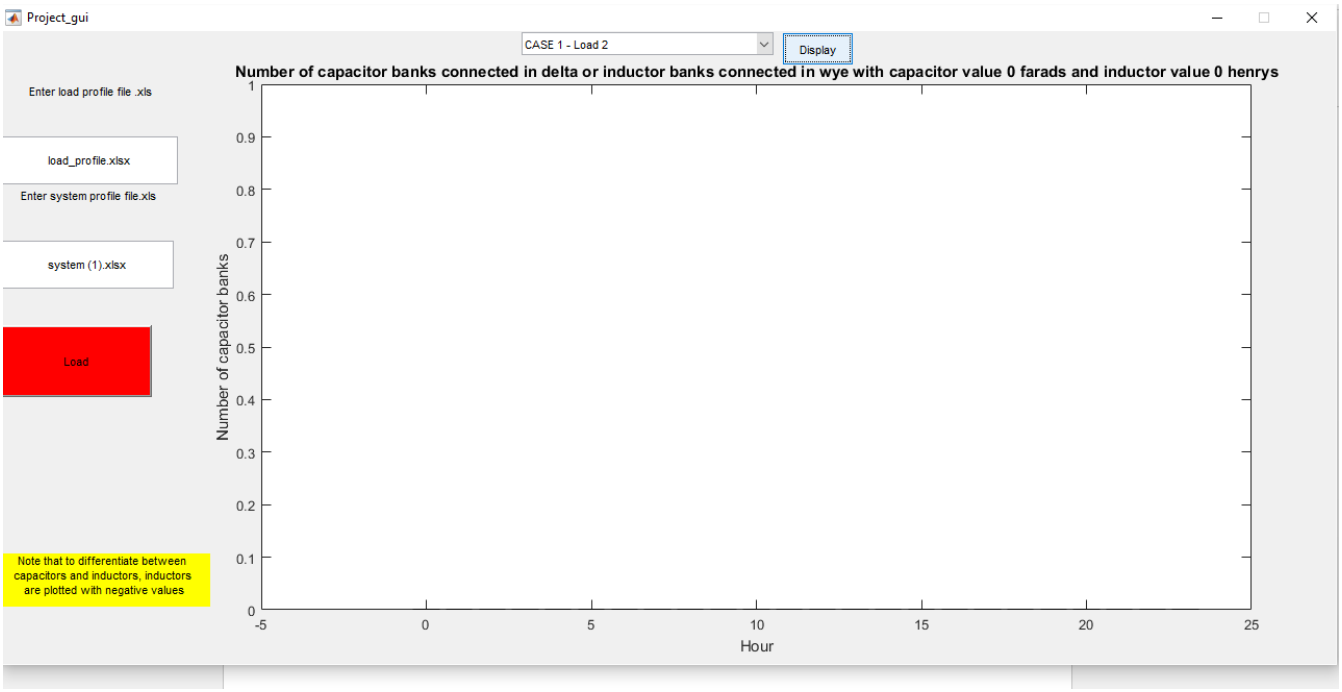


Figure 2: Compensation units, CASE 1 – LOAD 2

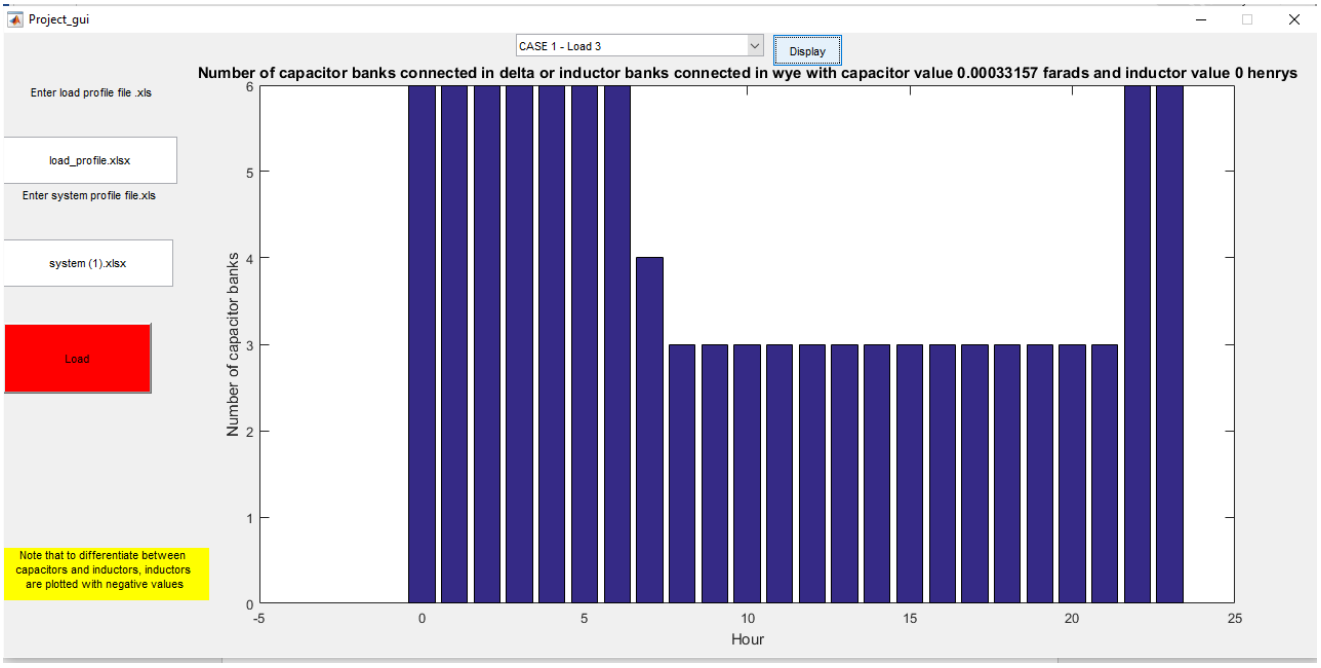


Figure 3: Compensation units, CASE 1 – LOAD 3

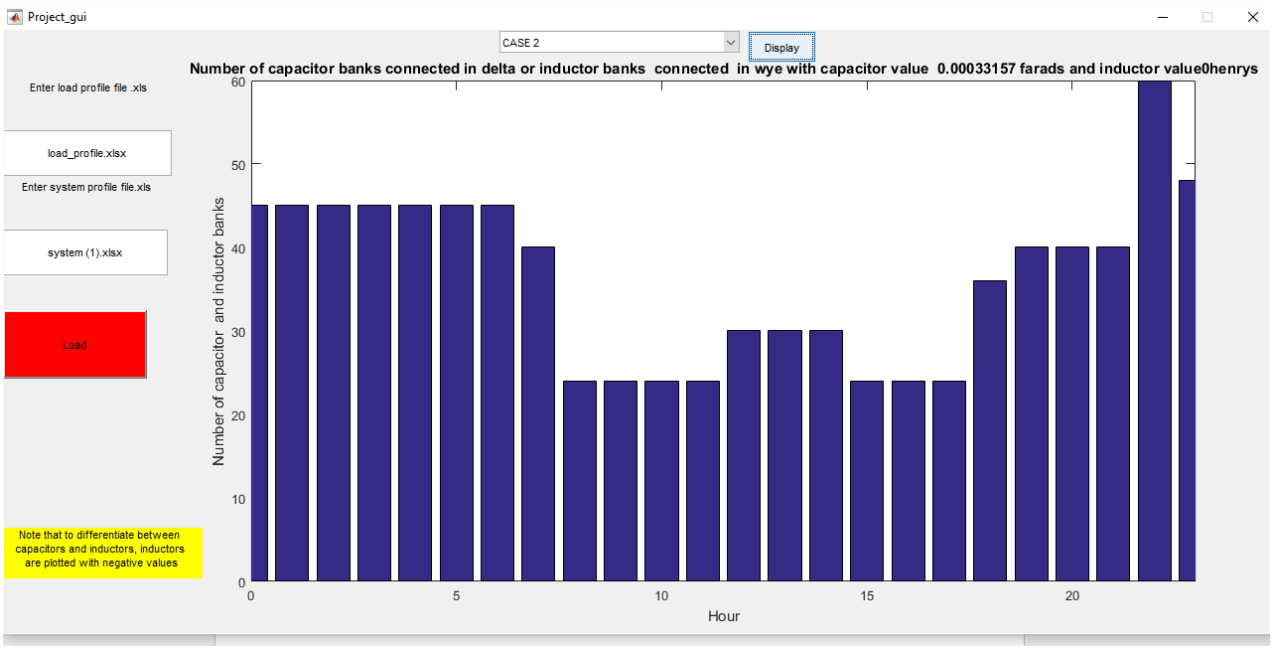


Figure 4: Compensation units, CASE 2

COMMENTS:

