School of Engineering and Applied Sciences
Electronics and Computer Engineering (ECE)

Electric Machines

ECEN 403 / ECEN 316

Fall 2020

Project Report

Practical Transformer Simulator

Names:

Ahmed Mohamed Fouad
Ahmed Ayman Ahmed
Emil Mourad Matta
Karim Mohamed Abouaish
Sara Hossam Anwar

Submitted To: Dr. Heba Ahmed Hassan

Associate Professor Dr. Heba Ahmed Hassan
Engineer Alaa Osama Abel Fattah

ACKNOWLEDGEMENTS

This work was supported by Associate Professor Dr. Heba Ahmed Hassan and Engineer Alaa Osama Abel Fattah.

ABSTRACT

The Transformers operation is based on the theory that a varying magnetic field generated by alternating current can efficiently transfer energy by magnetic induction from one winding to another winding. On the other hand, practical transformers have some imperfections compared to its ideal model. Thus, practical transformer parameters are harder to measure. In this paper, we developed a program "Practical Transformer Simulator" that utilizes a GUI to calculate the single-phase AC practical transformer parameters and its efficiency given the open-circuit and short-circuit tests data. This project has been designed and developed by using MATLAB & SIMULINK. This program could be used in the form of simulation for laboratory session as the user will only need to perform the short-circuit and open-circuit tests practically then put the measured data into the program and the practical transformer simulator will get all the missing parameters. It can also be used as an illustrator/simulation for online distant-learning process in terms on how to analyze practically through the software. This paper has presented a simulation of AC transformer by using MATLAB & SIMULINK which it is a user-friendly programming language and easy to be learnt by new programmer. The MATLAB & SIMULINK result have been verified and compared with hand calculation in order to ensure they are correct and reliable.

Keywords: MATLAB, SIMULINK, Transformer, GUI, test, efficiency, etc.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

NOMENCLATURE

E_g: Source/Rated/Applied Voltage (E_{oc})	R_m: Resistance representing the iron core losses
R: Load resistance	X_m: Magnetizing reactance of the primary winding
P_T: Total active power dissipated in the circuit	R_L: Transformer Resistance
P_{oc}: Active Power Absorbed by the iron core (P_i)	X_L: Leakage Reactance
S_m: Apparent Power Absorbed by the iron core	Z_L: Transformer Impedance
Q_m: Reactive Power Absorbed by the iron core	I_p: Source Current (I_{oc})
P_{sc}: Active Power Absorbed by the copper windings (P_{cu})	I_o: Excitation/No-load current
E_{sc}: Short-circuit Voltage supply	I_f: current flowing in X_m
I_{sc}: Short-circuit Current	I_m: Magnetizing current
a: Turns ratio	I_1: Primary current
η: Transformer Efficiency	I_2: Secondary current
$\emptyset_m$: Mutual Flux	E_1: Primary Voltage
k: loading factor ratio	E_2: Induced Secondary Voltage
S_{fl}: transformer rated apparent power at full-load	E_s: Output Secondary Voltage

CHAPTER 1: INTRODUCTION AND LITERATURE REVIEW

1.1 Transformer

A transformer is a passive electrical device that transfer electrical energy from one electrical circuit to the other. A changing current in the primary coil of the transformer creates a varying magnetic flux, which create a changing electromotive force across the secondary coil. Electrical energy can be moved between discrete loops without a metallic (conductive) association between the two circuits.

1.2 Types of transformers

1.2.1 Ideal Transformer

Characteristics of Ideal Transformer

- **No losses in the perfect core (Its core is infinitely permeable):**
  As the permeability increase, the electromotive force required for flux establishment decreases. which means that, if permeability is high, nearly no magnetizing current (I_m) is required to produce the flux [1].
- **No losses due to the resistance of the windings.**
  It is assumed that, the resistance of the primary and the secondary winding is zero. Which means the both coils are purely inductive [1].
- **No Flux Leakage**
  In an ideal transformer, it is assumed that any flux produced from the primary coil is completely captured by the secondary coil, which means that no lose in the magnetic flux [1].

Which implies that an ideal transformer gives an output power that is actually equivalent to the input power. The efficiency of the ideal transformer is 100%. In real life, it is difficult to have such a transformer, however an ideal transformer model is only used to facilitate modeling practical transformer [1].

