Alexandria University

Faculty Of Engineering

Electrical Communication and Electronics Engineering Department

Power Systems Project

Transmission Line Parameters

ABCD Parameters

Transmission Line Performance

Team Members: -

1	**Mahmoud Ashraf Hanafy**	**ID**	**21011255**
2	**Youssef Yasser Abo Hussien**	**ID**	**21011633**
3	**Mohamed Hassan Elsakaar**	**ID**	**21011110**
4	**Abdalrahman Younis**	**ID**	**21010793**
5	**Mohamed Mahmoud Shaltoot**	**ID**	**21011223**
6	**Ahmed Abd Allah Motawie**	**ID**	**21010120**
7	**Alaa Roshdy Sakraan**	**ID**	**21010825**
8	**Mahmoud Abdelsamie**	**ID**	**21011273**
9	**Mohamed Gamal Tag Elden**	**ID**	**21011105**
10	**Ebrahim Usama Kamal**	**ID**	**19015159**

Report For:

*Dr / **Hossam Kotb***

Introduction

Our MATLAB code implements a graphical user interface (GUI) for calculating and analyzing the performance of a transmission line.

Key Features:

Calculates DC and AC resistance of the conductor based on its length, resistivity, and diameter.

Calculates inductance and capacitance per phase based on conductor spacing and diameter.

Determines ABCD parameters of the transmission line based on its length, frequency, and calculated parameters.

Analyzes the performance of the transmission line under different load conditions, including varying power and power factor.

Generates plots of efficiency and voltage regulation versus power and power factor.

Functionality:

Users can input various parameters of the transmission line, including conductor properties, length, frequency, and load conditions.

Parameter calculations: The code accurately calculates various electrical parameters of the transmission line based on user-defined inputs. This includes DC and AC resistance, inductance, capacitance, and ABCD parameters.

Model selection: Users can choose between pi and T models, allowing them to analyze the transmission line behavior under different assumptions.

Performance analysis: The code analyzes the transmission line performance under various load conditions, including varying power and power factor. This provides insights into the efficiency and voltage regulation of the line under different operating scenarios.

Visualization: The code generates plots of efficiency and voltage regulation, which visually represent the performance characteristics of the transmission line.

Benefits:

This GUI provides a user-friendly interface for analyzing the performance of transmission lines.

It allows users to quickly calculate and visualize the impact of different parameters on the transmission line’s performance.

The GUI can be a valuable tool for engineers and students working with transmission line design and analysis.

Inputs:

Conductor length (m): Length of the transmission line conductor.

Conductor resistivity (Ω.m): Resistivity of the conductor material.

Conductor diameter (mm or cm): Diameter of the conductor.

Frequency (Hz): Operating frequency of the transmission line.

Transmission line model: Users can choose between pi and T models.

Load conditions: Users can specify the receiving voltage and either the active power or the power factor.

Outputs:

DC resistance (Ω): DC resistance of the conductor.

AC resistance (Ω): AC resistance of the conductor at the specified frequency.

Inductance per phase (H/m): Inductance per phase of the transmission line.

Capacitance per phase (F/m): Capacitance per phase of the transmission line.

ABCD parameters: ABCD parameters of the transmission line, which characterize its electrical behavior.

Efficiency (%): Efficiency of the transmission line under different load conditions.

Voltage regulation (%): Voltage regulation of the transmission line under different load conditions.

Plots:

Efficiency vs. Power: Plot showing the efficiency of the transmission line as a function of active power.

Efficiency vs. Power Factor: Plot showing the efficiency of the transmission line as a function of power factor.

Voltage Regulation vs. Power: Plot showing the voltage regulation of the transmission line as a function of active power.

Voltage Regulation vs. Power Factor: Plot showing the voltage regulation of the transmission line as a function of power factor.

Code Structure:

The code is well-organized and includes comments explaining different sections.

It uses functions to modularize the code and improve readability.

Error handling is implemented to prevent unexpected crashes.

Overall, our MATLAB GUI code offers a powerful and versatile tool for analyzing the performance of transmission lines. Its user-friendly interface and comprehensive functionalities make it a valuable resource for a wide range of applications in power systems engineering.