- The compensation units for both cases are shown above in figures 1,2,3,4. The 3-phase capacitor banks used as compensating units for this sample case are connected in delta to minimize cost as if they were connected in wye, 3 times more capacitors would be required. The reverse is the case for the inductors and thus inductor banks are connected in wye.
- This tool is built to minimize costs and the number of capacitor utilized in all cases is the minimum to satisfy the power factor requirements. For load 1, a maximum of 2 capacitors are used with switching done at the required time to activate the desired number of capacitors. The same applies with Load 3 which has 6 capacitor banks connected to it. Load 3 has a power factor of unity all the time so there is no need for compensation.
- For case 2, there is a large number of capacitors used, albeit the minimum to satisfy the power factor requirements. However this is substation power factor compensation so it's a valid design regardless as it would be expected that the space for these capacitor banks exist.

b. Line loadings

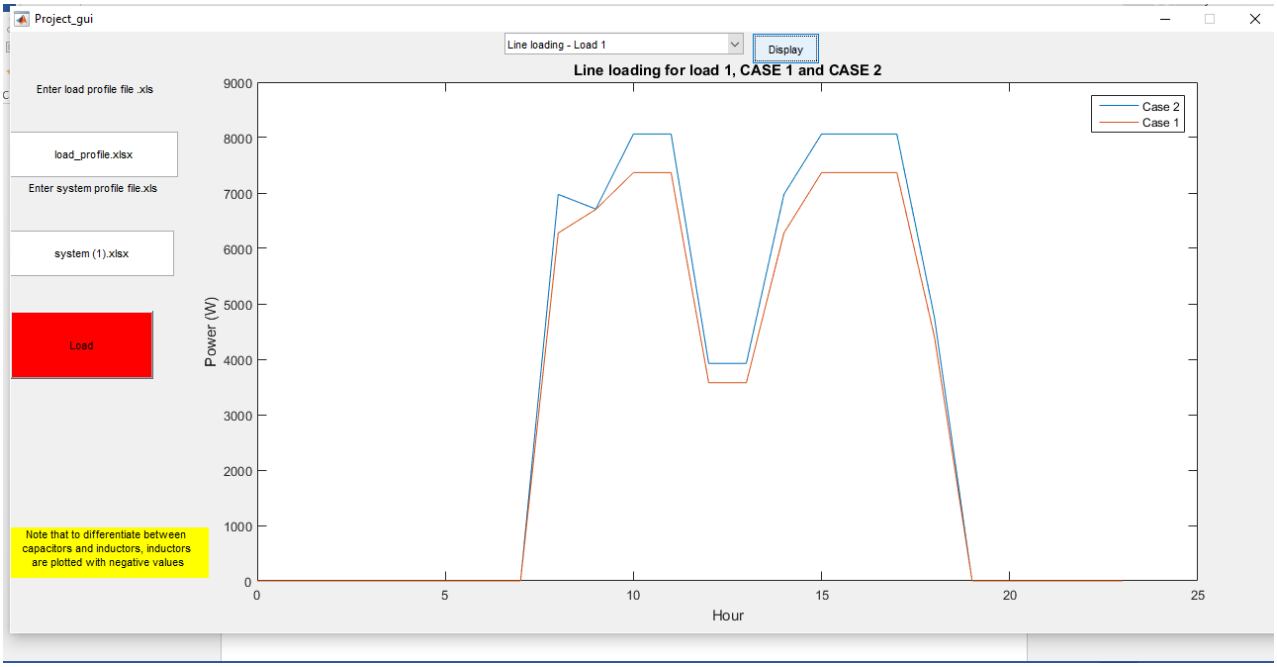


Figure 5: Line losses, LOAD 1 - CASE 1 & CASE 2

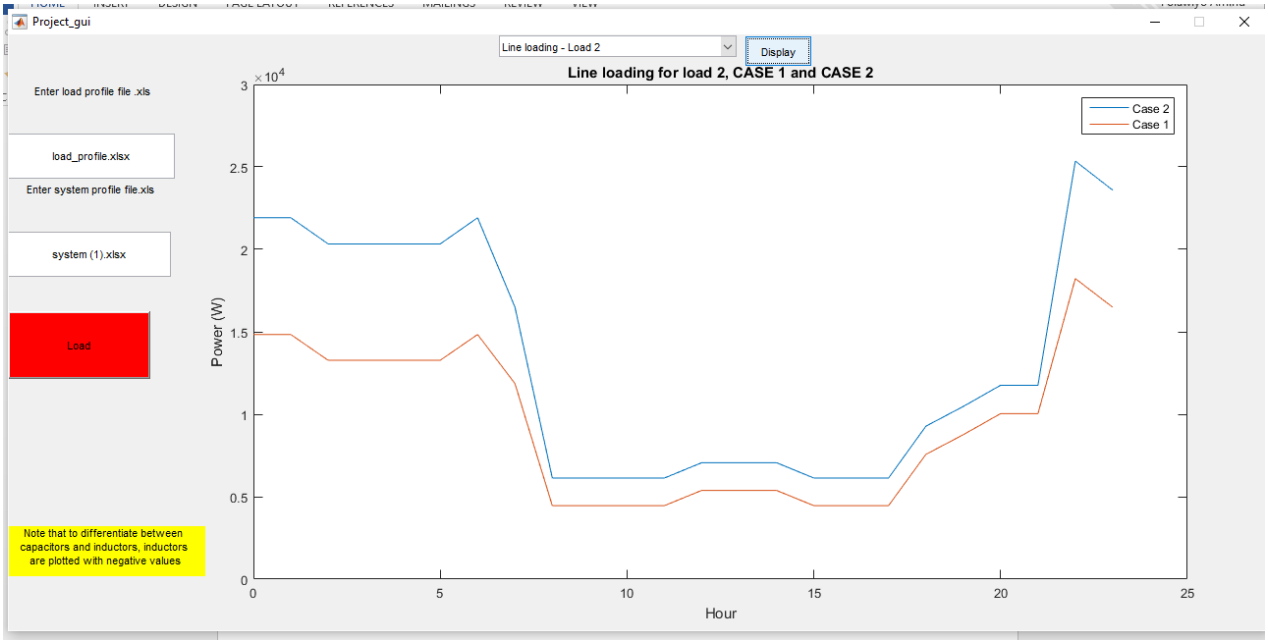


Figure 6: Line losses, LOAD 2 - CASE 1 & CASE 2

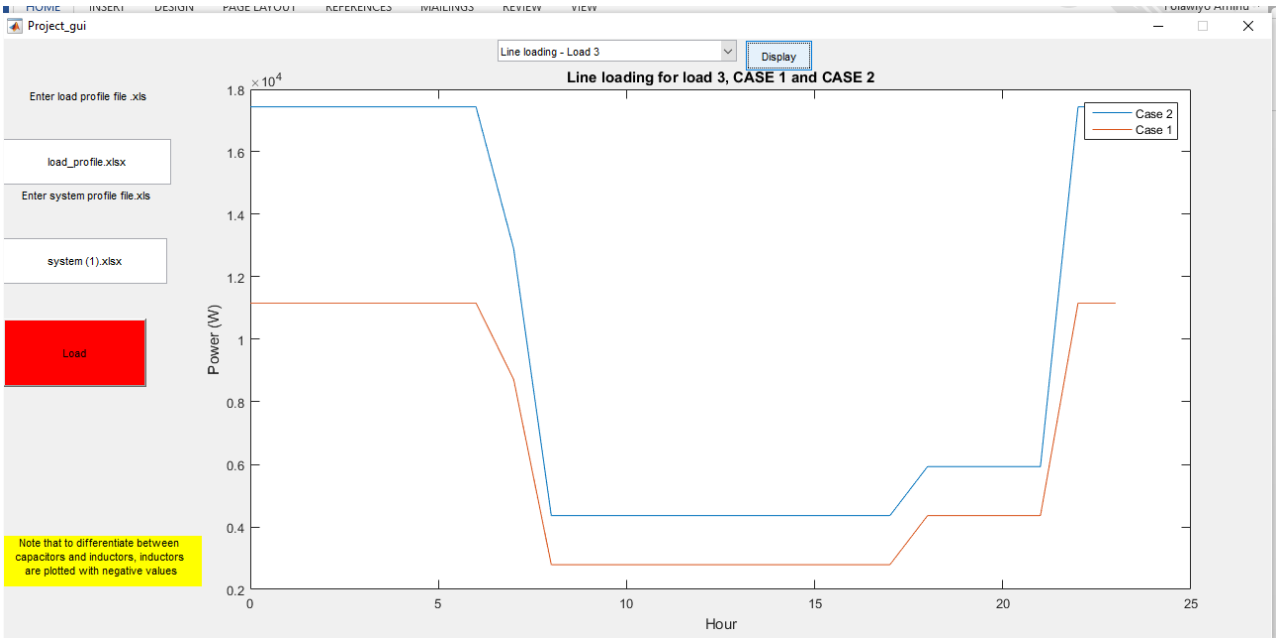