1.2.2 **Practical Transformer**

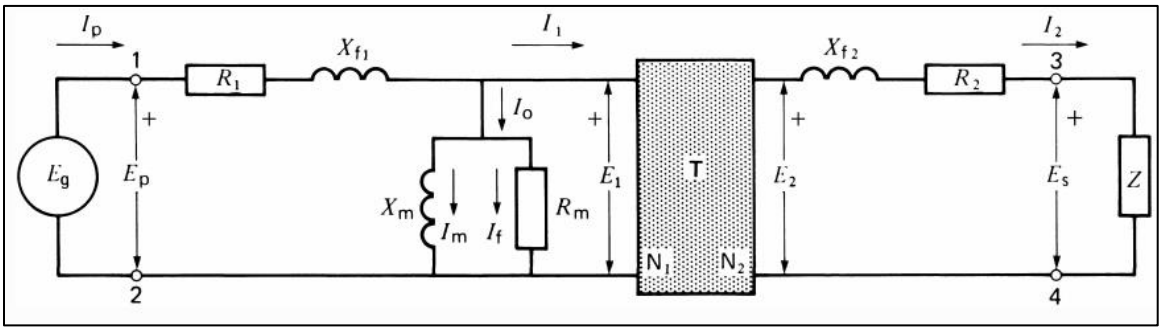

Figure 1 Equivalent Circuit of Practical Transformer [1]

Characteristics of Practical Transformer

- **Losses in iron core (Imperfect core effect)**
  By replacing the perfect core with an iron core which have a lower permeability causes an eddy current and hysteresis loss in it. These losses are being represented by two circuit elements R_m and X_m which are parallel to the ideal transformers. Where R_m represents the lose iron lose due to heat produced by the current I_f passing through R_m. Magnetizing reactance (X_m) measures the permeability of the transformer core, passing through it the I_m which is the magnetizing current needed to produce flux in the core [1].
- **Losses due to the resistance of the windings (Loose coupling effect I)**
  Since the windings consists of copper, which means there is losses due to its resistance in form of heat. The resistance of the primary windings is R1, and the resistance of the secondary winding is R2 as shown in the figure (1) [1].
- **Flux leakage (Loose coupling effect II)**
  Primary and secondary coil produces flux, the useful flux that links the both winding is called mutual flux ($ø_m$). However, the current from the primary produces some flux which is not linked with the secondary which is called flux leakage ($ø_{L1}$) and same with the secondary coil ($ø_{L2}$). These lose is represented with an inductive reactance connected series with the windings resistance respectively which are (X_{F1}) and (X_{F2}) as shown in figure (1) [1].

Active Power Losses Practical Transformer

Unlike ideal transformer, practical transformer has power losses due to its winding resistances, loose coupling and imperfect core. Active power losses have two types: fixed power losses (P_i) and variable power losses (P_{cu}). The fixed active power losses are related to the power dissipated in the iron core and it's independent of the circuit loading. Thus, $P_i = \frac{E_g^2}{R_m}$. The variable active power losses are related to the power dissipated in the transformer copper windings and it's dependent on the circuit loading. Hence, we have variable active power losses (P_{cu}) and variable active full-load power losses (P_{cufl}), where: $P_{cu} = k^2 P_{cufl}$ (k is the loading factor ratio) [1].

Efficiency of Practical Transformer

Since there are power losses, the efficiency of practical transformer is no more 100% as the ideal one. Output power of the transformer at certain loading could be represented in terms of the transformer rated apparent power at full-load (S_{fl} in KVA), the loading factor ratio (k) and the load power factor (PF) to convert from full-load to the actual load. Hence, $P_{O/P} = k * S_{Fl} * PF$. Output power of the transformer at certain loading could be represented in terms of the output power of the transformer added to the total power losses. Since, the efficiency of a machine is equal to $\frac{P_{O/P}}{P_{I/P}} * 100\%$, thus efficiency ($\eta$) can be calculated as follows [1]:

$$\eta = \frac{k \times S_{fl} \times PF}{\left(k \times S_{fl} \times PF\right) + P_i + P_{cu}} \times 100$$

$$\eta = \frac{k \times S_{fl} \times PF}{\left(k \times S_{fl} \times PF\right) + P_i + k^2 P_{cufl}} \times 100$$

1.3 Transformer Parameters

1.3.1 Open-circuit test

In the open-circuit test, the rated voltage (Eg) is applied to the primary, while the secondary is opened (ie. The transformer is at no-load). Thus, the winding resistance and leakage reactance effects are cancelled (ie. Loose coupling effect is cancelled). By this, we can calculate the parameters of the magnetizing branch, given that Pi , Io and Eg are measured by a wattmeter, ammeter and voltameter respectively. By the aforementioned approximations, we can reduce figure (1) to figure (2) as shown [2].

