BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Report on the project

Transformers: Voltage Regulation and Efficiency

Course- EEE 212: Numerical Technique Sessional

Submitted to:

I.K.M. Reaz Rahman
Lecturer, Dept. of EEE

Dr.Md. Shah Alam
Professor, Dept. of EEE

Submitted by:

Sudipta Saha (1706162)
Raisa Mashtura (1706163)

Group no. 16
Section: C1
Level: 2 ***Term:*** 1

Date of submission: 14/09/2019

OBJECTIVE:

The objective of this project was to plot:

- the variation of voltage regulation with load
- the variation of efficiency with load
- The equivalent circuit used in obtaining the voltage regulation

For both three-phase (Y-Y, Δ-Y, Δ-Δ, Y-Δ) transformers and single phase transformers.

The parameters are obtained from the user via a GUI interface and on pressing the push button, the main code is executed and the figure and output plots are displayed on the GUI window.

EQUATIONS AND CALCULATIONS:

The turns ratio from primary to secondary windings vary with the type of connection of the transformer:

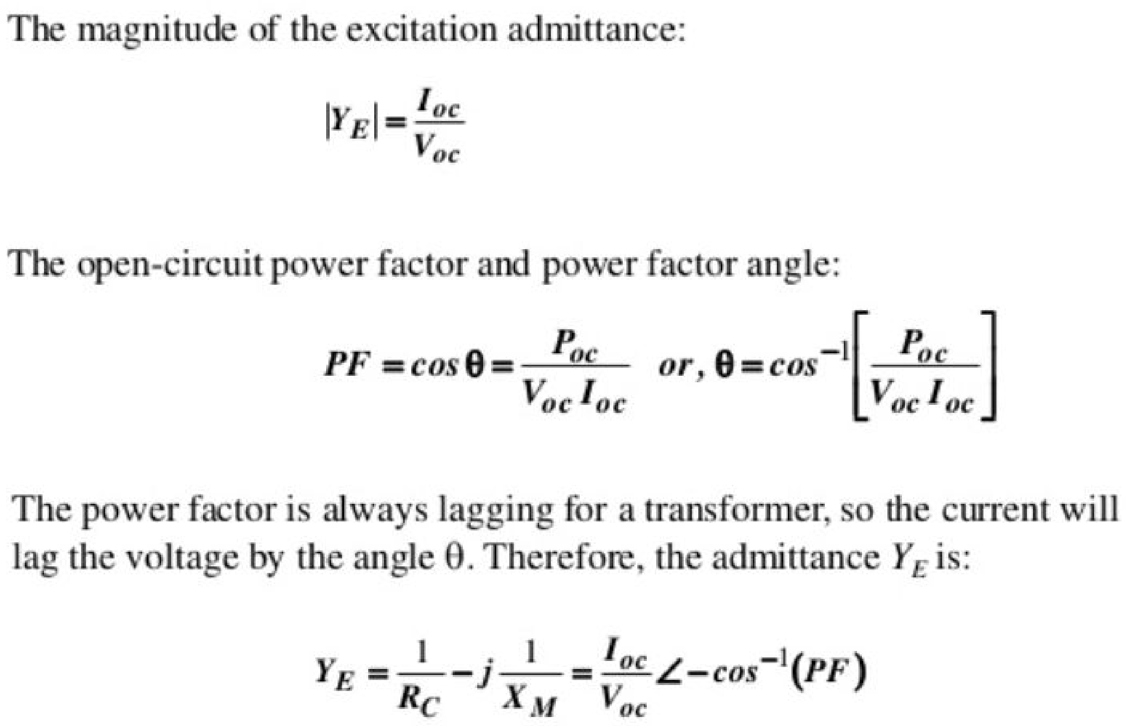
Type of connection	Turns ratio,a
Y-Y	Vp_line/Vs_line
Y-Δ	Vp_line/√3Vs_line
Δ-Δ	Vp_line/Vs_line
Δ-Y	√3Vp_line/Vs_line
Single phase	Vp_line/Vs_line

Short circuit tests are conducted referred to high voltage side and open circuit tests are conducted referred to low voltage side.

To calculate the core resistance and reactance the open circuit voltage, current and power are used.

To calculate the equivalent resistance and reactance, the short circuit voltage, current and power are used.

<u>Calculation of Rc, Xm:</u>



The magnitude of the excitation admittance:

$$|Y_E| = \frac{I_{oc}}{V_{oc}}$$

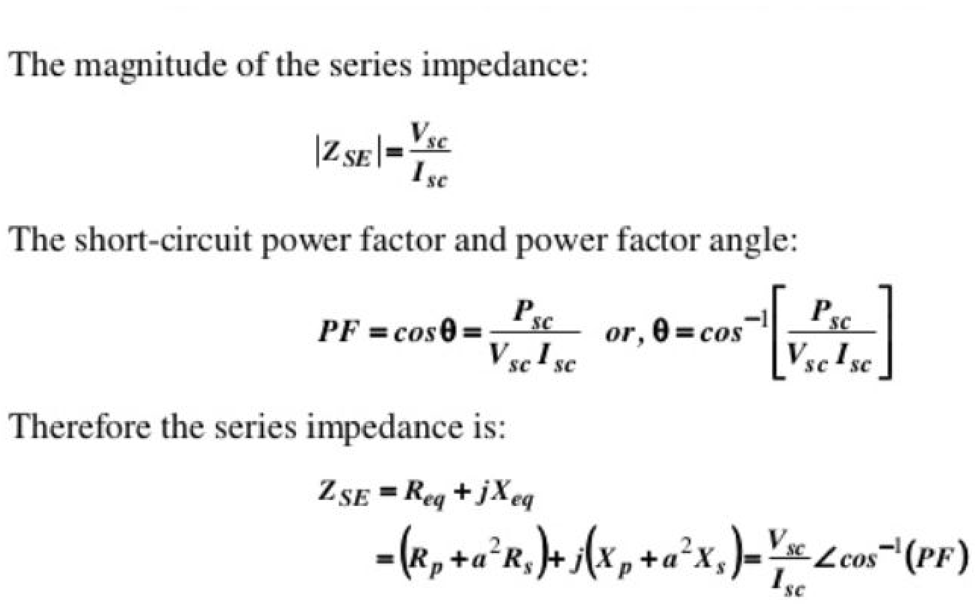
The open-circuit power factor and power factor angle:

$$PF = \cos\theta = \frac{P_{oc}}{V_{oc} I_{oc}} \quad or,\ \theta = \cos^{-1}\left[\frac{P_{oc}}{V_{oc} I_{oc}}\right]$$

The power factor is always lagging for a transformer, so the current will lag the voltage by the angle θ. Therefore, the admittance Y_E is:

$$Y_E = \frac{1}{R_C} - j\frac{1}{X_M} = \frac{I_{oc}}{V_{oc}}\angle -\cos^{-1}(PF)$$

Calculation of Req, Xeq:



The magnitude of the series impedance:

$$|Z_{SE}| = \frac{V_{sc}}{I_{sc}}$$

The short-circuit power factor and power factor angle:

$$PF = cos\theta = \frac{P_{sc}}{V_{sc} I_{sc}} \quad or, \theta = cos^{-1}\left[\frac{P_{sc}}{V_{sc} I_{sc}}\right]$$

Therefore the series impedance is:

$$\begin{aligned} Z_{SE} &= R_{eq} + jX_{eq} \\ &= \left(R_p + a^2 R_s\right) + j\left(X_p + a^2 X_s\right) = \frac{V_{sc}}{I_{sc}} \angle cos^{-1}(PF) \end{aligned}$$

Calculation of Voltage-Regulation:

For a≥1 we calculated VR by taking all values with respect to primary side.

We varied the load current from 0 to rated value taking intervals of 0.01A.

Equations we used for different types of connections:-

1)Single phase:

$aV_s = V_p - (r_{eq} \times I + j.x_{eq} \times I)$

Voltage regulation, VR= $(V_p - aV_s)/(\ aV_s) \times 100\ \%$

2)Three phase:

$aV_{s\text{Ø}} = V_{p\text{Ø}} - (\ r_{eq}I + j.x_{eq}I)$

Voltage regulation, VR=$(V_{pØ} -aV_{sØ})/(aV_{sØ}) \times 100$ %

For a<1 we calculated VR by taking all values with respect to the secondary side.

We varied the load current from 0 to rated value taking intervals of 0.01A.

Equations we used for different types of connections:-

1)Single phase:

$V_p /a =V_s+(r_{eq}I+j.x_{eq}I)$

Voltage regulation, VR=$(V_p/a-V_s)/ V_s \times 100$ %

2)Three phase:

$V_{pØ} /a=V_{sØ} +(r_{eq}I+j.x_{eq}I)$

Voltage regulation, VR=$(V_{pØ} /a- V_{sØ})/ V_{sØ}\times 100$ %

Calculation of Efficiency:

We calculated efficiency by taking all values with respect to secondary side.

We varied load current from 0 to rated value taking intervals of 0.01A.

Equations we used for different types of connection:

1) Single phase:

$P_{copper}=I_s^2r_{eq}$

$P_{core}=(V_p/a)^2/R_c$

$P_{out}=V_sI_s\times pf$

$P_{in} = P_{copper} + P_{core} + P_{out}$

$\text{Efficiency} = P_{out} / P_{in} \times 100\%$

2)Three-phase:

$P_{copper} = 3 I_s^2 r_{eq}$

$P_{core} = 3 (V_p / a)^2 / R_c$

$P_{out} = \sqrt{3} V_{s\text{-}line} I_{s\text{-}line} \times pf$

$P_{in} = P_{copper} + P_{core} + P_{out}$

$\text{Efficiency} = P_{out} / P_{in} \times 100\%$

INSTRUCTION FOR USERS:

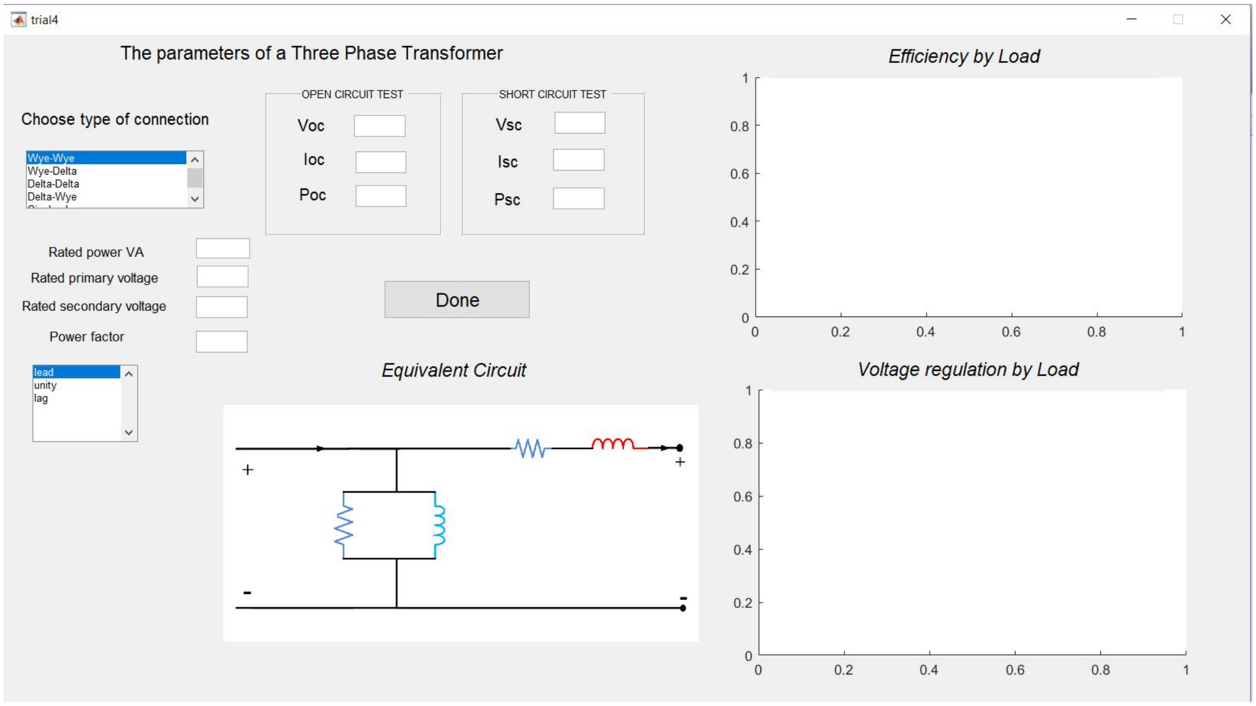


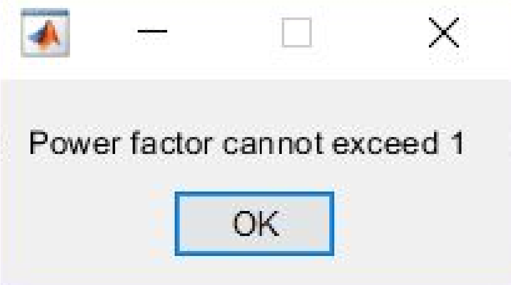
Figure: GUI window

Type of connection:

The user must choose a type of connection from the following:
Y-Y, Δ-Y, Δ-Δ, Y-Δ, Single phase.

Transformer ratings:

The user has to enter the rated power in VA, rated current in A, rated voltage in V and the power factor for which the voltage regulation values are to be obtained. The power factor cannot exceed 1 otherwise the following message box will appear.



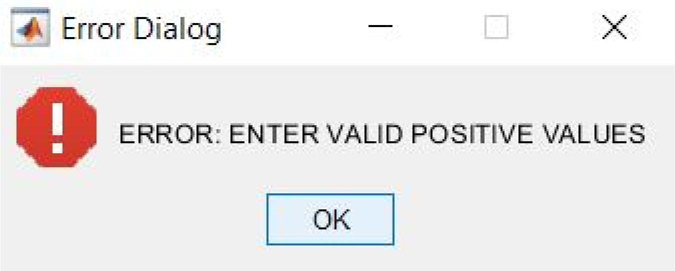
Power factor cannot exceed 1

Fig. Message box

Open and short circuit test parameters:

The user has to insert the parameters obtained from short-circuit and open-circuit tests.

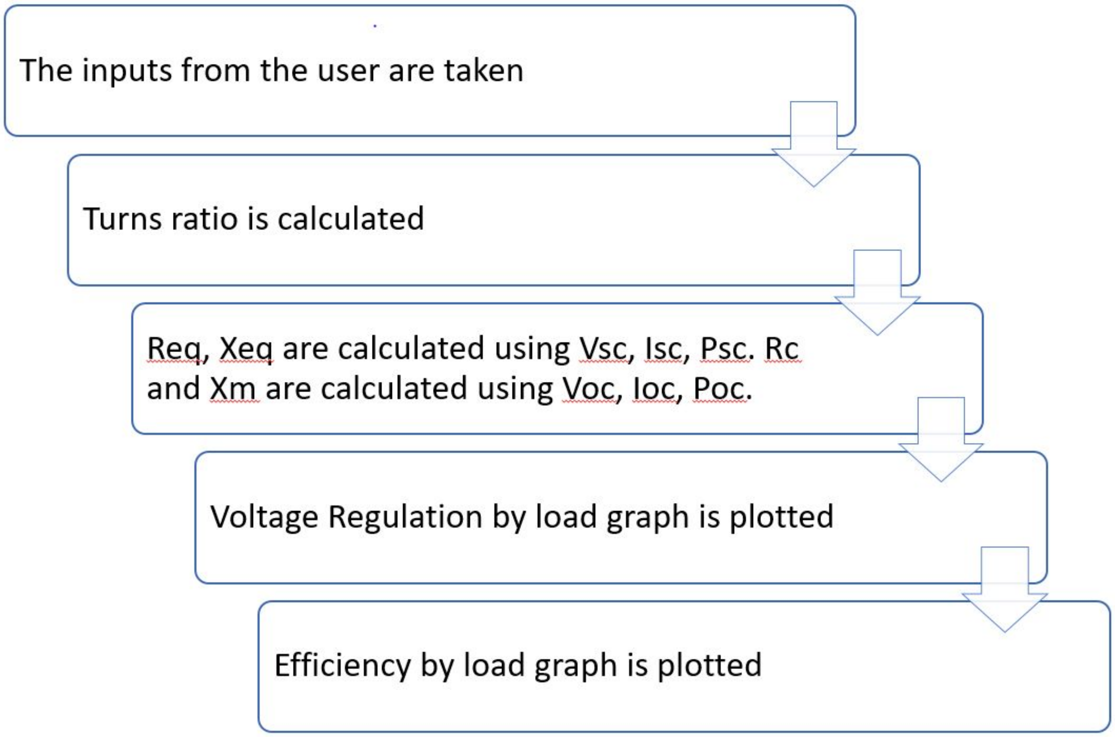
None of the aforementioned values can be negative. Otherwise an error dialogue box will appear:



ERROR: ENTER VALID POSITIVE VALUES

Fig. Error dialogue box

METHODOLOGY:



The inputs from the user are taken

Turns ratio is calculated

Req, Xeq are calculated using Vsc, Isc, Psc. Rc and Xm are calculated using Voc, Ioc, Poc.

Voltage Regulation by load graph is plotted

Efficiency by load graph is plotted

APPENDIX:

```
function varargout = trial4(varargin)
% TRIAL4 MATLAB code for trial4.fig
%      TRIAL4, by itself, creates a new TRIAL4 or raises the
existing
%      singleton*.
%
%      H = TRIAL4 returns the handle to a new TRIAL4 or the
handle to
%      the existing singleton*.
%
%      TRIAL4('CALLBACK',hObject,eventData,handles,...) calls
the local
```

```
%      function named CALLBACK in TRIAL4.M with the given input
arguments.
%
%      TRIAL4('Property','Value',...) creates a new TRIAL4 or
raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before trial4_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to trial4_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 trial4

% Last Modified by GUIDE v2.5 08-Sep-2019 23:47:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @trial4_OpeningFcn, ...
                   'gui_OutputFcn',  @trial4_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 trial4 is made visible.
function trial4_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 trial4 (see VARARGIN)

% Choose default command line output for trial4
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes trial4 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = trial4_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 vp_Callback(hObject, eventdata, handles)
% hObject    handle to vp (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 vp as text
%        str2double(get(hObject,'String')) returns contents of
vp as a double
vp=str2double(get(hObject,'String'));
setappdata(0,'vp',vp);


% --- Executes during object creation, after setting all
properties.
function vp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vp (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 vs_Callback(hObject, eventdata, handles)
% hObject    handle to vs (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 vs as text
%        str2double(get(hObject,'String')) returns contents of
vs as a double
vs=str2double(get(hObject,'String'));
setappdata(0,'vs',vs);


% --- Executes during object creation, after setting all
properties.
function vs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vs (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 pf_Callback(hObject, eventdata, handles)
% hObject    handle to pf (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 pf as text
%        str2double(get(hObject,'String')) returns contents of
pf as a double
pf=str2double(get(hObject,'String'));
setappdata(0,'pf',pf);
```