Figure 7: Line losses, LOAD 3 - CASE 1 & CASE 2

COMMENTS:

- The line loadings are as expected, relatively low losses compared to the load.
- As shown above in figures 5,6,7, the line losses are greater in case 2 than in case 1. This is expected as case 1 is on site compensation and thus the current on the lines and the power drawn by the loads in case 1 is less than in case 2 which is substation compensation. Hence the line losses are less in case 1 than in case 2. Consequently, to minimize line loss, on-site compensation is preferable.

c. Power factor as seen at the substation.

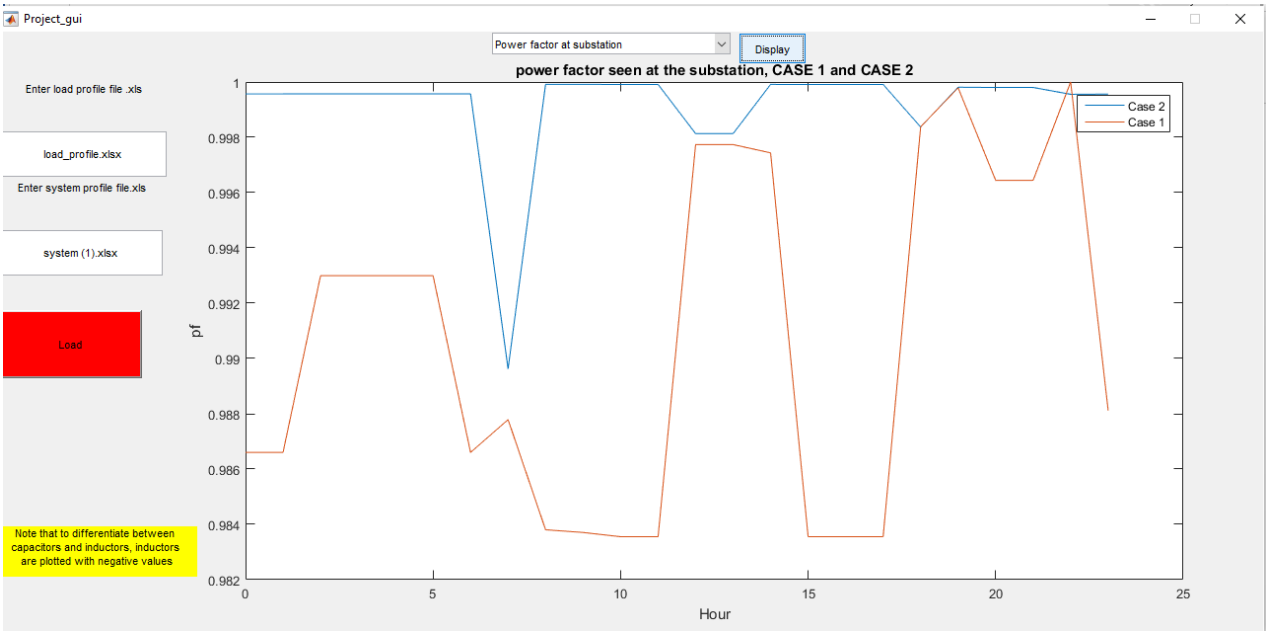


Figure 8: Power factor as seen from the transformer substation - CASE 1 & CASE 2

COMMENTS:

- In both cases, power factor is greatly improved and improved and satisfy the power factor requirements.
- However, the power factor in case 2 is generally better than in case 1 since the compensation in case 2 is substation compensation and thus directly is involved with the power factor as seen from the substation unlike that of the case 1 in which the power factor correction is done at the load and thus does not take into consideration the effect of the lines at the power factor. Consequently if power factor seen at the substation is the most important design consideration, substation compensation is preferable.