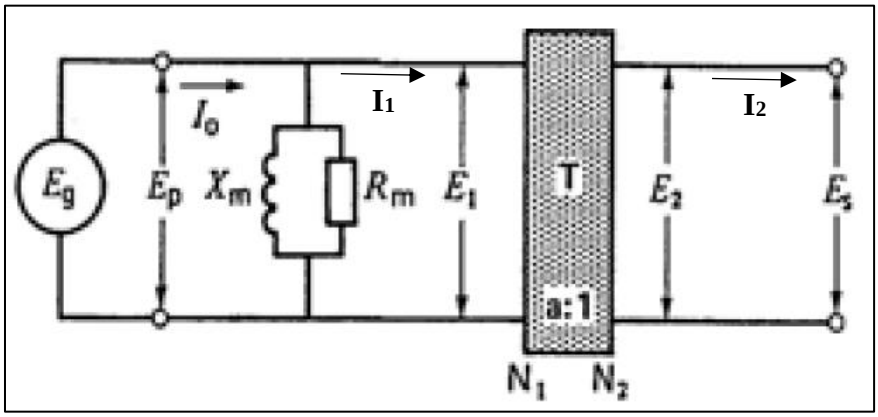

Figure 2 Approximation of the Equivalent Transformer Circuit at: No-Load [1]

Open-circuit test calculations:

Form the open circuit test, we can calculate the parameters of the magnetizing branch as follows[1]:

- Apparent power absorbed by core (S_m): $S_m = E_g I_0$, Thus, $Q_m = \sqrt{S_m^2 - P_i^2}$
- Active power absorbed by core (P):
  - There are no copper losses (P_{cu}) because there is no load.
  - Concerning the iron core losses (P_i): $P_i = \frac{E_g^2}{R_m}$, so we can calculate R_m. Since, $I_f = \frac{Eg}{R_m}$,we can also calculate it directly.
- Reactive power absorbed by core (Q_m): $Q_m = \frac{E_g^2}{X_m}$, so we can calculate X_m. Since, $I_m = \frac{Eg}{R_m}$,we can also calculate it directly.

1.3.2 **Short-circuit test**

In the open-circuit test, the rated voltage (Eg) is applied to the primary, while the secondary is short-circuited. The supply voltage required to circulate rated current through the transformer is usually very small and is of the order of a few percent of the nominal voltage and this 5% voltage is applied across primary side. The core losses are very small because applied voltage is only a small percentage of the nominal voltage and hence can be neglected (ie. Imperfect core effect is neglected). Thus, the wattmeter reading measures only the full load copper losses, the ammeter reading measures I_{sc} and the voltameter reading measures E_{sc}. By the aforementioned approximations, we can reduce figure (1) to figure (3) as shown [3].

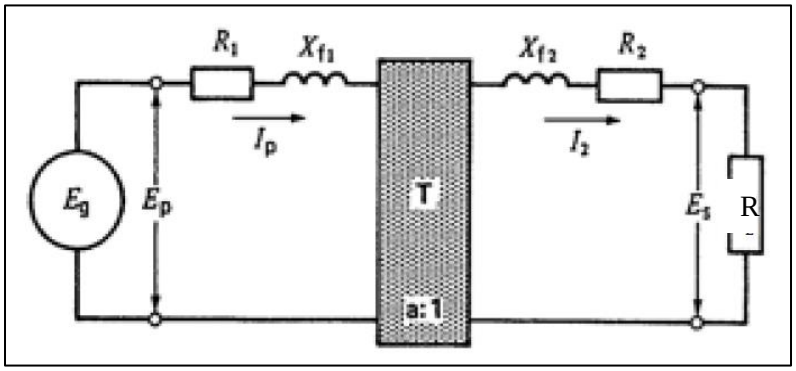

Figure 3 Approximation of the Equivalent Transformer Circuit at: Full-Load [1]

When short-circuit the secondary, R=0, so we can get the electrical equivalent circuit in figure (4).