The Whole Code

```
function varargout = final_report(varargin)
% FINAL_REPORT MATLAB code for final_report.fig
%      FINAL_REPORT, by itself, creates a new FINAL_REPORT or raises the existing
%      singleton*.
%
%      H = FINAL_REPORT returns the handle to a new FINAL_REPORT or the handle to
%      the existing singleton*.
%
%      FINAL_REPORT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FINAL_REPORT.M with the given input arguments.
%
%      FINAL_REPORT('Property','Value',...) creates a new FINAL_REPORT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before final_report_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to final_report_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 final_report

% Last Modified by GUIDE v2.5 05-May-2024 15:54:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @final_report_OpeningFcn, ...
                   'gui_OutputFcn',  @final_report_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


% --- Executes just before final_report is made visible.
function final_report_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 final_report (see VARARGIN)

% Choose default command line output for final_report
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = final_report_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;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (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



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (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



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (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 edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (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
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
conductor_length = str2num(get(handles.edit1,'string'));
conductor_resistivity = str2num(get(handles.edit2,'string'));
conductor_diameter = str2num(get(handles.edit3,'string'));
freq = str2num(get(handles.edit5,'string'));
%%
if get(handles.mmcheckbox,'value') == 1
    dia_mm = str2num(get(handles.edit3,'string'));
    conductor_diameter = dia_mm/1000;
    set(handles.cmcheckbox,'enable','off');
elseif get(handles.cmcheckbox,'value') == 1
    dia_cm = str2num(get(handles.edit3,'string'));
    conductor_diameter = dia_cm/100;
    set(handles.mmcheckbox,'enable','off');
end

%%
dc_resistance = (conductor_resistivity*conductor_length)/( pi.*(conductor_diameter/2).*(conductor_diameter/2));
set(handles.edit4,'string',num2str(dc_resistance));
%%
ac_resistance = 1.1.*dc_resistance;
set(handles.edit10,'string',num2str(ac_resistance));
%%
if get(handles.symcheckbox,'value') == 1
    D_eq = str2num(get(handles.Deq,'string'));
    set(handles.unsymcheckbox,'enable','off');
    set(handles.D1,'enable','off');
    set(handles.D2,'enable','off');
    set(handles.D3,'enable','off');
elseif get(handles.unsymcheckbox,'value') == 1
    d1 = str2num(get(handles.D1,'string'));
    d2 = str2num(get(handles.D2,'string'));
    d3 = str2num(get(handles.D3,'string'));
    D_eq = (d1*d2*d3).^(1/3);
    set(handles.Deq,'enable','off');
    set(handles.symcheckbox,'enable','off');
end
inductance_per_phase = 2*(10)^(-7).*log( D_eq / ( 0.7788.*(conductor_diameter/2) ) )*conductor_length;
set(handles.inductance,'string',num2str(inductance_per_phase));
%%
capacitance_per_phase  =  (  (  2.*pi.*(8.854187817.*10.^-12)  )  ./  log(D_eq  ./  (conductor_diameter./2)  )  )  *
conductor_length;