d. Voltage drop at the load

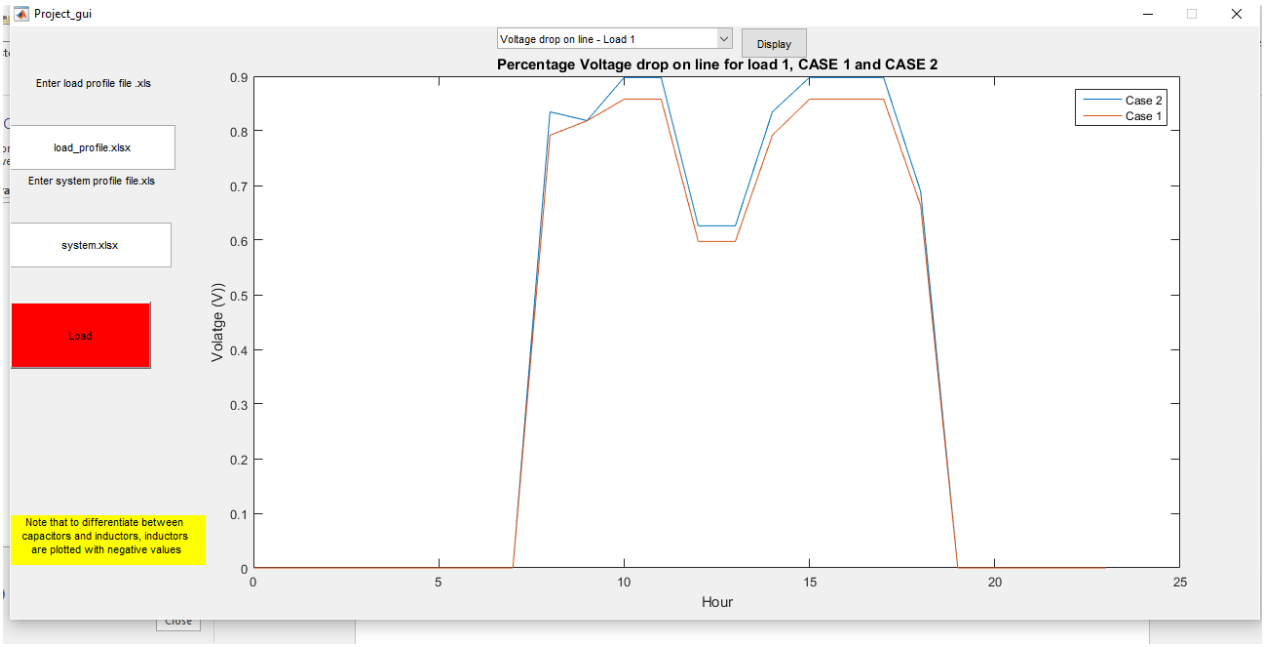


Figure 9: Percentage Voltage drop on line for load 1, CASE 1 and CASE 2

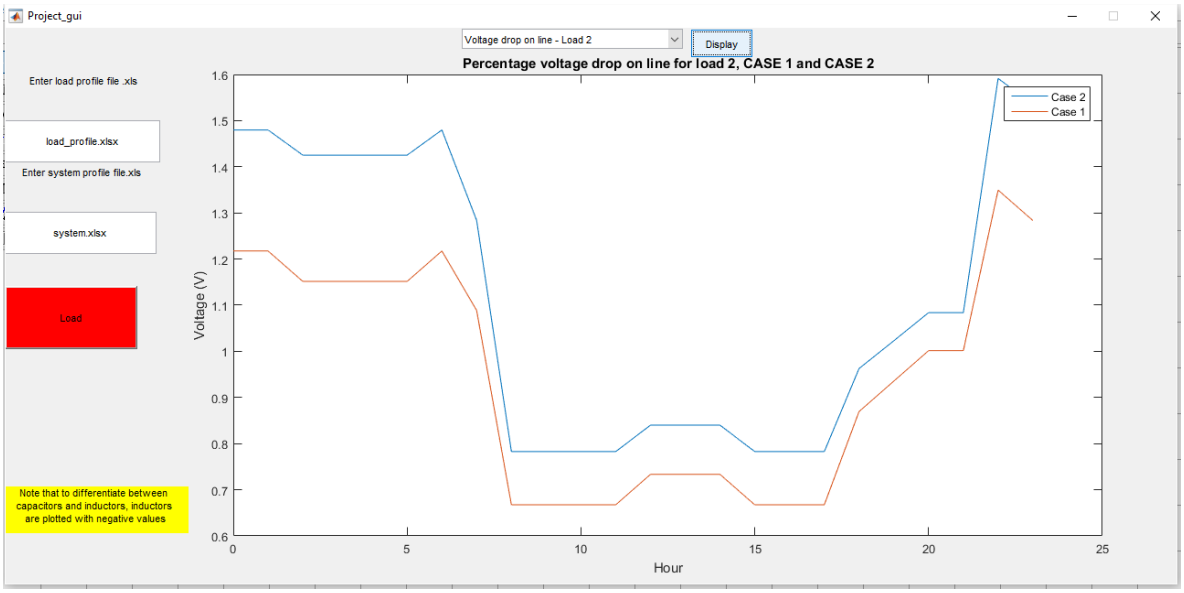


Figure 10: Percentage Voltage drop on line for load 2, CASE 1 and CASE 2

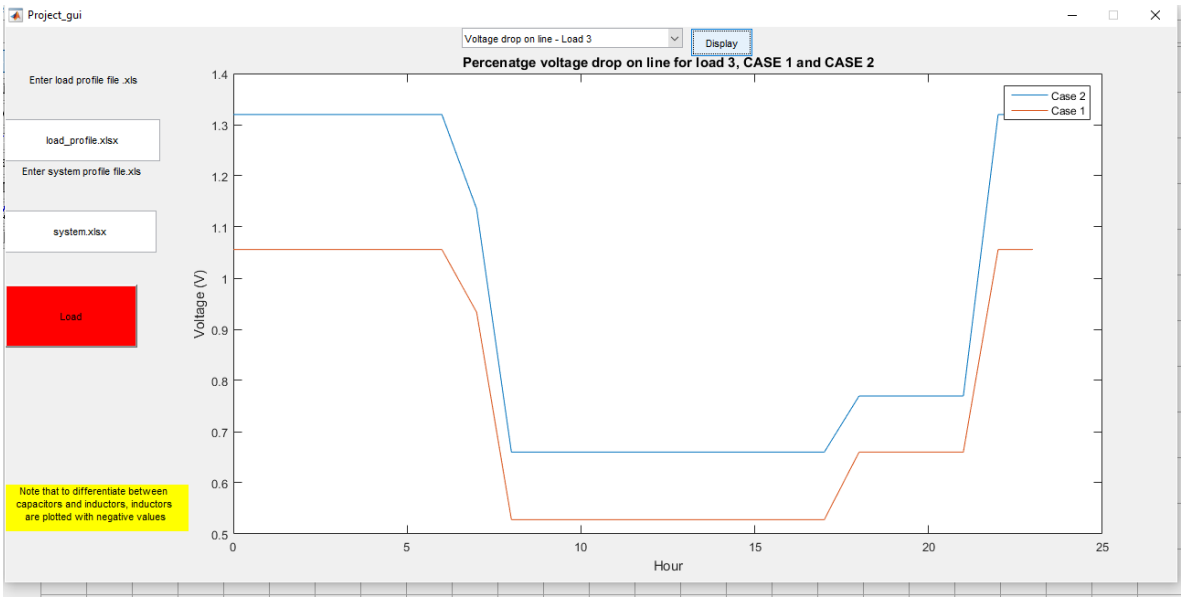


Figure 11: Percentage Voltage drop on line for load 3, CASE 1 and CASE 2

COMMENTS:

- As can be seen in figures 9, 10 & 11, the percentage voltage drop on the lines are extremely small, less than 2% on all the lines in both cases so original assumption of constant voltage at load is shown to be a valid assumption for ease of calculations.
- In this test case, there is more voltage drop on the lines in case 2 which can be directly inferred from the line loadings as more current corresponds to more voltage drop on the lines. Consequently, for better regulation, on-site compensation is preferable.

V. CONCLUSION

In this project, the design of a reactive power compensation tool has been implemented to provide both on-site compensation and substation compensation. The results of the compensation in both cases has been compared with regards to line loadings, power factor as seen from the substation and voltage drop across the load. With a sample case of inductive loads, it is shown that for less line losses and better regulation, on-site compensation is preferable while for power factor quality (closeness to unity) seen from the substation, substation compensation is the way to go. This tool implemented on Matlab, provided an opportunity to develop and practice computational programming as well an outlet to utilize knowledge gained in, EE374 - Electrical equipment and applications course. The algorithm for the case 2 was a real head scratcher and had to be done using basic loops thus the length of the code. This tool was designed to be generic for any 3 load curves or less. A protection scheme for this system is proposed in appendix A. Overall this project was enjoyable and provided a good introduction to power systems engineering.

VI. APPENDIX

a. PROTECTION SCHEME

The following protective devices can be utilized in this power system.

- Fuses – External fuses can be utilized in series with the loads, in series with the compensation units and also at the substation to interrupt faults.
- Circuit breakers – In the same vein, circuit breakers can be utilized in series with the fuses for extra/ back up protection
- Over current & over voltage relays – These can be used to protect the loads and the compensation units.
- Relays for Bank Closing Control & discharge resistor – A compensation unit should not be connected back to the system without discharge of the capacitor bank after disconnection, otherwise catastrophic damage to the circuit breaker, fuses or switch can occur. To accelerate the discharge of the bank, each individual capacitor unit should have a resistor to discharge the stored charges
- Capacitor switches or/and Power electronic devices should be used for the switching of the capacitors.