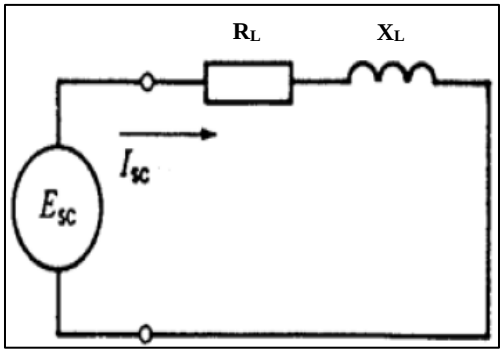

Figure 4 Short-circuit test electrical equivalent circuit [1]

Short-circuit test calculations:

Form the open circuit test, we can calculate the transformer impedances (referred to the primary side) as follows [1]:

- Total transformer impedance (Z_L): $Z_L = \frac{E_{sc}}{I_{sc}}$
- Total transformer resistance (R_L): $R_L = \frac{P_{sc}}{{I_{sc}}^2}$
- Total transformer leakage reactance (X_L): $X_L = \sqrt{z_L^2 - R_L^2}$

CHAPTER 2: METHODOLOGY

Since practical transformer parameters are hard to measure. We developed a program called "Practical Transformer Simulator" that utilizes a Graphical User Interface (GUI) to calculate the practical transformer parameters and its efficiency. This program is developed by using MATLAB & SIMULINK. The Practical Transformer Simulator could be used in laboratory session as the user will only need to perform the short-circuit and open-circuit tests practically then put the measured data into the GUI and the Practical Transformer Simulator will get all the missing parameters. It can also be used to check hand calculations. We used MATLAB & SIMULINK because they are user-friendly and easy to be learnt by beginners. The MATLAB & SIMULINK result have been verified and compared with hand calculation in order to ensure they are correct and reliable.

The flowchart in figure (5) illustrates the programs logic and sequence of calculations as follows:

- Firstly, some specifications are given to the software as input such as (E_{sc}, P_{sc}, I_{sc}, E_g, R_{Load}, a, E_{oc}, P_{oc}, I_{oc}). These specifications are used to compute the missing data of the transformer circuit.
- Secondly, an algorithm was created for the software that will use the mathematical equations to automatically calculate the values of different parameters of the electric circuit of the power transformer which are: (X_m, R_m, X_L, R_L).
- Thirdly, the SIMULINK-based software was designed to read the required input (X_m, R_m, X_L, R_L) and then automatically calculate and display the rest of the parameters of the equivalent circuit of transformer (I_p ,I_2 ,E_1 ,E_2 ,P_t , η) . And it finally shows a graph of the electric circuit of the power transformer. The development of the MATLAB software for designing and analysis the transformers consists of the following: MATLAB supports developing applications with graphical user interface features. MATLAB includes GUIDE (GUI development environment) for graphically designing GUIs. It also has tightly integrated graph-plotting features.