set(handles.capacitance,'string',num2str(capacitance_per_phase));
%%
Z=complex(ac_resistance,2*pi*freq*inductance_per_phase);
Y=complex(0,2*pi*freq*capacitance_per_phase);
if conductor_length < 80000
    A =1;
    B = Z;
    C = 0;
    D = 1;
 elseif  conductor_length >= 80000
    t1 = 1 + Z*Y/2;
    t2 = 1 + Z*Y/4;
    if get(handles.pi_model,'value')==1
        A = t1;
        B = Z;
        C = Y.*t2;
        D = t1;
        set(handles.T_model,'enable','off');
    elseif get(handles.T_model,'value')==1
        A = t1;
        B = Z.*t2;
        C = Y;
        D = t1;
        set(handles.pi_model,'enable','off');
    end
end
A_string = strcat(num2str(real(A)),' + ','j',num2str(imag(A)));
B_string = strcat(num2str(real(B)),' + ','j',num2str(imag(B)));
C_string = strcat(num2str(real(C)),' + ','j',num2str(imag(C)));
D_string = strcat(num2str(real(D)),' + ','j',num2str(imag(D)));

set(handles.A,'string',A_string);
set(handles.B,'string',B_string);
set(handles.C,'string',C_string);
set(handles.D,'string',D_string);
%%
Receiving_Voltage = str2num(get(handles.receiving_voltage,'string'))/sqrt(3);
if get(handles.checkbox7,'value') == 1
    Receiving_PowerFactor = 0.8;
                sign =-1;
                Receiving_Power = linspace(0, 100000, 1000);

                eff= zeros (1000);
                VR= zeros (1000);
                Receiving_Current= zeros (1000);
```

```
                Sending_Voltage= zeros (1000);
                Sending_Current= zeros (1000);
                Sending_PowerFactor= zeros (1000);
                Sending_Power= zeros (1000);


                for i = 1:1000
                    Receiving_Current(i)     =(     Receiving_Power(i)     /     (3     *     Receiving_Voltage     *
Receiving_PowerFactor))*exp(1i * acos(Receiving_PowerFactor)* (pi/180) * sign);
                    Sending_Voltage(i)= A* Receiving_Voltage + B*Receiving_Current(i);
                    Sending_Current(i)= C* Receiving_Voltage + D*Receiving_Current(i);
                    Sending_PowerFactor(i)= cos(angle(Sending_Voltage(i)) - angle(Sending_Current(i)));
                    Sending_Power(i)    =    3    *    abs(Sending_Voltage(i))    *    abs(Sending_Current(i))    *
Sending_PowerFactor(i);
                    eff(i) = Receiving_Power(i)*100 / Sending_Power(i);
                    VR(i) = (abs(Sending_Voltage(i))/abs(A) - abs( Receiving_Voltage))*100 / abs(Receiving_Voltage)
;
                end

    axes(handles.axes1);
    title('Efficiency')
    plot(Receiving_Power, eff);
    ylim([0 100]);
    ylabel('Efficiency')
    xlabel('Active Power (W)')

    axes(handles.axes2);
    title('Voltage Regulation')
    plot(Receiving_Power, VR);
    ylabel('Voltage Regulation')
    xlabel('Active Power (W)')
    set(handles.checkbox8,'enable','off');
elseif get(handles.checkbox8,'value') == 1
    Receiving_PowerFactor = linspace(0.3, 1, 1000);
                sign=-1;
                Receiving_Power  = 100000;

                eff= zeros (1000);
                VR= zeros (1000);
                Receiving_Current= zeros (1000);
                Sending_Voltage= zeros (1000);
                Sending_Current= zeros (1000);
                Sending_PowerFactor= zeros (1000);
                Sending_Power= zeros (1000);
                for i = 1:1000
                    Receiving_Current(i)     =(     Receiving_Power     /     (3     *     Receiving_Voltage     *
Receiving_PowerFactor(i)))*exp(1i * acos(Receiving_PowerFactor(i))* (pi/180) * sign);