b. MATLAB CODE

```
function varargout = Project_gui(varargin)
% PROJECT_GUI MATLAB code for Project_gui.fig
%      PROJECT_GUI, by itself, creates a new PROJECT_GUI or raises the
existing
%      singleton*.
%
%      H = PROJECT_GUI returns the handle to a new PROJECT_GUI or the handle
to
%      the existing singleton*.
%
%      PROJECT_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROJECT_GUI.M with the given input
arguments.
%
%      PROJECT_GUI('Property','Value',...) creates a new PROJECT_GUI or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Project_gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Project_gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Project_gui

% Last Modified by GUIDE v2.5 19-May-2016 23:08:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Project_gui_OpeningFcn, ...
                   'gui_OutputFcn',  @Project_gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
end

% --- Executes just before Project_gui is made visible.
function Project_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Project_gui (see VARARGIN)

clc
% Choose default command line output for Project_gui
handles.output = hObject;
handles.L1= 0;
handles.C1= 0;
handles.n1= 0;
handles.n2= 0;
handles.L2= 0;
handles.C2= 0;
handles.n3= 0;
handles.n4= 0;
handles.L3= 0;
handles.C3= 0;
handles.n5= 0;
handles.n6= 0;
handles.L4= 0;
handles.C4= 0;
handles.n7= 0;
handles.n8= 0;
handles.w = 0;

handles.Pl1 = 0;
handles.PlL1 = 0;

handles.Pl2 =0;
handles.PlL2 =0;

handles.Pl3 =0;
handles.PlL3 =0;


handles.pfCSS =0;
handles.pfSS =0;

handles.VL1 = 0;
handles.VLC1 = 0;

handles.VL2 = 0;
```

```
handles.VLC2 = 0;

handles.VL3 = 0;
handles.VLC3 = 0;

handles.V1 = 0;
handles.V2 = 0;
handles.V3 = 0;




% Update handles structure
guidata(hObject, handles);




% UIWAIT makes Project_gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);

end
% --- Outputs from this function are returned to the command line.
function varargout = Project_gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
end
function load_profile_Callback(hObject, eventdata, handles)
% hObject    handle to load_profile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of load_profile as text
%        str2double(get(hObject,'String')) returns contents of load_profile
as a double
end

% --- Executes during object creation, after setting all properties.
function load_profile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to load_profile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

end

function system_profile_Callback(hObject, eventdata, handles)
% hObject    handle to system_profile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of system_profile as text
%        str2double(get(hObject,'String')) returns contents of system_profile
as a double

end
% --- Executes during object creation, after setting all properties.
function system_profile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to system_profile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in Load_files.
function Load_files_Callback(hObject, eventdata, handles)
% hObject    handle to Load_files (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

LP = get (handles.load_profile, 'String');
S = get (handles.system_profile, 'String');

% handles.L1 = zeros (3,24);
%
% L2 = zeros (3,24);
% L3 = zeros (3,24);


handles.L1 = xlsread (LP ,1, 'A2:C25'); %Load profile 1

guidata(hObject,handles);

handles.L2 = xlsread (LP ,2, 'A2:C25'); %Load profile 2
guidata(hObject,handles);
L2 = handles.L2
handles.L3 = xlsread (LP ,3, 'A2:C25'); %Load profile 3
```

```
L3 = handles.L3;
Bs1 = xlsread (S ,1, 'A2'); %Sending bus for load 1
Bs2 = xlsread (S ,1, 'A3'); %Sending bus for load 2
Bs3 = xlsread (S ,1, 'A4'); %Sending bus for load 3

Br1 = xlsread (S ,1, 'B2'); %Receiving bus for load 1
Br2 = xlsread (S ,1, 'B3'); %Receiving bus for load 2
Br3 = xlsread (S ,1, 'B4'); %Receiving bus for load 3

Np1 = xlsread (S ,1, 'd2'); %Number of parallel lines for load 1
Np2 = xlsread (S ,1, 'd3'); %Number of parallel lines for load 2
Np3 = xlsread (S ,1, 'd4'); %Number of parallel lines for load 3

d1 = xlsread (S ,1, 'e2')/1000; %distance between buses for load 1 in km
d2 = xlsread (S ,1, 'e3')/1000; %distance between buses for load 2 in km
d3 = xlsread (S ,1, 'e4')/1000; %distance between buses for load 3 in km

%NIa is not important
[NIa,CT1] = xlsread (S ,1, 'c2'); %Cable type for load 1
[NIa,CT2] = xlsread (S ,1, 'c3'); %Cable type for load 1
[NIa,CT3] = xlsread (S ,1, 'c4'); %Cable type for load 1

[NIa, CTS1] = xlsread (S ,2, 'a2'); %Cable specifics on system page 2
[NIa, CTS2] = xlsread (S ,2, 'a3');
[NIa, CTS3] = xlsread (S ,2, 'a4');

CTi1 = cast (CT1,'char');
CTi2 = cast (CT2,'char');
CTi3 = cast (CT3,'char');

        handles.L1(:,2) = 1e3*handles.L1(:,2);
        guidata(hObject,handles);
        handles.L1(:,3) = 1e3*handles.L1(:,3);
        guidata(hObject,handles);
        L1 = handles.L1;


        L2(:,2) = 1e3*L2(:,2);
        L2(:,3) = 1e3*L2(:,3);
        L3(:,2) = 1e3*L3(:,2);
        L3(:,3) = 1e3*L3(:,3);

        Pl1 = zeros(24,1);
        Ql1 = zeros(24,1);
        Pl2 = zeros(24,1);
        Ql2 = zeros(24,1);
        Pl3 = zeros(24,1);
        Ql3 = zeros(24,1);

        PLL1 = zeros(24,1);
        QLL1 = zeros(24,1);
        PLL2 = zeros(24,1);
        QLL2 = zeros(24,1);
        PLL3 = zeros(24,1);
        QLL3 = zeros(24,1);
```

```
        PB = zeros(24,4);
        QB = zeros(24,4);
        QCB = zeros(24,4);

        B = [ Bs1,Br1;Bs2,Br2;Bs3,Br3];
        PL =  [handles.L1(:,2),L2(:,2),L3(:,2)];
        QL =  [handles.L1(:,3),L2(:,3),L3(:,3)];
```

```
for r = 1:5
A1(r) = str2double(CTi1(r+2));
if isnan(A1(r))
 A1(r) = [];
 break
end
end
A1 = sscanf( sprintf( '%u', A1 ), '%1d' )';
for r = 1:length(CTi2)-2
A2(r) = str2double(CTi2(r+2));
if isnan(A2(r))
 A2(r) = [];
 break
end
end
A2 = sscanf( sprintf( '%u', A2 ), '%1d' )';
for r = 1:length(CTi3)-2
A3(r) = str2double(CTi3(r+2));
if isnan(A3(r))
 A3(r) = [];
 break
end
end
a3 = 0;
a2 = 0;
a1= 0;
for k1 = length(A3):-1:1
    a3 = a3 + A3(k1)*10.^(length(A3)-k1);
end
for k2 = length(A2):-1:1
    a2 = a2 + A2(k2)*10.^(length(A2)-k2);
end
for k3 = length(A1):-1:1
    a1 = a1 + A1(k3)*10.^(length(A1)-k3);
end
```

```
%X1
```

```
if ( strcmp(CT1,CTS1 ))
    [~,X1] = xlsread(S ,2, 'b2');
    X1  = str2double(X1)*d1
    %inductance for line feeding load 1, distance * Ind reactance
    V1 = xlsread (S ,2, 'c2'); %nominal voltage
   [NIa,m1] = xlsread (S ,2, 'd2'); %Cable material
   if ( strcmp(m1,'copper'))
        rho1 = 56;
   elseif ( strcmp(m1,'aluminium'))
      rho1 = 35;
   end