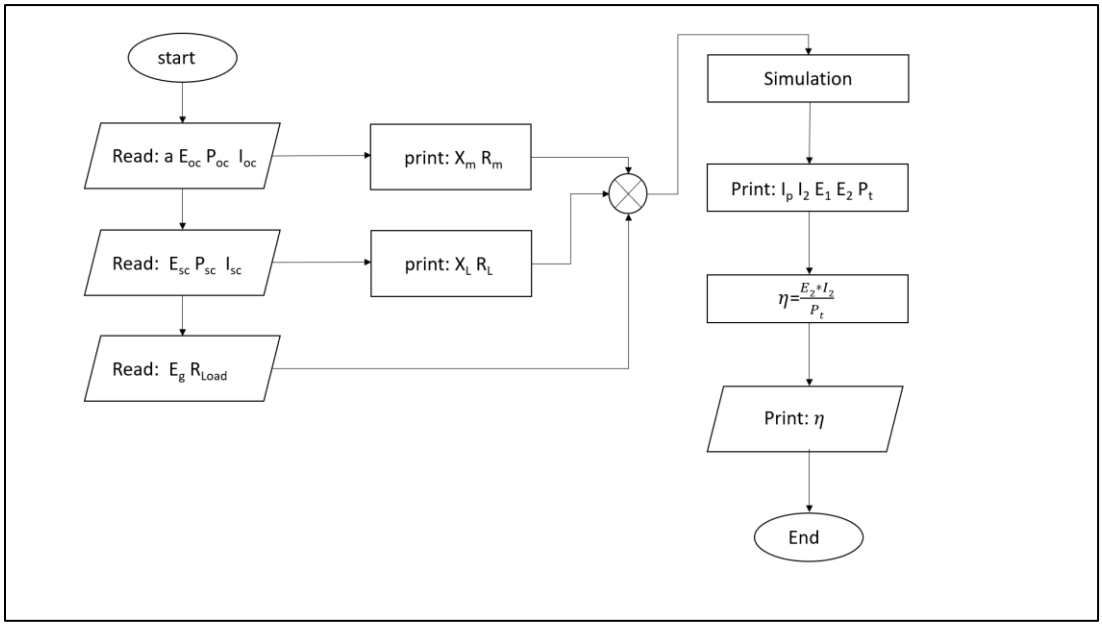

Figure 5 Program Flowchart

CHAPTER 3: RESULTS AND DISCUSSION

Part of the calculations are made through equations and the other part is measured from the transformer equivalent circuit simulation by SIMULINK. The following figure shows that equivalent circuit.

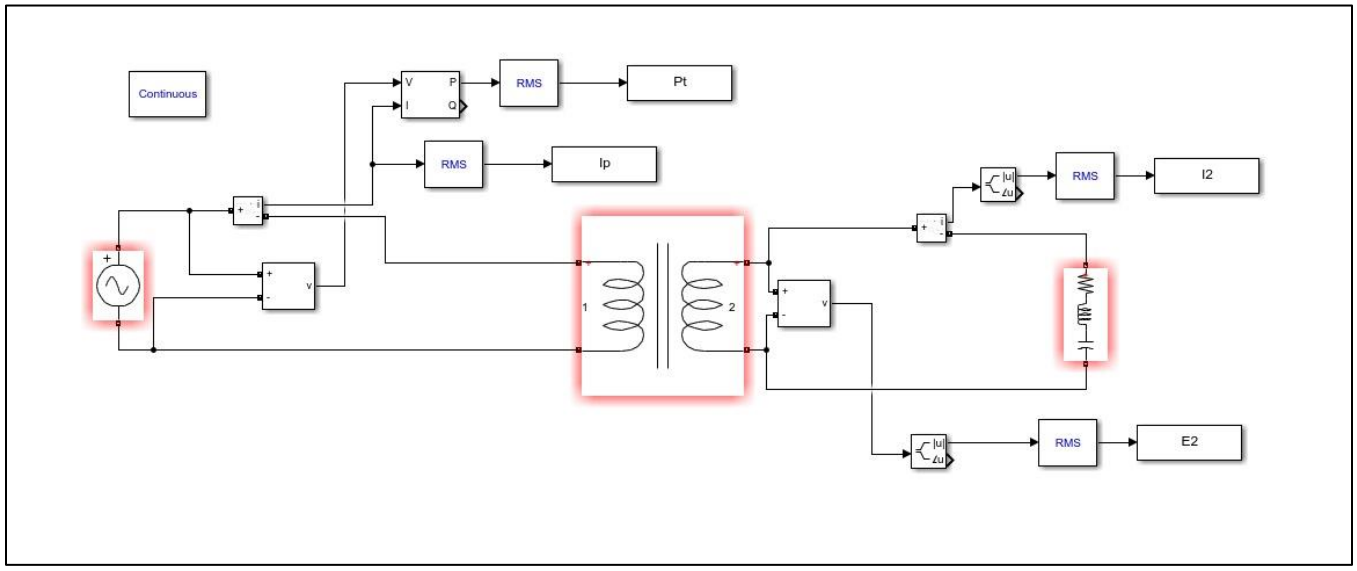

Figure 6 transformer equivalent circuit simulation by SIMULINK

To make the "Practical Transformer Simulator" more user friendly, MATLAB GUIDE feature (GUI development environment) is used for designing the GUI in figure (7).