                    Sending_Voltage(i)= A* Receiving_Voltage + B*Receiving_Current(i);
                    Sending_Current(i)= C* Receiving_Voltage + D*Receiving_Current(i);
                    Sending_PowerFactor(i)= cos(angle(Sending_Voltage(i)) - angle(Sending_Current(i)));
                    Sending_Power(i)    =    3    *    abs(Sending_Voltage(i))    *    abs(Sending_Current(i))    *
Sending_PowerFactor(i);
                    eff(i) = Receiving_Power*100 / Sending_Power(i);
                    VR(i) = (abs(Sending_Voltage(i))/abs(A) - abs( Receiving_Voltage))*100 / abs(Receiving_Voltage)
;
                end
    axes(handles.axes1);
    title('Efficiency')
    plot(abs(Receiving_PowerFactor), eff);
    ylabel('Efficiency')
    xlabel('Power Factor - Lagging')

    axes(handles.axes2);
    title('Voltage Regulation')
    plot(abs(Receiving_PowerFactor), VR);
    ylabel('Voltage Regulation')
    xlabel('Power Factor - Lagging')

                sign=1;
                eff= zeros (1000);
                VR= zeros (1000);
                Receiving_Current= zeros (1000);
                Sending_Voltage= zeros (1000);
                Sending_Current= zeros (1000);
                Sending_PowerFactor= zeros (1000);
                Sending_Power= zeros (1000);


                for i = 1:1000
                    Receiving_Current(i)     =(     Receiving_Power     /     (3     *     Receiving_Voltage     *
Receiving_PowerFactor(i)))*exp(1i * acos(Receiving_PowerFactor(i))* (pi/180) * sign);
                    Sending_Voltage(i)= A* Receiving_Voltage + B*Receiving_Current(i);
                    Sending_Current(i)= C* Receiving_Voltage + D*Receiving_Current(i);
                    Sending_PowerFactor(i)= cos(angle(Sending_Voltage(i)) - angle(Sending_Current(i)));
                    Sending_Power(i)    =    3    *    abs(Sending_Voltage(i))    *    abs(Sending_Current(i))    *
Sending_PowerFactor(i);
                    eff(i) = Receiving_Power*100 / Sending_Power(i);
                    VR(i) = (abs(Sending_Voltage(i))/abs(A) - abs( Receiving_Voltage))*100 / abs(Receiving_Voltage)
;
                end
```

```
    axes(handles.axes3);
    title('Efficiency')
    plot(abs(Receiving_PowerFactor), eff);
    ylabel('Efficiency ')
    xlabel('Power Factor - Leading')

    axes(handles.axes4);
    title('Voltage Regulation')
    plot(abs(Receiving_PowerFactor), VR);
    ylabel('Voltage Regulation')
    xlabel('Power Factor - Leading')
    set(handles.checkbox7,'enable','off');
end
%%


% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


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

% Hint: get(hObject,'Value') returns toggle state of cmcheckbox


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

% Hint: get(hObject,'Value') returns toggle state of mmcheckbox


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
if get(handles.mmcheckbox,'value') == 1
    set(handles.cmcheckbox,'enable','on');
elseif get(handles.cmcheckbox,'value') == 1
    set(handles.mmcheckbox,'enable','on');
end

if get(handles.symcheckbox,'value') == 1
    set(handles.unsymcheckbox,'enable','on');
    set(handles.D1,'enable','on');
    set(handles.D2,'enable','on');
    set(handles.D3,'enable','on');
elseif get(handles.unsymcheckbox,'value') == 1
    set(handles.Deq,'enable','on');
    set(handles.symcheckbox,'enable','on');
end

if get(handles.pi_model,'value') == 1
    set(handles.T_model,'enable','on');
elseif get(handles.T_model,'value') == 1