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



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

```
va=str2double(get(hObject,'String'));
setappdata(0,'va',va);


% --- Executes during object creation, after setting all
properties.
function va_CreateFcn(hObject, eventdata, handles)
% hObject    handle to va (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 selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (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
listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from listbox1
index_selected2 = get(hObject,'Value');
list2 = get(hObject,'String');
item_selected2 = list2{index_selected2};
handles.listbox1 = index_selected2;
disp(handles);
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all
properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: listbox 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 selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (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
listbox2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from listbox2
index_selected = get(hObject,'Value');
% disp(index_selected);
list = get(hObject,'String');
% disp(list);
item_selected = list{index_selected};
% disp(item_selected);
handles.listbox2 = index_selected;
```

```
disp(handles);
guidata(hObject, handles);
% l=get(handles.listbox1,'Value');


% --- Executes during object creation, after setting all
properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: listbox 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 vsc_Callback(hObject, eventdata, handles)
% hObject    handle to vsc (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 vsc as text
%        str2double(get(hObject,'String')) returns contents of
vsc as a double
vsc=str2double(get(hObject,'String'));
setappdata(0,'vsc',vsc);


% --- Executes during object creation, after setting all
properties.
```

```
function vsc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vsc (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 isc_Callback(hObject, eventdata, handles)
% hObject    handle to isc (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 isc as text
%        str2double(get(hObject,'String')) returns contents of
isc as a double
isc=str2double(get(hObject,'String'));
setappdata(0,'isc',isc);


% --- Executes during object creation, after setting all
properties.
function isc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to isc (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 psc_Callback(hObject, eventdata, handles)
% hObject    handle to psc (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 psc as text
%        str2double(get(hObject,'String')) returns contents of
psc as a double
psc=str2double(get(hObject,'String'));
setappdata(0,'psc',psc);


% --- Executes during object creation, after setting all
properties.
function psc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to psc (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 voc_Callback(hObject, eventdata, handles)
% hObject    handle to voc (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 voc as text
%        str2double(get(hObject,'String')) returns contents of
voc as a double
voc=str2double(get(hObject,'String'));
setappdata(0,'voc',voc);


% --- Executes during object creation, after setting all
properties.
function voc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to voc (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 ioc_Callback(hObject, eventdata, handles)
% hObject    handle to ioc (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 ioc as text
%        str2double(get(hObject,'String')) returns contents of
ioc as a double
ioc=str2double(get(hObject,'String'));
setappdata(0,'ioc',ioc);


% --- Executes during object creation, after setting all
properties.
function ioc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ioc (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 poc_Callback(hObject, eventdata, handles)
% hObject    handle to poc (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 poc as text
%        str2double(get(hObject,'String')) returns contents of
poc as a double
poc=str2double(get(hObject,'String'));
setappdata(0,'poc',poc);
```

```
% --- Executes during object creation, after setting all
properties.
function poc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to poc (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)
% 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)
va=str2num(get(handles.va,'string'));
vp=str2num(get(handles.vp,'string'));
vs=str2num(get(handles.vs,'string'));
voc=str2num(get(handles.voc,'string'));
vsc=str2num(get(handles.vsc,'string'));
ioc=str2num(get(handles.ioc,'string'));
isc=str2num(get(handles.isc,'string'));
poc=str2num(get(handles.poc,'string'));