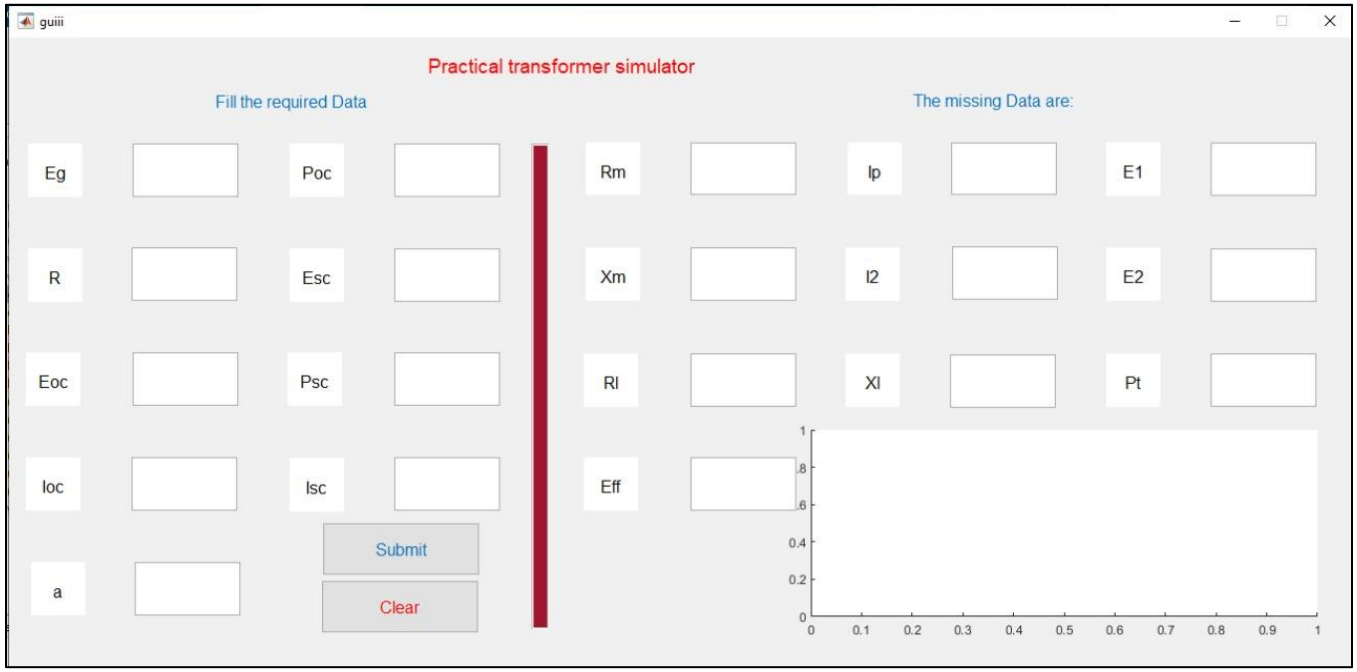

Figure 7 Practical Transformer Simulator User Interface

Example: The following example was used to test the Practical Transformer Simulator output:

A transformer of turns ratio 2:1, and its rated voltage is 3KV. From the open-circuit test, the iron losses were found to be 20W and the primary current was measured 0.7A at 200 V supply. A short-circuit test was performed at 15 V supply, the variable losses were found to be 50W and the short-circuit current was measured 10A.

The following results were obtained:

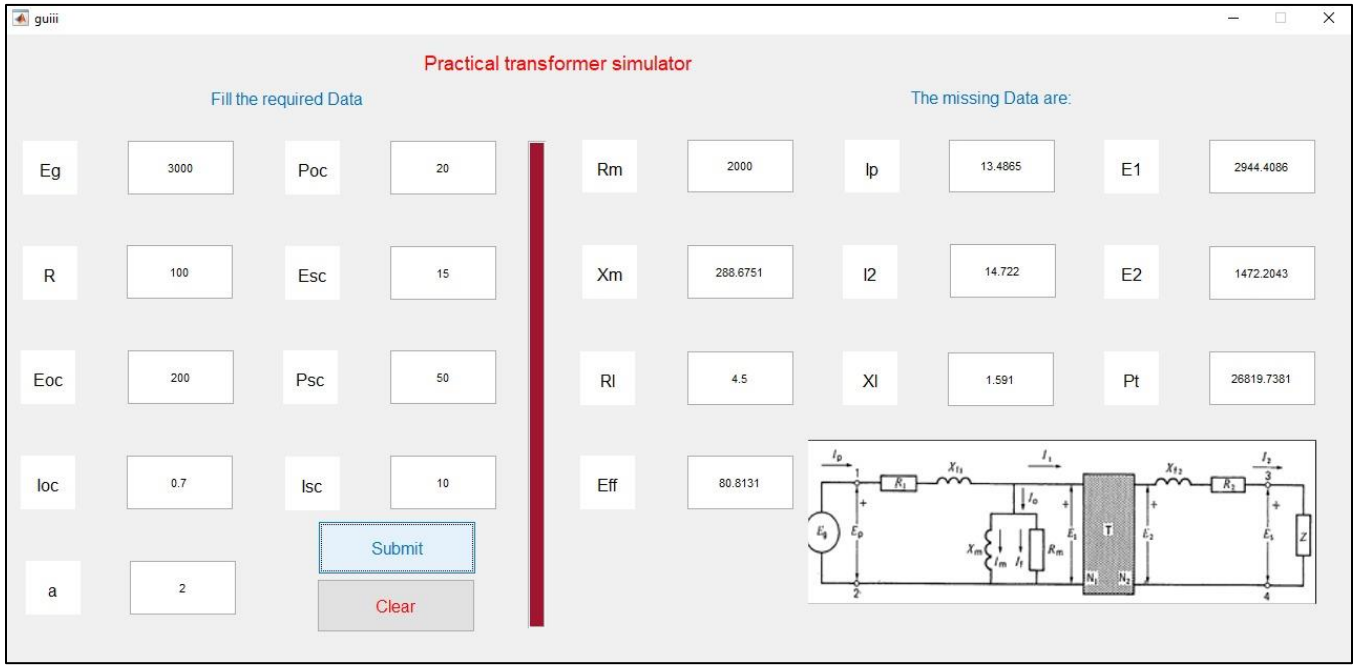

Figure 8 Practical results

CHAPTER 4: CONCLUSION AND FUTURE WORK

The “Practical Transformer Simulator” enables users to calculate the transformer parameters after performing the open-circuit and short-circuit tests practically in the lab. It can also be used to check hand calculations. In this paper, we have discussed the transformer parameters and tests, and have illustrated the process of the simulator program operation. The scope of this program calculates the circuit parameters of single-phase AC transformers only if the open-circuit and short-circuit tests data are made available. Further work is needed to perform the open-circuit and short-circuit tests on the simulation. Also, further work is needed to enhance the GUI to present graphical data and be more user friendly. Finally, this paper didn’t include comparison between the field results and the simulation results. Hence, we need to conduct the open-circuit and short-circuit tests practically and present such comparison.

REFERENCES

[1] T. Wildi, “Electrical Machines, Drives and Power Systems”, Pearson Education Inc., Pearson Prentice Hall, Sixth International Edition, USA, 2006.

[2] Administrator, Says, V., Vrutik, Says, A., & Ray, A. (2017, December 24). Open Circuit and Short Circuit Test on Transformer. Retrieved January 09, 2021, from: https://www.electronicshub.org/open-circuit-and-short-circuit-test-on-transformer/

[3] Garin, A. N., & Paluev, K. K. (1936). *Transformer Circuit Impedance Calculation*. Transactions of the American Institute of Electrical Engineers, 55(6), 717–730. doi:10.1109/t-aiee.1936.5057336

[4] Patil, N., PG Student, & Patil, J., Prof. (2015). *Transformer Testing and Analysis using MATLAB/Simuink* (02nd ed., Vol. 2, Working paper). IJIRST –International Journal for Innovative Research in Science & Technology. doi:ISSN (online): 2349-6010

Appendix A

```
function varargout = guiii(varargin)

% GUIII MATLAB code for guiii.fig

%      GUIII, by itself, creates a new GUIII or raises the existing

%      singleton*.

%      H = GUIII returns the handle to a new GUIII or the handle to

%      the existing singleton*.

%      GUIII('CALLBACK',hObject,eventData,handles,...) calls the local

%      function named CALLBACK in GUIII.M with the given input arguments.

%      GUIII('Property','Value',...) creates a new GUIII or raises the

%      existing singleton*.  Starting from the left, property value pairs are

%      applied to the GUI before guiii_OpeningFcn gets called.  An

%      unrecognized property name or invalid value makes property application

%      stop.  All inputs are passed to guiii_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 guiii

% Last Modified by GUIDE v2.5 09-Jan-2021 15:47:09

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;

gui_State = struct('gui_Name',       mfilename, ...

                   'gui_Singleton',  gui_Singleton, ...

                   'gui_OpeningFcn', @guiii_OpeningFcn, ...

                   'gui_OutputFcn',  @guiii_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 guiii is made visible.
function guiii_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 guiii (see VARARGIN)


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

```
% Update handles structure
guidata(hObject, handles);


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


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



% --- 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)

Eg= str2double(get(handles.edit67,'string'));