    set(handles.pi_model,'enable','on');
end

if get(handles.checkbox7,'value') == 1
    set(handles.checkbox8,'enable','on');
elseif get(handles.checkbox8,'value') == 1
    set(handles.checkbox7,'enable','on');
end
x = 0;
set(handles.checkbox7,'value',x);
set(handles.checkbox8,'value',x);
set(handles.mmcheckbox,'value',x);
set(handles.cmcheckbox,'value',x);
set(handles.symcheckbox,'value',x);
set(handles.unsymcheckbox,'value',x);
set(handles.pi_model,'value',x);
set(handles.T_model,'value',x);
set(handles.edit2,'string',num2str(x));
set(handles.edit1,'string',num2str(x));
set(handles.edit3,'string',num2str(x));
set(handles.edit5,'string',num2str(x));
set(handles.Deq,'string',num2str(x));
set(handles.D1,'string',num2str(x));
set(handles.D2,'string',num2str(x));
set(handles.D3,'string',num2str(x));
set(handles.receiving_voltage,'string',num2str(x));
set(handles.edit4,'string',num2str(x));
set(handles.edit10,'string',num2str(x));
set(handles.inductance,'string',num2str(x));
```

```
set(handles.capacitance,'string',num2str(x));
set(handles.A,'string',num2str(x));
set(handles.B,'string',num2str(x));
set(handles.C,'string',num2str(x));
set(handles.D,'string',num2str(x));
clc
cla(handles.axes1);
cla(handles.axes2);
cla(handles.axes3);
cla(handles.axes4);



clear

% 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)


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
close
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (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 edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (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



function Deq_Callback(hObject, eventdata, handles)
% hObject    handle to Deq (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 Deq as text
%        str2double(get(hObject,'String')) returns contents of Deq as a double


% --- Executes during object creation, after setting all properties.
function Deq_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Deq (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



function D1_Callback(hObject, eventdata, handles)
% hObject    handle to D1 (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 D1 as text
%        str2double(get(hObject,'String')) returns contents of D1 as a double


% --- Executes during object creation, after setting all properties.
function D1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to D1 (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



function D2_Callback(hObject, eventdata, handles)
% hObject    handle to D2 (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 D2 as text
%        str2double(get(hObject,'String')) returns contents of D2 as a double


% --- Executes during object creation, after setting all properties.
function D2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to D2 (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



function D3_Callback(hObject, eventdata, handles)
% hObject    handle to D3 (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 D3 as text
%        str2double(get(hObject,'String')) returns contents of D3 as a double


% --- Executes during object creation, after setting all properties.
function D3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to D3 (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



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

% Hint: get(hObject,'Value') returns toggle state of symcheckbox


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

% Hint: get(hObject,'Value') returns toggle state of unsymcheckbox