psc=str2num(get(handles.psc,'string'));
pf=str2num(get(handles.pf,'string'));

c=handles.listbox1;
l=handles.listbox2;
```

```
if((isnan(va))||(isnan(vp))||(isnan(vs))||(isnan(voc))||(isnan(v
sc))||(isnan(ioc))||(isnan(isc))||(isnan(poc))||(isnan(psc))||(i
snan(pf)))
     msgbox('ENTER NUMERICAL VALUES');
end

if(va<=0||vp<=0||vs<=0||voc<=0||vsc<=0||ioc<=0||isc<=0||poc<=0||
psc<=0||pf<=0)
    errordlg('ERROR: ENTER VALID POSITIVE VALUES');
end

if(pf>1)
    msgbox('Power factor cannot exceed 1');
end

vp_line=vp;
vs_line=vs;
voc_line=voc;
vsc_line=vsc;
ioc_line=ioc;
isc_line=isc;
poc_3phase=poc;
psc_3phase=psc;

amps =zeros(1,10);
VR =zeros(1,10);
I=zeros(1,10);
amps1=zeros(1,10);
Eff=zeros(1,10);

if(c==1)  %yy connection
    a=vp_line/vs_line;
end
if(c==2)  %yd connection
    a=vp_line/(vs_line*sqrt(3));
end
 if(c==3)  %dd connection
     a=vp_line/vs_line;
 end
 if(c==4)  %dy connection
    a=sqrt(3)*vp_line/vs_line;
 end
 if(c==5) %single phase
     a=vp_line/vs_line;
 end

if(a>=1)
%     meaning sc values obtained are referred to primary side

% for yy connection
 if(c==1)
    vsc_phase=vsc_line/sqrt(3);
    isc_phase=isc_line;
    psc_per_phase=psc_3phase/3;

    voc_phase=voc_line/sqrt(3);
    ioc_phase=ioc_line;
    poc_per_phase=poc_3phase/3;
    theta_oc=-acos(poc_per_phase/(voc_phase*ioc_phase));
    yeq=ioc_phase/voc_phase;
    rc_secondary=1/(yeq*cos(theta_oc));
    xm_secondary=-1/(yeq*sin(theta_oc));
    rc_primary=a^2*rc_secondary;
    xm_primary=a^2*xm_secondary;


    theta_sc=acos(psc_per_phase/(vsc_phase*isc_phase));
    zeq=vsc_phase/isc_phase;
    req=zeq*cos(theta_sc);
    xeq=zeq*sin(theta_sc);
    req_secondary=req./a^2;
    xeq_secondary=xeq./a^2;

    ip_rated=va/(3*vp_line); %primary side
    amps=0:0.01:ip_rated;
    re=pf;
    im=sin(acos(pf));
    if(l==1) %lead
```

```
        I=amps.*(re+j*im);
    end
    if(l==3) %lag
        I=amps.*(re-j*im);
    end
    if(l==2) %unity
        I=amps.*1;
    end

    vp_phase=vp_line/sqrt(3); %primary phase voltage
    avs_phase=vp_phase-(req.*I+j.*xeq.*I);
    VR=((vp_phase-abs(avs_phase))./abs(avs_phase)).* 100;

    is_rated=va/(vs_line*3);
    amps1=0:0.01:is_rated;
    P_cu=3.*((amps1).^2).*req_secondary;
    P_core=3.*(vp_phase./a).^2/rc_secondary;
    P_out=sqrt(3).*vs_line.*amps1.*pf;
    P_in=P_cu+P_core+P_out;
    Eff=(P_out./P_in).*100;

    rc=round(rc_primary,2);
    xm=round(xm_primary,2);
    req=round(req,3);
    xeq=round(xeq,3);

 end

if(c==2) %YD
    vsc_phase=vsc_line/sqrt(3);
    isc_phase=isc_line;
    psc_per_phase=psc_3phase/3;
    theta_sc=acos(psc_per_phase/(vsc_phase*isc_phase));
    zeq=vsc_phase/isc_phase;
    req=zeq*cos(theta_sc);
    xeq=zeq*sin(theta_sc);
    req_secondary=req./a^2;
    xeq_secondary=xeq./a^2;
    voc_phase=voc_line;
    ioc_phase=ioc_line/sqrt(3);
    poc_per_phase=poc_3phase/3;
    theta_oc=-acos(poc_per_phase/(voc_phase*ioc_phase));
    yeq=ioc_phase/voc_phase;
    rc_secondary=1/(yeq*cos(theta_oc));
    xm_secondary=-1/(yeq*sin(theta_oc));
    rc_primary=a^2*rc_secondary;
    xm_primary=a^2*xm_secondary;


    ip_rated=va/(3*vp_line);
    amps=0:0.01:ip_rated;
    re=pf;
    im=sin(acos(pf));
    if(l==1)%LEAD
        I=amps.*(re+j*im);
    end
    if(l==3)  %LAG
        I=amps.*(re-j*im);
    end
    if(l==2) %UNITY
        I=amps.*1;
    end

    vp_phase=vp_line/sqrt(3); %primary phase voltage
    avs_phase=vp_phase-(req.*I+j.*xeq.*I); % Calculate secondary
phase voltage referred to the primary side for each current and
power factor.
    VR=((vp_phase-abs(avs_phase))./abs(avs_phase)).* 100;


    is_rated=va/(vs_line*3);
    amps1=0:0.01:is_rated;
    P_cu=3.*((amps1).^2).*req_secondary;
    P_core=3.*(vp_phase./a).^2/rc_secondary;
    P_out=sqrt(3).*vs_line.*amps1.*pf;
    P_in=P_cu+P_core+P_out;
    Eff=(P_out./P_in).*100;
    rc=round(rc_primary,2);
    xm=round(xm_primary,2);
    req=round(req,3);
    xeq=round(xeq,3);

end
if(c==3) %dd
    vsc_phase=vsc_line;
    isc_phase=isc_line/sqrt(3);
    psc_per_phase=psc_3phase/3;
    theta_sc=acos(psc_per_phase/(vsc_phase*isc_phase));
    zeq=vsc_phase/isc_phase;
    req=zeq*cos(theta_sc);
    xeq=zeq*sin(theta_sc);
    req_secondary=req./a^2;
    xeq_secondary=xeq./a^2;

    voc_phase=voc_line;
    ioc_phase=ioc_line/sqrt(3);
    poc_per_phase=poc_3phase/3;
    theta_oc=-acos(poc_per_phase/(voc_phase*ioc_phase));
    yeq=ioc_phase/voc_phase;
    rc_secondary=1/(yeq*cos(theta_oc));
    xm_secondary=-1/(yeq*sin(theta_oc));
    rc_primary=a^2*rc_secondary;
    xm_primary=a^2*xm_secondary;



    ip_rated=va/(3*vp_line);
    amps=0:0.01:ip_rated;
    re=pf;
    im=sin(acos(pf));
    if(l==1)  %lead
        I=amps.*(re+j*im);
    end
    if(l==3) %lag
        I=amps.*(re-j*im);
    end
    if(l==2) %unity
        I=amps.*1;
    end