elseif ( strcmp(CT1,CTS2 ))
    [~,X1] = xlsread (S ,2, 'b3');
    X1  = str2double(X1)*d1
    V1 = xlsread (S ,2, 'c3');
   [NIa,m1] = xlsread (S ,2, 'd3');
   if ( strcmp(m1,'copper'))
      rho1 = 56;
   elseif ( strcmp(m1,'aluminium'))
      rho1 = 35;  %
   end

elseif ( strcmp(CT1,CTS3 ))
    [~,X1] = xlsread (S ,2, 'b4');
    X1  = str2double(X1)*d1
    V1 = xlsread (S ,2, 'c4');
   [NIa,m1] = xlsread (S ,2, 'd4');
   if ( strcmp(m1,'copper'))
      rho1 = 56;
   elseif ( strcmp(m1,'aluminium'))
         rho1 = 35;
   end
   end
   R1 = 1e3*d1/(rho1*a1) ;

%X2
if ( strcmp(CT2,CTS1 ))
    [~,X2] =  xlsread (S ,2, 'b2');
     X2  = str2double(X2)*d2;
    %inductance for load 1, distance * Ind reactance
    V2 = xlsread (S ,2, 'c2'); %nominal voltage
   [NIa,m2] = xlsread (S ,2, 'd2'); %Cable material
   if ( strcmp(m2,'copper'))
     rho2 = 56;
   elseif ( strcmp(m2,'aluminium'))
      rho2 = 35;
   end
elseif ( strcmp(CT2,CTS2 ))
     [~,X2] =  xlsread (S ,2, 'b3');
     X2  = str2double(X2)*d2;
    V2 = xlsread (S ,2, 'c3');
   [NIa,m2] = xlsread (S ,2, 'd3');
   if ( strcmp(m2,'copper'))
     rho2 = 56;
   elseif ( strcmp(m2,'aluminium'))
     rho2 = 35;
```

```
   end
elseif ( strcmp(CT2,CTS3 ))
   [~,X2] =  xlsread (S ,2, 'b4');
     X2  = str2double(X2)*d2;
    V2 = xlsread (S ,2, 'c4');
   [NIa,m2] = xlsread (S ,2, 'd4');
   if ( strcmp(m2,'copper'))
     rho2 = 56;
   elseif ( strcmp(m2,'aluminium'))
       rho2 = 35;
   end
end
R2 = 1e3*d2/(rho2*a2);

%X3
if ( strcmp(CT3,CTS1 ))
    [~,X3] = xlsread (S ,2, 'b2');
    X3  = str2double(X3)*d3;
    %inductance for load 1, distance * Ind reactance
    V3 = xlsread (S ,2, 'c2'); %nominal voltage
   [NIa,m3] = xlsread (S ,2, 'd2'); %Cable material
   if ( strcmp(m3,'copper'))
            rho3 = 56;
   elseif ( strcmp(m3,'aluminium'))
       rho3 = 35;
   end
elseif ( strcmp(CT3,CTS2 ))
     [~,X3] = xlsread (S ,2, 'b3');
    X3  = str2double(X3)*d3;
    V3 = xlsread (S ,2, 'c3');
   [NIa,m3] = xlsread (S ,2, 'd3');
   if ( strcmp(m3,'copper'))
    rho3 = 56;
   elseif ( strcmp(m3,'aluminium'))
       rho3 = 35;
   end
elseif ( strcmp(CT3,CTS3 ))
    [~,X3] = xlsread (S ,2, 'b4');
    X3  = str2double(X3)*d3;
    V3 = xlsread (S ,2, 'c4');
   [NIa,m3] = xlsread (S ,2, 'd4');
   if ( strcmp(m3,'copper'))
      rho3 = 56;
   elseif ( strcmp(m3,'aluminium'))
     rho3 = 35;
   end
end

R3 = 1e3*d3/(rho3*a3);


R1 = R1/Np1; %effective resistance considering parallel lines
R2 = R2/Np2;
R3 = R3/Np3;
```

```
X1 = X1/Np1 %effective reactance considering parallel lines
X2 = X2/Np2
X3 = X3/Np3
```

```
handles.w = 100*pi;
guidata(hObject,handles);
w = handles.w;
```

```
        handles.V1 = V1;
        guidata(hObject,handles);
        handles.V2 = V2;
        guidata(hObject,handles);

        handles.V3 = V3;
        guidata(hObject,handles);
```

```
         [C1,n1,n2,Q1] =  CU_actual(L1,V1)
         handles.C1 = C1;
         guidata(hObject,handles);
          handles.n1 = n1;
          guidata(hObject,handles);
           handles.n2 = n2;
           guidata(hObject,handles);
```

```
         [C2,n3,n4,Q2] =  CU_actual(L2,V2)
          handles.C2 = C2;
         guidata(hObject,handles);
          handles.n3 = n3;
          guidata(hObject,handles);
           handles.n4 = n4;
           guidata(hObject,handles);

         [C3,n5,n6,Q3] =  CU_actual(L3,V3)
              handles.C3 = C3;
             guidata(hObject,handles);
              handles.n5 = n5;
             guidata(hObject,handles);
             handles.n6 = n6;
             guidata(hObject,handles);
```

```
        PB(:,Br1+1) =  handles.L1(:,2);
        QB(:,Br1+1) =  handles.L1(:,3);
        QCB (:,Br1+1) =  Q1;
        VB (:,Br1+1) = V1 * ones(1,24);
        VBC (:,Br1+1) = V1 * ones(1,24);

        PB(:,Br2+1) = L2(:,2);
        QCB(:,Br2+1) =  Q2;
        QB(:,Br2+1) =  L2(:,3);
        VB (:,Br2+1) = V2 * ones(1,24);
        VBC (:,Br2+1) = V2 * ones(1,24);

        PB(:,Br3+1) =  L3(:,2);
        QB(:,Br3+1) =  L3(:,3);
        QCB(:,Br3+1) =  Q3;
        VB (:,Br3+1) = V3 * ones(1,24);
        VBC (:,Br3+1) = V3 * ones(1,24);
        PCB = PB;

        for i = 0:3



        %Is load 1 (d2) fed from an exclusively receiving bus?

        if (Br1==1 && Bs2~=1 && Bs3~=1 && Bs1~=1)
            [VL1, Pl1,Ql1,Il1] = S_line(PB(:,Br1+1),QB(:,Br1+1),X1,R1,V1) ;
%line losses in case 2 without compensation
            [VLC1, PlL1,QlL1,IlL1] =
S_line(PCB(:,Br1+1),QCB(:,Br1+1),X1,R1,V1) ; %line losses in case 1 with
compensation

            PLL1 = Pl1 + PB(:,Br1+1); % Power on Line plus load in case 2
without compensation
            QLL1 = Ql1 + QB(:,Br1+1); % Reactive power on Line plus load in
case 2 without compensation
            PB(:,Bs1+1) = PB(:,Bs1+1)+ PlL1;
            QB(:,Bs1+1) = QB(:,Bs1+1)+ QlL1;


            PCLL1 = PlL1 + PCB(:,Br1+1); % Power on Line plus load in case 2
with compensation
            QCLL1 = QlL1 + QCB(:,Br1+1); % Reactive power on Line plus load
in case 2 with compensation
            PCB(:,Bs1+1) = PCB(:,Bs1+1)+ PCLL1;
            QCB(:,Bs1+1) = QCB(:,Bs1+1)+ QCLL1;

            Bs1 = Br1;
        elseif (Br1==2 && Bs2~=2 && Bs3~=2 && Bs1~=2)
            [VL1,Pl1,Ql1,Il1] = S_line(PB(:,Br1+1),QB(:,Br1+1),X1,R1,V1);
            [VLC1,PlL1,QlL1,IlL1] =
S_line(PCB(:,Br1+1),QCB(:,Br1+1),X1,R1,V1);

            PLL1 = Pl1 + PB(:,Br1+1);
```

```
            QLL1 = Ql1 + QB(:,Br1+1);
            PB(:,Bs1+1) = PB(:,Bs1+1)+ PlL1;
            QB(:,Bs1+1) = QB(:,Bs1+1)+ QlL1;

             PCLL1 = PlL1 + PCB(:,Br1+1); % Power on Line plus load in case 2
with compensation
            QCLL1 = QlL1 + QCB(:,Br1+1); % Reactive power on Line plus load
in case 2 with compensation
            PCB(:,Bs1+1) = PCB(:,Bs1+1)+ PCLL1;
            QCB(:,Bs1+1) = QCB(:,Bs1+1)+ QCLL1;

            Bs1=Br1;
        elseif (Br1==3 && Bs2~=3 && Bs3~=3 && Bs1~=3)
              [VL1,Pl1,Ql1,Il1] = S_line(PB(:,Br1+1),QB(:,Br1+1),X1,R1,V1);
            [VLC1,PlL1,QlL1,IlL1] =
S_line(PCB(:,Br1+1),QCB(:,Br1+1),X1,R1,V1);

            PLL1 = Pl1 + PB(:,Br1+1);
            QLL1 = Ql1 + QB(:,Br1+1);
            PB(:,Bs1+1) = PB(:,Bs1+1)+ PlL1;
            QB(:,Bs1+1) = QB(:,Bs1+1)+ QlL1;

             PCLL1 = PlL1 + PCB(:,Br1+1); % Power on Line plus load in case 2
with compensation
            QCLL1 = QlL1 + QCB(:,Br1+1); % Reactive power on Line plus load
in case 2 with compensation
            PCB(:,Bs1+1) = PCB(:,Bs1+1)+ PCLL1;
            QCB(:,Bs1+1) = QCB(:,Bs1+1)+ QCLL1;

            Bs1=Br1;
        end

%

            % else is load 2 (Br2) fed from an exclusive receiving bus?