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (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 edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit10 (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



function inductance_Callback(hObject, eventdata, handles)
% hObject    handle to inductance (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 inductance as text
%        str2double(get(hObject,'String')) returns contents of inductance as a double


% --- Executes during object creation, after setting all properties.
function inductance_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inductance (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



function capacitance_Callback(hObject, eventdata, handles)
% hObject    handle to capacitance (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 capacitance as text
%        str2double(get(hObject,'String')) returns contents of capacitance as a double


% --- Executes during object creation, after setting all properties.
function capacitance_CreateFcn(hObject, eventdata, handles)
% hObject    handle to capacitance (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



function A_Callback(hObject, eventdata, handles)
% hObject    handle to A (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 A as text
%        str2double(get(hObject,'String')) returns contents of A as a double


% --- Executes during object creation, after setting all properties.
function A_CreateFcn(hObject, eventdata, handles)
% hObject    handle to A (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



function B_Callback(hObject, eventdata, handles)
% hObject    handle to B (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 B as text
%        str2double(get(hObject,'String')) returns contents of B as a double


% --- Executes during object creation, after setting all properties.
function B_CreateFcn(hObject, eventdata, handles)
% hObject    handle to B (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



function C_Callback(hObject, eventdata, handles)
% hObject    handle to C (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 C as text
%        str2double(get(hObject,'String')) returns contents of C as a double


% --- Executes during object creation, after setting all properties.
function C_CreateFcn(hObject, eventdata, handles)
% hObject    handle to C (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



function D_Callback(hObject, eventdata, handles)
% hObject    handle to D (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 D as text
%        str2double(get(hObject,'String')) returns contents of D as a double


% --- Executes during object creation, after setting all properties.
function D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to D (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


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

% Hint: get(hObject,'Value') returns toggle state of pi_model


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

% Hint: get(hObject,'Value') returns toggle state of T_model



function receiving_voltage_Callback(hObject, eventdata, handles)
% hObject    handle to receiving_voltage (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 receiving_voltage as text
%        str2double(get(hObject,'String')) returns contents of receiving_voltage as a double


% --- Executes during object creation, after setting all properties.
function receiving_voltage_CreateFcn(hObject, eventdata, handles)