R= str2double(get(handles.edit68,'string'));

Eoc= str2double(get(handles.edit69,'string'));

Ioc= str2double(get(handles.edit70,'string'));

Poc= str2double(get(handles.edit71,'string'));

Esc= str2double(get(handles.edit72,'string'));

Psc= str2double(get(handles.edit73,'string'));

Isc= str2double(get(handles.edit74,'string'));

a= str2double(get(handles.edit75,'string'));
```

```
assignin('base','Eg',Eg);

assignin('base','R',R);

assignin('base','Eoc',Eoc);

assignin('base','Ioc',Ioc);

assignin('base','Poc',Poc);

assignin('base','Esc',Esc);

assignin('base','Psc',Psc);

assignin('base','Isc',Isc);

assignin('base','a',a);



Rm=(Eoc*Eoc)/Poc;

Qoc=sqrt(Eoc*Eoc*Ioc*Ioc-Poc*Poc);

Xm=(Eoc*Eoc)/Qoc;

Rl=(Esc*Esc)/Psc;

Qsc=sqrt(Esc*Esc*Isc*Isc-Psc*Psc);

Xl=(Esc*Esc)/Qsc;
```

```
Lm=Xm/(2*pi*50);

Ll=Xl/(2*pi*50);



assignin('base','Rm',Rm);

assignin('base','Xm',Xm);

assignin('base','Rl',Rl);

assignin('base','Xl',Xl);

assignin('base','Lm',Lm);

assignin('base','Ll',Ll);



sim('MachinesOC1');

set(handles.edit76,'string',num2str(Rm));

set(handles.edit77,'string',num2str(Xm));

set(handles.edit78,'string',num2str(Rl));

set(handles.edit79,'string',num2str(Xl));

set(handles.edit80,'string',num2str(max(Pt(50,:))));
```

```
set(handles.edit81,'string',num2str(max(Ip(50,:))));

set(handles.edit82,'string',num2str(max(I2(50,:))));

set(handles.edit84,'string',num2str(max(E2(50,:))));

E1=max(E2(50,:))*a;

xx=max(E2(50,:));

zz=max(I2(50,:));

zx= max(Pt(50,:));

set(handles.edit83,'string',num2str(E1));

Efficiency=((xx*zz)/zx)*100;

assignin('base','Efficiency',Efficiency);

set(handles.edit85,'string',num2str(Efficiency));

imshow('2.png')



function edit67_Callback(hObject, eventdata, handles)

% hObject    handle to edit67 (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 edit67 as text
%        str2double(get(hObject,'String')) returns contents of edit67 as a double


% --- Executes during object creation, after setting all properties.
function edit67_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit67 (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 edit68_CreateFcn(hObject, eventdata, handles)

% hObject    handle to edit68 (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 edit69_CreateFcn(hObject, eventdata, handles)

% hObject    handle to edit69 (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 edit70_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit70 (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 edit71_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit71 (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 edit72_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit72 (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 edit73_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit73 (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 edit74_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit74 (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 edit75_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit75 (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 edit76_CreateFcn(hObject, eventdata, handles)

% hObject    handle to edit76 (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 edit77_CreateFcn(hObject, eventdata, handles)

% hObject    handle to edit77 (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 edit78_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit78 (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 edit79_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit79 (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 edit80_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit80 (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 edit81_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit81 (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 during object creation, after setting all properties.
function edit82_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit82 (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 edit83_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit83 (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 edit84_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit84 (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 edit85_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit85 (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 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)
set(handles.edit67,'string',num2str(0));
set(handles.edit68,'string',num2str(0));
set(handles.edit69,'string',num2str(0));
set(handles.edit70,'string',num2str(0));
set(handles.edit71,'string',num2str(0));
set(handles.edit72,'string',num2str(0));
```

```
set(handles.edit73,'string',num2str(0));

set(handles.edit74,'string',num2str(0));

set(handles.edit75,'string',num2str(0));

set(handles.edit76,'string',num2str(0));

set(handles.edit77,'string',num2str(0));

set(handles.edit78,'string',num2str(0));

set(handles.edit79,'string',num2str(0));

set(handles.edit80,'string',num2str(0));

set(handles.edit81,'string',num2str(0));

set(handles.edit82,'string',num2str(0));

set(handles.edit83,'string',num2str(0));

set(handles.edit84,'string',num2str(0));

set(handles.edit85,'string',num2str(0));

clear

clc
```