        if (Br2==1 && Bs1~=1 && Bs3~=1 && Bs2~=1)
            [VL2,Pl2,Ql2,Il2] = S_line(PB(:,Br2+1),QB(:,Br2+1),X2,R2,V2) ;
            [VLC2,PlL2,QlL2,IlL2] =
S_line(PCB(:,Br2+1),QCB(:,Br2+1),X2,R2,V2) ;

            PLL2 = Pl2 + PB(:,Br2+1);
            QLL2 = Ql2 + PB(:,Br2+1);
            PB(:,Bs2+1) = PB(:,Bs2+1)+ PLL2;
            QB(:,Bs2+1) = QB(:,Bs2+1)+ QLL2;

             PCLL2 = PlL2 + PCB(:,Br2+1);
            QCLL2 = QlL2 + QCB(:,Br2+1);
            PCB(:,Bs2+1) = PCB(:,Bs2+1)+ PCLL2;
            QCB(:,Bs2+1) = QCB(:,Bs2+1)+ QCLL2;


            Bs2 = Br2;

        elseif (Br2==2 && Bs1~=2 && Bs3~=2 && Bs2~=2)
```

```
            [VL2,Pl2,Ql2,Il2] = S_line(PB(:,Br2+1),QB(:,Br2+1),X2,R2,V2) ;
            [VLC2,PlL2,QlL2,IlL2] =
S_line(PCB(:,Br2+1),QCB(:,Br2+1),X2,R2,V2) ;
            QLL2 = Ql2 + QB(:,Br2+1);
            PB(:,Bs2+1) = PB(:,Bs2+1)+ PLL2;
            QB(:,Bs2+1) = QB(:,Bs2+1)+ QLL2;

            PCLL2 = PlL2 + PCB(:,Br2+1);
            QCLL2 = QlL2 + QCB(:,Br2+1);
            PCB(:,Bs2+1) = PCB(:,Bs2+1)+ PCLL2;
            QCB(:,Bs2+1) = QCB(:,Bs2+1)+ QCLL2;

            Bs2 = Br2;

        elseif (Br2==3 && Bs1~=3 && Bs3~=3 && Bs2~=3)
          [VL2,Pl2,Ql2,Il2] = S_line(PB(:,Br2+1),QB(:,Br2+1),X2,R2,V2) ;
          [VLC2,PlL2,QlL2,IlL2] = S_line(PCB(:,Br2+1),QCB(:,Br2+1),X2,R2,V2)
;

            PLL2 = Pl2 + PB(:,Br2+1);
            QLL2 = Ql2 + QB(:,Br2+1);
            PB(:,Bs2+1) = PB(:,Bs2+1)+ PLL2;
            QB(:,Bs2+1) = QB(:,Bs2+1)+ QLL2;

            PCLL2 = PlL2 + PCB(:,Br2+1);
            QCLL2 = QlL2 + QCB(:,Br2+1);
            PCB(:,Bs2+1) = PCB(:,Bs2+1)+ PCLL2;
            QCB(:,Bs2+1) = QCB(:,Bs2+1)+ QCLL2;

            Bs2 = Br2;
        end

            % else is load 3 (Br3) fed from an exclusive receiving bus?

        if  (Br3==3 && Bs1~=3 && Bs2~=3 && Bs3~=3)
            [VL3, Pl3,Ql3,Il3] = S_line(PB(:,Br3+1),QB(:,Br3+1),X2,R2,V2) ;
             [VLC3,PlL3,QlL3,IlL3] =
S_line(PCB(:,Br3+1),QCB(:,Br3+1),X2,R2,V2) ;

            PLL3 = Pl3 + PB(:,Br3+1);
            QLL3 = Ql3 + QB(:,Br3+1);
            PB(:,Bs3+1) = PB(:,Bs3+1)+ PLL3;
            QB(:,Bs3+1) = QB(:,Bs3+1)+ QLL3;

            PCLL3 = PlL3 + PCB(:,Br3+1);
            QCLL3 = QlL3 + QCB(:,Br3+1);
            PCB(:,Bs3+1) = PCB(:,Bs3+1)+ PCLL3;
            QCB(:,Bs3+1) = QCB(:,Bs3+1)+ QCLL3;


            Bs3 = Br3;
            elseif  (Br3==2 && Bs1~=2 && Bs2~=2 && Bs3~=2)
            [VL3, Pl3,Ql3,Il3] = S_line(PB(:,Br3+1),QB(:,Br3+1),X2,R2,V2) ;
```

```
                        [VLC3,PlL3,QlL3,IlL3] =
            S_line(PCB(:,Br3+1),QCB(:,Br3+1),X2,R2,V2) ;


            PLL3 = Pl3 + PB(:,Br3+1);
            QLL3 = Ql3 + QB(:,Br3+1);
            PB(:,Bs3+1) = PB(:,Bs3+1)+ PLL3;
            QB(:,Bs3+1) = QB(:,Bs3+1)+ QLL3;

            PCLL3 = PlL3 + PCB(:,Br3+1);
            QCLL3 = QlL3 + QCB(:,Br3+1);
            PCB(:,Bs3+1) = PCB(:,Bs3+1)+ PCLL3;
            QCB(:,Bs3+1) = QCB(:,Bs3+1)+ QCLL3;

            Bs3 = Br3;
            elseif  (Br3==1 && Bs1~=1 && Bs2~=1 && Bs3~=1)
             [VL3, Pl3,Ql3,Il3] = S_line(PB(:,Br3+1),QB(:,Br3+1),X2,R2,V2) ;
             [VLC3,PlL3,QlL3,IlL3] =
S_line(PCB(:,Br3+1),QCB(:,Br3+1),X2,R2,V2) ;

            PLL3 = Pl3 + PB(:,Br3+1);
            QLL3 = Ql3 + QB(:,Br3+1);
            PB(:,Bs3+1) = PB(:,Bs3+1)+ PLL3;
            QB(:,Bs3+1) = QB(:,Bs3+1)+ QLL3;

            PCLL3 = PlL3 + PCB(:,Br3+1);
            QCLL3 = QlL3 + QCB(:,Br3+1);
            PCB(:,Bs3+1) = PCB(:,Bs3+1)+ PCLL3;
            QCB(:,Bs3+1) = QCB(:,Bs3+1)+ QCLL3;

            Bs3 = Br3;
          end
          end

          handles.L4 = [handles.L1(:,1), PB(:,1), QB];
        handles.L5 = [handles.L1(:,1), PCB(:,1), QCB(:,1)];

        L4 =  handles.L4;
        L5 = handles.L5;

            [C4,n7,n8,Q4] =  CU_actual(L4,V3);
            handles.C4 = C4;
             guidata(hObject,handles);
             handles.n7 = n7;
             guidata(hObject,handles);
             handles.n8 = n8;
              guidata(hObject,handles);

             L4 = [handles.L1(:,1), PB(:,1), Q4'];

             handles.PlL1 = PlL1;
             guidata(hObject,handles);
            handles.Pl1 = Pl1;
            guidata(hObject,handles);
```

```
          handles.PlL2 = PlL2;
           guidata(hObject,handles);
          handles.Pl2 = Pl2;
          guidata(hObject,handles);

          handles.PlL3 = PlL3;
           guidata(hObject,handles);
          handles.Pl3 = Pl3;
          guidata(hObject,handles);

           handles.VL1 = VL1;
           guidata(hObject,handles);
          handles.VLC1 = VLC1;
          guidata(hObject,handles);

          handles.VL2 = VL2;
           guidata(hObject,handles);
          handles.VLC2 = VLC2;
          guidata(hObject,handles);

          handles.VL3 = VL3;
           guidata(hObject,handles);
          handles.VLC3 = VLC3;
          guidata(hObject,handles);




        pfCSS = find_pf(L5);
       pfSS =  find_pf(L4);
        handles.pfCSS = pfCSS;
           guidata(hObject,handles);
          handles.pfSS = pfSS;
          guidata(hObject,handles);

function VL = find_voltage_drop (P,Q,I)
        VL = sqrt(P.^2 + Q.^2)./I;
end
  function a = find_pf(L)
        a(:,1) = L(:,1);
        b = sqrt( L(:,2).^2 + L(:,3).^2);
        a(:,2) = L(:,2)./b;
  end
  function [CL,Q_new] = CU_ideal(L,V) % compensation to unity pf at load
point and desired reactive power
      w = handles.w;
        Q_new = -1*L(:,3);
        for p = 1: numel(Q_new)
            if (Q_new(p) < 0)
                CL(p) = abs(Q_new(p))./ (3*V.^2*w); % Capacitors connected in
delta configuration

            elseif (Q_new(p) > 0 )
```

```
                CL(p) = -1*(V.^2)./(Q_new(p).*w); % Inductors connected in
wye configuration

            else
                CL(p) = 0;
            end
        end
  end
function [pf,Qa] = check_pf1(d,A,B,C,V) %power factor checker for lagging
load
        Q = -3*V.^2*w*A;

       if (Q<B)
            pf = d;
            Qa = B;
        else
            Qa = Q - B;
            pf = C./sqrt(C.^2+ Qa.^2);
       end
end
       function [pf,Qa] = check_pf2(d,A,B,C,V) %power factor checker for
leading load
        Q = V.^2*w*-A;

        if (Q>B)
            pf = d;
            Qa = B;
        else
            Qa = Q - B;
            pf = C/sqrt(C.^2+ Qa.^2);
        end
       end
        function [CU,n,n2,Qa] = CU_actual(L,V)
        [CL,Q_new] = CU_ideal(L,V);
        P_L = L(:,2);
        CU = zeros(1, numel(CL));

        % CU = check_pf (CL(12), Q_new(12), P_L(12))

        %%{
        PF = find_pf(L); % Actual power factor