% hObject    handle to receiving_voltage (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



% --- Executes on button press in checkbox7.
function checkbox7_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox7



% --- Executes on button press in checkbox8.
function checkbox8_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox8


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

General Explanation of The Code Functionality

Our code defines the functionality of a graphical user interface (GUI) for calculating and analyzing the performance of a transmission line.

1. Initialization:

The code starts by defining the GUI layout and callback functions.

-It sets up the initial values of the GUI components like edit boxes, checkboxes, pushbuttons, and axes.

- The `final_report_OpeningFcn` function initializes the GUI components and sets their initial values.

- The `edit1_Callback`, `edit2_Callback`, etc. functions handle user input in the edit boxes.

- The `edit1_CreateFcn`, `edit2_CreateFcn`, etc. functions set the background color of the edit boxes.

- The `pushbutton1_Callback` function is the main function that triggers the calculations and updates the output.

2. User Input:

- The user can enter values for the following parameters:
  - `edit1`: Conductor length (m)
  - `edit2`: Conductor resistivity (Ω-m)
  - `edit3`: Conductor diameter (mm or cm)
  - `edit5`: Frequency (Hz)
  - `receiving_voltage`: Receiving voltage (V)
  - `checkbox7`: Varying power (active power)
  - `checkbox8`: Varying power factor
  - `mmcheckbox`: Units for conductor diameter (mm)
  - `cmcheckbox`: Units for conductor diameter (cm)
  - `symcheckbox`: Symmetrical line configuration
  - `unsymcheckbox`: Unsymmetrical line configuration
  - `pi_model`: π model for transmission line
  - `T_model`: T model for transmission line
  - `Deq`: Equivalent diameter for unsymmetrical lines
  - `D1`, `D2`, `D3`: Conductor diameters for unsymmetrical lines

3. Calculations:

- The `pushbutton1_Callback` function performs the following calculations:
  - Calculates DC resistance of the conductor using `conductor_length`, `conductor_resistivity`, and `conductor_diameter`.
  - Calculates AC resistance of the conductor using `dc_resistance` and a correction factor.
  - Determines the equivalent diameter for unsymmetrical lines using `D1`, `D2`, and `D3`.
  - Calculates inductance per phase using `conductor_length`, `conductor_diameter`, and `freq`.
  - Calculates capacitance per phase using `conductor_length`, `D_eq`, and `conductor_diameter`.
  - Calculates ABCD parameters of the transmission line model based on `ac_resistance`, `inductance_per_phase`, `capacitance_per_phase`, and `freq`.
  - Calculates sending voltage, current, and power based on ABCD parameters, receiving voltage, and power/power factor.

- Calculates efficiency and voltage regulation at different power levels and power factors.

4. Output:

- The calculated values are displayed in the corresponding edit boxes.

  - `edit4`: DC resistance

  - `edit10`: AC resistance

  - `inductance`: Inductance per phase

  - `capacitance`: Capacitance per phase

  - `A`, `B`, `C`, `D`: ABCD parameters

- Efficiency and voltage regulation curves are plotted on the axes for different power levels and power factors.

5. Control and Functionality:

- The code includes various checkboxes to allow the user to select different options and control the calculations.