    vp_phase=vp_line; %primary phase voltage
    avs_phase=vp_phase-(req.*I+j.*xeq.*I);
    VR=((vp_phase-abs(avs_phase))./abs(avs_phase)).* 100;

is_rated=va/(vs_line*3);
    amps1=0:0.01:is_rated;
    P_cu=3.*((amps1).^2).*req_secondary;
    P_core=3.*(vp_phase./a).^2/rc_secondary;
    P_out=sqrt(3).*vs_line.*amps1.*pf;
    P_in=P_cu+P_core+P_out;
    Eff=(P_out./P_in).*100;

    rc=round(rc_primary,2);
    xm=round(xm_primary,2);
    req=round(req,3);
    xeq=round(xeq,3);

end



if(c==4) %dy
    vsc_phase=vsc_line;
    isc_phase=isc_line/sqrt(3);
    psc_per_phase=psc_3phase/3;
    theta_sc=acos(psc_per_phase/(vsc_phase*isc_phase));
    zeq=vsc_phase/isc_phase;
    req=zeq*cos(theta_sc);
    xeq=zeq*sin(theta_sc);
    req_secondary=req./a^2;
    xeq_secondary=xeq./a^2;

    voc_phase=voc_line/sqrt(3);
    ioc_phase=ioc_line;
    poc_per_phase=poc_3phase/3;
    theta_oc=-acos(poc_per_phase/(voc_phase*ioc_phase));
    yeq=ioc_phase/voc_phase;
    rc_secondary=1/(yeq*cos(theta_oc));
    xm_secondary=-1/(yeq*sin(theta_oc));
    rc_primary=a^2*rc_secondary;
    xm_primary=a^2*xm_secondary;

    ip_rated=va/(3*vp_line);
    amps=0:0.01:ip_rated;
    re=pf;
    im=sin(acos(pf));
     if(l==1)  %lead
        I=amps.*(re+j*im);
    end
    if(l==3) %lag
        I=amps.*(re-j*im);
    end
    if(l==2) %unity
        I=amps.*1;
    end

    vp_phase=vp_line; %primary phase voltage
    avs_phase=vp_phase-(req.*I+j.*xeq.*I);
    VR=((vp_phase-abs(avs_phase))./abs(avs_phase)).* 100;

    is_rated=va/(vs_line*3);
    amps1=0:0.01:is_rated;
    P_cu=3.*((amps1).^2).*req_secondary;
    P_core=3.*(vp_phase./a).^2/rc_secondary;
    P_out=sqrt(3).*vs_line.*amps1.*pf;
    P_in=P_cu+P_core+P_out;
    Eff=(P_out./P_in).*100;

    rc=round(rc_primary,2);
    xm=round(xm_primary,2);
    req=round(req,3);
    xeq=round(xeq,3);

end
```

```
if(c==5)
    theta_oc=-acos(poc/(voc*ioc));
    yeq=ioc/voc;
    rc_secondary=1/(yeq*cos(theta_oc));
    xm_secondary=-1/(yeq*sin(theta_oc));
    rc_primary=a^2*rc_secondary;
    xm_primary=a^2*xm_secondary;


    theta_sc=acos(psc/(vsc*isc));
    zeq=vsc/isc;
    req=zeq*cos(theta_sc);
    xeq=zeq*sin(theta_sc);
    req_secondary=req./a^2;
    xeq_secondary=xeq./a^2;

    ip_rated=va/(vp); %primary side
    amps=0:0.01:ip_rated;
    re=pf;
    im=sin(acos(pf));
    if(l==1) %lead
        I=amps.*(re+j*im);
    end
    if(l==3) %lag
        I=amps.*(re-j*im);
    end
    if(l==2) %unity
        I=amps.*1;
    end


    avs=vp-(req.*I+j.*xeq.*I);
    VR=((vp-abs(avs))./abs(avs)).* 100;

    is_rated=va/(vs);
    amps1=0:0.01:is_rated;
    P_cu=((amps1).^2).*req_secondary;
    P_core=(vp./a).^2/rc_secondary;
    P_out=vs.*amps1.*pf;
    P_in=P_cu+P_core+P_out;
    Eff=(P_out./P_in).*100;

    rc=round(rc_primary,2);
    xm=round(xm_primary,2);
    req=round(req,3);
    xeq=round(xeq,3);
end
end

if(a<1)
%     meaning sc values obtained are referred to secondary side
and oc test
%     values are obtained referred to primary side
if(c==1) %yy
    vsc_phase=vsc_line/sqrt(3);
    isc_phase=isc_line;
    psc_per_phase=psc_3phase/3;

    theta_sc=acos(psc_per_phase/(vsc_phase*isc_phase));
    zeq=vsc_phase/isc_phase;
    req=zeq*cos(theta_sc);
    xeq=zeq*sin(theta_sc);

    voc_phase=voc_line/sqrt(3);
    ioc_phase=ioc_line;
    poc_per_phase=poc_3phase/3;
    theta_oc=-acos(poc_per_phase/(voc_phase*ioc_phase));
    yeq=ioc_phase/voc_phase;
    rc_primary=1/(yeq*cos(theta_oc));
    xm_primary=-1/(yeq*sin(theta_oc));
    rc_secondary=rc_primary./a^2;
    xm_secondary=xm_primary./a^2;

    is_rated=va/(3*vs_line);
    amps=0:0.01:is_rated;
    re=pf;
    im=sin(acos(pf));
```

```
     if(l==1)  %lead
        I=amps.*(re+j*im);
    end
    if(l==3) %lag
        I=amps.*(re-j*im);
    end
    if(l==2) %unity
        I=amps.*1;
    end

    vsp=vs_line/sqrt(3); %secondary phase voltage
    vppbya=vsp+(req.*I+j.*xeq.*I);
    VR=((vppbya-abs(vsp))./abs(vsp)).* 100;

    vp_phase=vp_line/sqrt(3);
    amps1=0:0.01:is_rated;
    P_cu=3.*((amps1).^2).*req;
    P_core=3.*(vp_phase./a).^2/rc_secondary;
    P_out=sqrt(3).*vs_line.*amps1.*pf;
    P_in=P_cu+P_core+P_out;
    Eff=(P_out./P_in).*100;

    rc=round(rc_secondary,2);
    xm=round(xm_secondary,2);
    req=round(req,3);
    xeq=round(xeq,3);


end

if(c==2) %yd
    vsc_phase=vsc_line;
    isc_phase=isc_line/sqrt(3);
    psc_per_phase=psc_3phase/3;
    theta_sc=acos(psc_per_phase/(vsc_phase*isc_phase));
    zeq=vsc_phase/isc_phase;
    req=zeq*cos(theta_sc);
    xeq=zeq*sin(theta_sc);
```

```
voc_phase=voc_line/sqrt(3);
ioc_phase=ioc_line;
poc_per_phase=poc_3phase/3;
theta_oc=-acos(poc_per_phase/(voc_phase*ioc_phase));
yeq=ioc_phase/voc_phase;
rc_primary=1/(yeq*cos(theta_oc));
xm_primary=-1/(yeq*sin(theta_oc));
rc_secondary=rc_primary./a^2;