        M1 = max(CL); % maximum capacitor values for CU
        M2 = min (CL); % maximum inductance values for CU
        t = ones (1, numel(CL));
        t1 = ones (1, numel(CL));
        Qa  = zeros(1, numel(CL));
        for k = 1:numel(CL)
            if (Q_new(k) < 0) % If lagging load
                pf = PF(k,2);
                j = 1;
                if (pf>0.98)
                    W = 0;
                    l=0;
                end
                while (pf < 0.98)
                   for l = 1:j
```

```
            [pf, W] = check_pf1(pf, M1 *l/j,Q_new(k),P_L(k),V);
            if (pf >=0.98)
                break
            end
           end
            j = (j+1);

        end
        Qa (k) = W;
         t(k) = j;
        CU(k) = M1*l/(j-1);
    elseif (Q_new(k) > 0) % If leading load
        pf = PF(k,2);
        j = 1;
        if (pf > 0.95)
            W = 0;
            l=0;
        end
        while (pf < 0.95)
            for l = 1:j
            [pf, W] = check_pf2(pf, M2*l/j,Q_new(k),P_L(k),V);
            if (pf >=0.95)
                break
            end
           end
            j = j+1;
        end
         Qa (k) = W;
         t1(k) = j;
        CU(k) = M2*l/(j-1);
    else
        CU(k) = 0;







        %%}
    end

            end

n = lcm(sym(t));
n = double(n);
if (max(CU)>0)
 n = max (CU)/n;
else
    n=0;
end


 n2 = lcm(sym(t1));
n2 = double(n2);
if (min(CU)<0)
 n2 = -1*min (CU)/n2;
```

```
        else
            n2 = 0;
        end
         for k = 1:numel(CU)
             if (CU(k)>=0)
                  CU(k) = CU(k)/n;
                  else
                 CU(k) = CU(k)/n2;
             end
             if (isnan(CU(k)))
             CU(k) = 0;
             end
         end
         g = gcd(sym(CU));
         g = double(g);
         n = g*n;
         n2 = g*n2;
         CU = CU./g;
         end

         function [v,a,b,I]= S_line(P,Q,X,R,V)
         I = sqrt(P.*P + Q.*Q)./V ;        % load current

        %}
         a =  (3*R)*(I.*I);
         b =  (3*X)*(I.*I);
         v = I * sqrt(R.^2+X.^2);

         end
         end


% --- Executes on selection change in Plot_2_display.
function Plot_2_display_Callback(hObject, eventdata, handles)
% hObject    handle to Plot_2_display (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Plot_2_display
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
Plot_2_display

end
% --- Executes during object creation, after setting all properties.
function Plot_2_display_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Plot_2_display (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end
set(hObject, 'String', {'CASE 1 - Load 1', 'CASE 1 - Load 2', 'CASE 1 - Load
3', 'CASE 2', 'Line loading - Load 1','Line loading - Load 2', 'Line loading
- Load 3', 'Power factor at substation', 'Voltage drop on line - Load 1',
'Voltage drop on line - Load 2', 'Voltage drop on line - Load 3' });
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes1);
cla;

popup_sel_index = get(handles.Plot_2_display, 'Value');
guidata(hObject,handles);
switch popup_sel_index
    case 1

        bar (handles.L1(:,1), handles.C1)
        str1 = ['Number of capacitor banks connected in delta or inductor
banks connected in wye with capacitor value ', num2str(handles.n1), ' farads
and inductor value ', num2str(handles.n2), ' henrys' ];
        title (str1)
        xlabel ('Hour')
        ylabel ('Number of capacitor banks')
    case 2

         bar (handles.L2(:,1), handles.C2)
        str2 = ['Number of capacitor banks connected in delta or inductor
banks connected in wye with capacitor value ', num2str(handles.n3), ' farads
and inductor value ', num2str(handles.n4), ' henrys' ];
        title (str2)
        xlabel ('Hour')
        ylabel ('Number of capacitor banks')

    case 3
        bar (handles.L2(:,1), handles.C3)
        str3 = ['Number of capacitor banks connected in delta or inductor
banks connected in wye with capacitor value ', num2str(handles.n5), ' farads
and inductor value ', num2str(handles.n6), ' henrys' ];
        title (str3)
        xlabel ('Hour')
        ylabel ('Number of capacitor banks')

    case 4
         handles.C4
         bar (handles.L1(:,1),handles.C4)
        str3 = ['Number of capacitor banks connected in delta or inductor
banks  connected  in wye with capacitor value  ', num2str(handles.n5), '
farads and inductor value', num2str(handles.n6), 'henrys' ]
        title (str3)
        xlabel ('Hour')
        ylabel ('Number of capacitor  and inductor banks')
        xlim([0,23])
```

```
    case 5
       plot (handles.L1(:,1) , handles.Pl1)
       hold on;
       plot (handles.L1(:,1) , handles.PlL1)
       title ('Line loading for load 1, CASE 1 and CASE 2')
       xlabel ('Hour')
       ylabel ('Power (W)')
       legend ('Case 2','Case 1')

    case 6

       plot (handles.L1(:,1) , handles.Pl2)
       hold on;
       plot (handles.L1(:,1) , handles.PlL2)
       title ('Line loading for load 2, CASE 1 and CASE 2')
       xlabel ('Hour')
       ylabel ('Power (W)')
       legend ('Case 2','Case 1')

    case 7
        plot (handles.L1(:,1) , handles.Pl3)
       hold on;
       plot (handles.L1(:,1) , handles.PlL3)
       title ('Line loading for load 3, CASE 1 and CASE 2')
       xlabel ('Hour')
       ylabel ('Power (W)')
       legend ('Case 2','Case 1')

    case 8
        plot (handles.L1(:,1) , handles.pfCSS(:,2))
       hold on;
       plot (handles.L1(:,1) , handles.pfSS(:,2))
       title ('power factor seen at the substation, CASE 1 and CASE 2')
       xlabel ('Hour')
       ylabel ('pf')
       legend ('Case 2','Case 1')

    case 9
        plot (handles.L1(:,1) ,100*handles.VL1(:,1)./handles.V1 )
       hold on;
       plot (handles.L1(:,1), 100*handles.VLC1(:,1)./handles.V1)
       title ('Percentage Voltage drop on line for load 1, CASE 1 and CASE
2')
       xlabel ('Hour')
       ylabel ('Volatge (V))')
       legend ('Case 2','Case 1')

    case 10
        plot (handles.L1(:,1) ,100*handles.VL2(:,1)./handles.V2 )
       hold on;
       plot (handles.L1(:,1), 100*handles.VLC2(:,1)./handles.V2)
       title ('Percentage voltage drop on line for load 2, CASE 1 and CASE
2')
       xlabel ('Hour')
```

```
        ylabel ('Voltage (V)')
        legend ('Case 2','Case 1')

    case 11
         plot (handles.L1(:,1) ,100* handles.VL3(:,1)./handles.V3 )
        hold on;
        plot (handles.L1(:,1), 100* handles.VLC3(:,1)./handles.V3)
        title ('Percenatge voltage drop on line for load 3, CASE 1 and CASE
2')
        xlabel ('Hour')
        ylabel ('Voltage (V)')
        legend ('Case 2','Case 1')

end
end
```