- There are buttons to perform calculations, reset the GUI, and close the window.

- The checkboxes allow the user to select different options and control the calculations:

  - `mmcheckbox` and `cmcheckbox`: Units for conductor diameter

  - `symcheckbox` and `unsymcheckbox`: Line configuration

  - `pi_model` and `T_model`: Transmission line model

  - `checkbox7` and `checkbox8`: Varying power or power factor

- The buttons provide control over the GUI:

  - `pushbutton1`: Perform calculations

  - `pushbutton2`: Reset GUI

  - `pushbutton3`: Close GUI

6. Additional Notes:

- The code includes comments to explain the functionality of different sections.

- The code uses functions like `str2num` to convert string input to numerical values.

- The code uses complex numbers to represent AC quantities.

- The code uses conditional statements to handle different user selections.

7. Specific Functionalities:

- `pushbutton1_Callback`: This function performs the calculations and updates the output based on the user input.

- `mmcheckbox_Callback` and `cmcheckbox_Callback`: These functions convert the conductor diameter between mm and cm units.

- `symcheckbox_Callback` and `unsymcheckbox_Callback`: These functions enable/disable the relevant input fields for symmetrical and unsymmetrical line configurations.

- `pi_model_Callback` and `T_model_Callback`: These functions enable/disable the relevant input fields for π and T models.

- `checkbox7_Callback` and `checkbox8_Callback`: These functions allow the user to select between varying power and power factor.

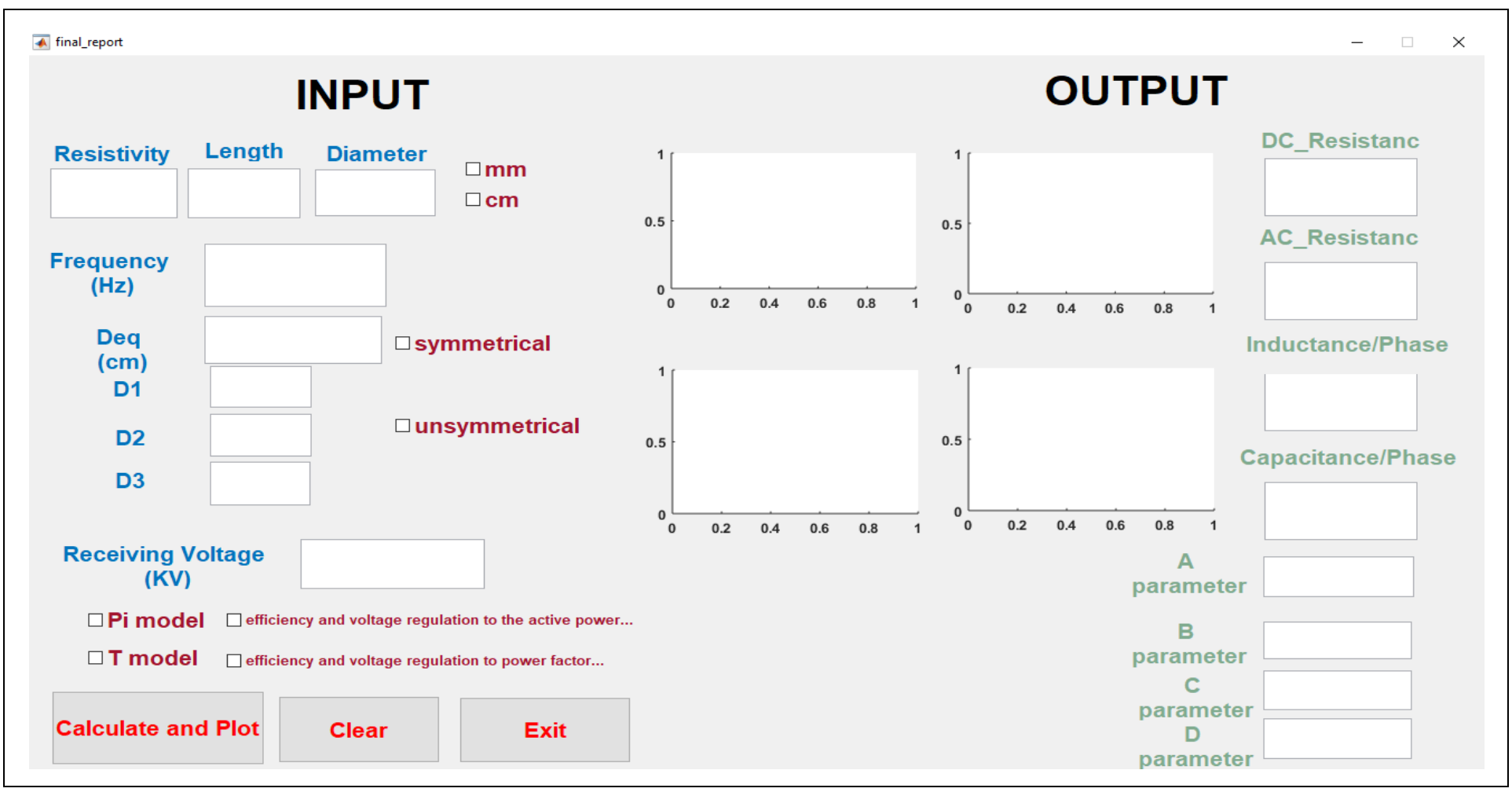
- `pushbutton2_Callback`: This function resets the GUI to its initial state.

- `pushbutton3_Callback`: This function closes the GUI window.

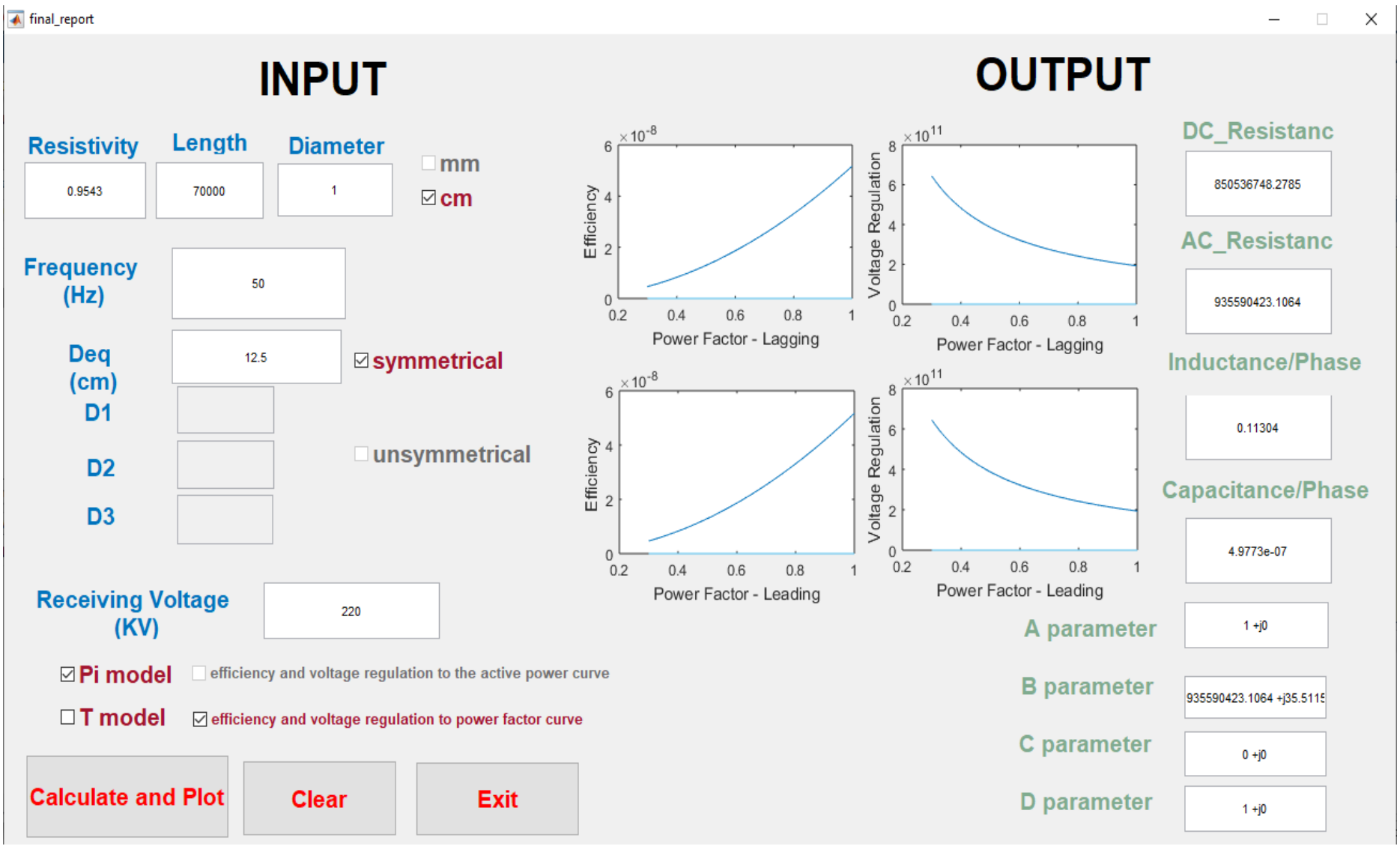
Conclusion:

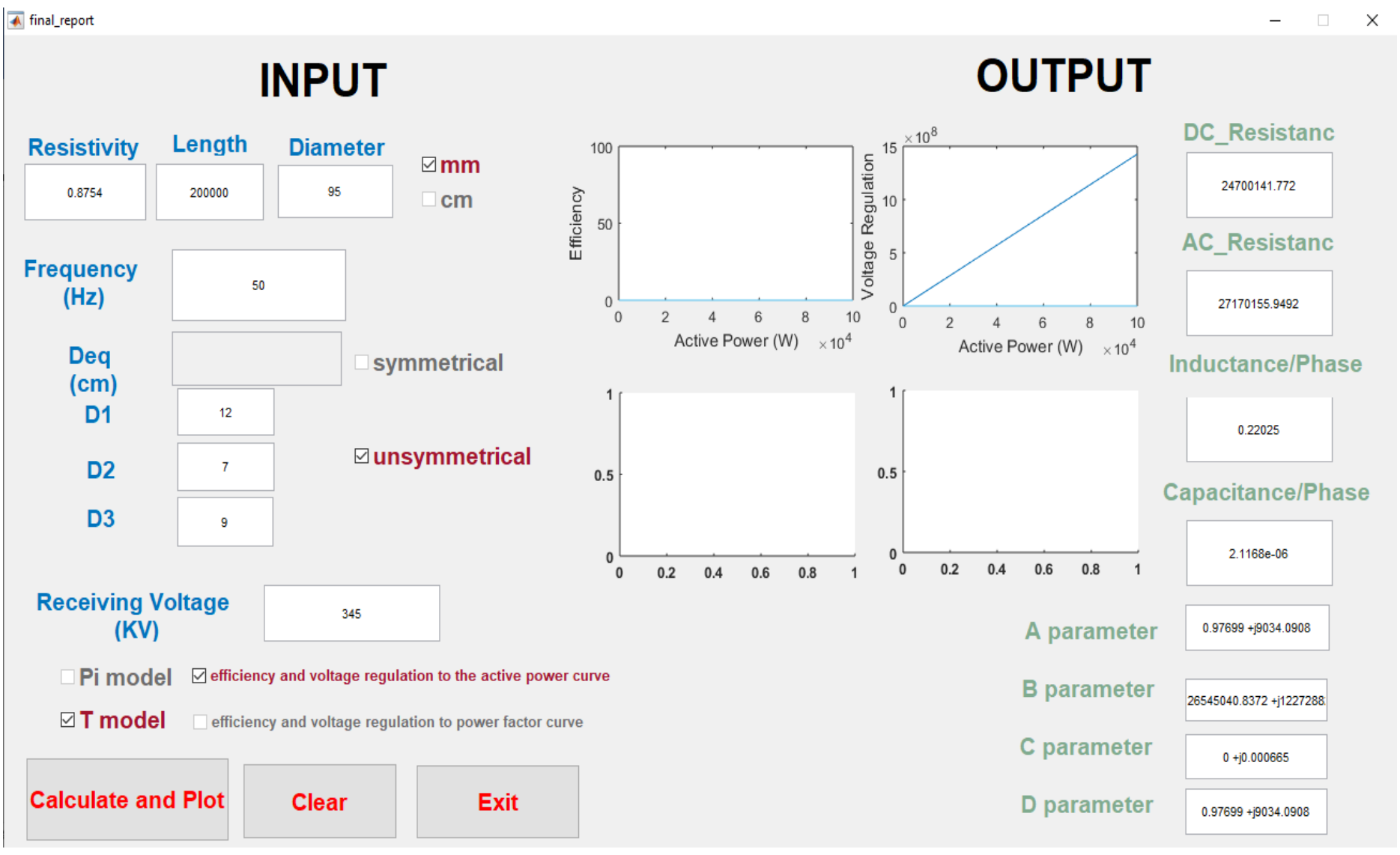
Our MATLAB GUI code is a well-structured and functional program for calculating and analyzing the performance of a transmission line. It offers a user-friendly interface and various options to control the calculations and visualize the results. The code is well-commented and uses appropriate functions.

GUI Window



Inputs and Outputs





final_report

INPUT

Resistivity: 0.6*10^(-3)
Length: 250000
Diameter: 0.75
☐ mm
☑ cm

Frequency (Hz): 50

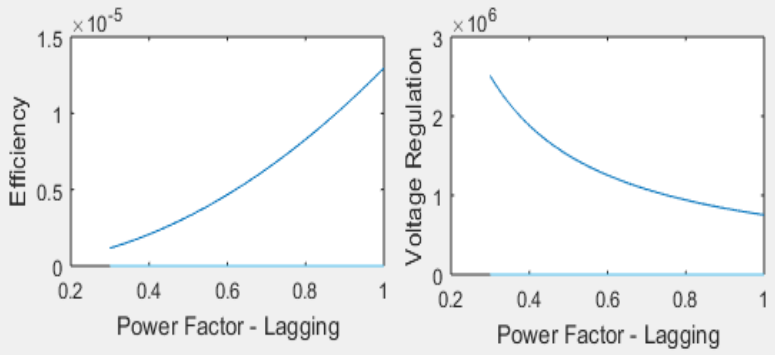
Deq (cm): 10.74
☑ symmetrical
D1: 0
D2: 0
D3: 0
☐ unsymmetrical

Receiving Voltage (KV): 220

☑ Pi model
☐ T model
☐ efficiency and voltage regulation to the active power curve
☑ efficiency and voltage regulation to power factor curve

Calculate and Plot | Clear | Exit

OUTPUT



DC_Resistanc: 3395305.4526
AC_Resistanc: 3734835.9979
Inductance/Phase: 0.4105
Capacitance/Phase: 1.7473e-06
A parameter: 0.96461 +j1025.0578
B parameter: 3734835.9979 +j128.962
C parameter: -0.28134 +j0.0005392
D parameter: 0.96461 +j1025.0578

final_report

INPUT

Resistivity: 0.974*10^(-4)
Length: 300000
Diameter: 67
☑ mm
☐ cm

Frequency (Hz): 50

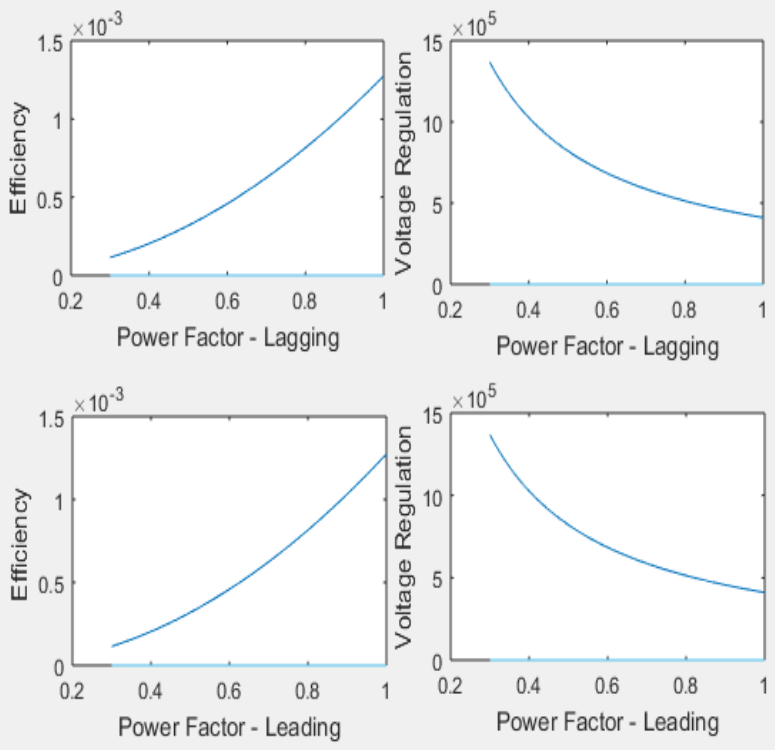
Deq (cm): 0
☐ symmetrical
D1: 10
D2: 9
D3: 7
☑ unsymmetrical

Receiving Voltage (KV): 345

☐ Pi model
☑ T model
☐ efficiency and voltage regulation to the active power curve
☑ efficiency and voltage regulation to power factor curve

Calculate and Plot | Clear | Exit

OUTPUT



DC_Resistanc: 8287.8279
AC_Resistanc: 9116.6107
Inductance/Phase: 0.34769
Capacitance/Phase: 3.01e-06
A parameter: 0.94836 +j4.3104
B parameter: 8645.7885 +j19754.5733
C parameter: 0 +j0.00094562
D parameter: 0.94836 +j4.3104