xm_secondary=xm_primary./a^2;

is_rated=va/(3*vs_line);
amps=0:0.01:is_rated;
re=pf;
im=sin(acos(pf));
if(l==1)  %lead
    I=amps.*(re+j*im);
end
if(l==3) %lag
    I=amps.*(re-j*im);
end
if(l==2) %unity
    I=amps.*1;
end

vsp=vs_line; %secondary phase voltage
vppbya=vsp+(req.*I+j.*xeq.*I);
VR=((vppbya-abs(vsp))./abs(vsp)).* 100;

vp_phase=vp_line/sqrt(3);
is_rated=va/(vs_line*3);
amps1=0:0.01:is_rated;
P_cu=3.*((amps1).^2).*req;
P_core=3.*(vp_phase./a).^2/rc_secondary;
P_out=sqrt(3).*vs_line.*amps1.*pf;
P_in=P_cu+P_core+P_out;
Eff=(P_out./P_in).*100;

rc=round(rc_secondary,2);
xm=round(xm_secondary,2);
```

```
    req=round(req,3);
    xeq=round(xeq,3);

end
if(c==3) %dd
    vsc_phase=vsc_line;
    isc_phase=isc_line/sqrt(3);
    psc_per_phase=psc_3phase/3;
    theta_sc=acos(psc_per_phase/(vsc_phase*isc_phase));
    zeq=vsc_phase/isc_phase;
    req=zeq*cos(theta_sc);
    xeq=zeq*sin(theta_sc);

    voc_phase=voc_line;
    ioc_phase=ioc_line/sqrt(3);
    poc_per_phase=poc_3phase/3;
    theta_oc=-acos(poc_per_phase/(voc_phase*ioc_phase));
    yeq=ioc_phase/voc_phase;
    rc_primary=1/(yeq*cos(theta_oc));
    xm_primary=-1/(yeq*sin(theta_oc));
    rc_secondary=rc_primary./a^2;
    xm_secondary=xm_primary./a^2;

    is_rated=va/(3*vs_line);
    amps=0:0.01:is_rated;
    re=pf;
    im=sin(acos(pf));
     if(l==1)  %lead
        I=amps.*(re+j*im);
    end
    if(l==3) %lag
        I=amps.*(re-j*im);
    end
    if(l==2) %unity
        I=amps.*1;
    end

   vsp=vs_line; %secondary phase voltage
    vppbya=vsp+(req.*I+j.*xeq.*I);
    VR=((vppbya-abs(vsp))./abs(vsp)).* 100;

    vp_phase=vp_line;
    is_rated=va/(vs_line*3);
    amps1=0:0.01:is_rated;
    P_cu=3.*((amps1).^2).*req;
    P_core=3.*(vp_phase./a).^2/rc_secondary;
    P_out=sqrt(3).*vs_line.*amps1.*pf;
    P_in=P_cu+P_core+P_out;
    Eff=(P_out./P_in).*100;

    rc=round(rc_secondary,2);
    xm=round(xm_secondary,2);
    req=round(req,3);
    xeq=round(xeq,3);

end

if(c==4)     %dy

    vsc_phase=vsc_line/sqrt(3);
    isc_phase=isc_line;
    psc_per_phase=psc_3phase/3;
    theta_sc=acos(psc_per_phase/(vsc_phase*isc_phase));
    zeq=vsc_phase/isc_phase;
    req=zeq*cos(theta_sc);
    xeq=zeq*sin(theta_sc);

    voc_phase=voc_line;
    ioc_phase=ioc_line/sqrt(3);
    poc_per_phase=poc_3phase/3;
    theta_oc=-acos(poc_per_phase/(voc_phase*ioc_phase));
    yeq=ioc_phase/voc_phase;
    rc_primary=1/(yeq*cos(theta_oc));
    xm_primary=-1/(yeq*sin(theta_oc));
    rc_secondary=rc_primary./a^2;
    xm_secondary=xm_primary./a^2;

    is_rated=va/(3*vs_line);
```

```
        amps=0:0.01:is_rated;
        re=pf;
        im=sin(acos(pf));
         if(l==1)  %lead
            I=amps.*(re+j*im);
        end
        if(l==3) %lag
            I=amps.*(re-j*im);
        end
        if(l==2) %unity
            I=amps.*1;
        end
        vsp=vs_line/sqrt(3); %secondary phase voltage
        vppbya=vsp+(req.*I+j.*xeq.*I);
        VR=((vppbya-abs(vsp))./abs(vsp)).* 100;

        vp_phase=vp_line;
        is_rated=va/(vs_line*3);
        amps1=0:0.01:is_rated;
        P_cu=3.*((amps1).^2).*req;
        P_core=3.*(vp_phase./a).^2/rc_secondary;
        P_out=sqrt(3).*vs_line.*amps1.*pf;
        P_in=P_cu+P_core+P_out;
        Eff=(P_out./P_in).*100;

        rc=round(rc_secondary,2);
        xm=round(xm_secondary,2);
        req=round(req,3);
        xeq=round(xeq,3);

    end
    if(c==5)
        theta_sc=acos(psc/(vsc*isc));
        zeq=vsc/isc;
        req=zeq*cos(theta_sc);
        xeq=zeq*sin(theta_sc);


        theta_oc=-acos(poc/(voc*ioc));
```

```
yeq=ioc/voc;
rc_primary=1/(yeq*cos(theta_oc));
xm_primary=-1/(yeq*sin(theta_oc));
rc_secondary=rc_primary./a^2;
xm_secondary=xm_secondary./a^2;

is_rated=va/vs;
amps=0:0.01:is_rated;
re=pf;
im=sin(acos(pf));
 if(l==1)  %lead
    I=amps.*(re+j*im);
end
if(l==3) %lag
    I=amps.*(re-j*im);
end
if(l==2) %unity
    I=amps.*1;
end


vpbya=vs+(req.*I+j.*xeq.*I);
VR=((vpbya-abs(vs))./abs(vs)).* 100;


is_rated=va/(vs);
amps1=0:0.01:is_rated;
P_cu=((amps1).^2).*req;
P_core=(vp./a).^2/rc_secondary;
P_out=vs.*amps1.*pf;
P_in=P_cu+P_core+P_out;
Eff=(P_out./P_in).*100;

rc=round(rc_secondary,2);
xm=round(xm_secondary,2);
req=round(req,3);
xeq=round(xeq,3);

end
```

```
end

str2double(set(handles.REQ,'string',req));
str2double(set(handles.XEQ,'string',xeq));
str2double(set(handles.RC,'string',rc));
str2double(set(handles.XM,'string',xm));

axes(handles.axes1);
plot(amps,VR);
xlabel('percentage load current');
ylabel('voltage regulation');

axes(handles.axes2);
plot(amps1,Eff);
xlabel('percentage load current');
ylabel('Efficiency');


% --- Executes during object creation, after setting all
properties.
function axes3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
imshow('cir.PNG');


% Hint: place code in OpeningFcn to populate axes3
```