POWER FLOW ANALYSIS SOFTWARE USING MATLAB

MOHD SHAHIMI BIN MOHAMAD ISA

UNIVERSITY MALAYSIA PAHANG

ABSTRACT

Power flow analysis is the backbone of power system analysis and design. They are necessary for planning, operation, economic scheduling and exchange of power between utilities. The principal information of power flow analysis is to find the magnitude and phase angle of voltage at each bus and the real and reactive power flowing in each transmission lines. Power flow analysis is an importance tool involving numerical analysis applied to a power system. In this analysis, iterative techniques are used due to there no known analytical method to solve the problem. To finish this analysis there are methods of mathematical calculations which consist plenty of step depend on the size of system. This process is difficult and takes a lot of times to perform by hand. The objective of this project is to develop a toolbox for power flow analysis that will help the analysis become easier. Power flow analysis software package develops by the author use MATLAB programming and MATLAB GUI. Data visualization and GUI design in MATLAB are based on the Handle Graphics System in which the objects organized in a Graphics Object Hierarchy can be manipulated by various high and low level commands. This software provides all three methods that commonly used, Newton Raphson method, Gauss-Seidel method and Fast Decoupled method in solving the power flow or load flow problem.

CHAPTER 1

INTRODUCTION

1.1 Overview

Power flow analysis is the backbone of power system analysis and design. They are necessary for planning, operation, economic scheduling and exchange of power between utilities. Power flow analysis is required for many other analyses such as transient stability, optimal power flow and contingency studies. The principal information of power flow analysis is to find the magnitude and phase angle of voltage at each bus and the real and reactive power flowing in each transmission lines.

Power flow analysis is an importance tool involving numerical analysis applied to a power system. In this analysis, iterative techniques are used due to there no known analytical method to solve the problem. This resulted nonlinear set of equations or called power flow equations are generated. To finish this analysis there are methods of mathematical calculations which consist plenty of step depend on the size of system. This process is difficult and takes much time to perform by hand. By develop a toolbox for power flow analysis surely will help the analysis become easier.

Power flow analysis software can help users to calculate the power flow problem. Over the past decade, a few versions of educational software packages using

advanced programming languages, such as C, C++, Pascal, or FORTRAN have been developed for power engineering curriculums. These choose an integrated study platform with support of database and GUI functions.

Power flow analysis software develops by the author use MATLAB software. MATLAB as a high-performance language for technical computation integrates calculation, visualization and programming in an easy-to-use environment, thus becomes a standard instructional tool for introductory and advanced courses in mathematics, engineering and science in the university environment. Most of the students are familiar with it.

MATLAB is viewed by many users not only as a high-performance language for technical computing but also as a convenient environment for building graphical user interfaces (GUI). Data visualization and GUI design in MATLAB are based on the Handle Graphics System in which the objects organized in a Graphics Object Hierarchy can be manipulated by various high and low level commands. If using MATLAB7 the GUI design more flexible and versatile, they also increase the complexity of the Handle Graphics System and require some effort to adapt to.

1.2 Objectives

The overall aim of the whole project is to develop a toolbox that allow user to solve power flow problem. However the other objective that needed to complete are:

i. To study the performance of the transmission lines, transformer and generator at steady state condition.
ii. To obtain the simulation of power flow analysis by MATLAB.
iii. To build a toolbox for power flow analysis for education and training purposes

1.3 Scope of Project

This project concentrates on MATLAB programming and GUI designing to enable the users to calculate power flow problem. MATLAB is used to program the power flow solution and Graphical User Interface (GUI) use to help a user easy to use.

To achieve all the project's objectives, the developer must have fulfilled all the scope below:

i. Studies MATLAB programming and MATLAB GUI
ii. Identify appropriate command for MATLAB M-files
iii. Build MATLAB program for the power flow analysis using M-files
iv. Run simulation of power flow analysis using MATLAB for small, medium and large scale system.
v. Design window for Power Flow Analysis Toolbox using MATLAB GUI

1.4 Thesis Organization

This thesis consists of five chapters including this chapter. The contents of each chapter are outlined as follows;

Chapter 2 contains a detailed description each part of project. It will explain about the MATLAB GUIDE and MATLAB Programming.

Chapter 3 includes the project methodology. This will explain how the project is organized and the flow of the process in completing this project.

Chapter 4 presents the expected result of simulation runs using MATLAB GUIDE.

Finally the conclusions and future recommendation of this project are presented in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter discuss about literature review that been collected for this project. Author has refers through journals and paper especially from IEEE. This chapter consists of three parts. It is describe generally on power flow analysis problems and the solutions, Graphical User Interface in MATLAB and power system toolbox in market.

2.2 Power Flow Analysis

In power engineering, the power flow analysis (also known as load-flow study) is an importance tool involving numerical analysis applied to a power system. Unlike traditional circuit analysis, a power flow study usually uses simplified notation such as a one-line diagram and per-unit system, and focuses on various form of AC power (ie: reactive, real and apparent) rather than voltage and current. The advantage in studying power flow analysis is in planning the future expansion of power systems as well as in determining the best operation of existing systems. Power flow analysis is being used for solving power flow problem. There are three methods can be used to solve power flow analysis. The methods are Newton-Raphson method, Fast-Decoupled method and

Gauss-Seidel method. This sub-chapter will discuss all three methods generally on formula or mathematical step in order to solve power flow problem.

2.2.1 Newton-Raphson Method

Newton-Raphson method is commonly use and introduce in most text book. This method widely used for solving simultaneous nonlinear algebraic equations. A Newton-Raphson method is a successive approximation procedure based on an initial estimate of the one-dimensional equation given by series expansion.

The Newton-Raphson method using the bus admittance matrix in either first or second – order expansion of Taylor series has been evaluate as a best solution for the reliability and the rapid convergence **[3].**

$$f(x)=c \tag{1}$$

If $x^{(0)}$ is an initial estimate of the solution, and $\Delta x^{(0)}$ is a small deviation from the correct solution, we must have

$$f(x^{(0)}+\Delta x^{(0)})=c \tag{2}$$

Expanding the left-hand side of the above equation in Taylor's series about $x^{(0)}$ yields

$$f(x^{(0)}) + (df/dx)^{(0)} \Delta x^{(0)} + 1/2! (d^2f/dx^2)^{(0)} (\Delta x^{(0)})^2 + \ldots = c \tag{3}$$

Assuming the error $\Delta x^{(0)}$ is very small, the higher-order terms can be neglected, which result in

$$\Delta c^{(0)} \approx (df/dx)^{(0)} \Delta x^{(0)} \tag{4}$$

where

$$\Delta c^{(0)} = c - f(x^{(0)})$$

Adding $\Delta x^{(0)}$ to the initial estimate will result in the second approximation

$$x^{(1)} = x^{(0)} + \Delta c^{(0)} / (df/dx)^{(0)} \quad (5)$$

Successive use of this procedure yields the Newton-Raphson algorithm

$$\Delta c^{(k)} = c - f(x^{(k)}) \quad (6)$$

$$\Delta x^{(k)} = \Delta c^{(k)} / (df/dx)^{(k} \quad (7)$$

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)} \quad (8)$$

(7) can be rearranged as

$$\Delta c^{(k)} = j^{(k)} \Delta x^{(k)} \quad \text{where } j^{(k)} = (df/dx)^{(k)} \quad (9)$$

In power system analysis, $J^{(k)}$ is called the Jacobian matrix. Element of this matrix are the partial derivatives evaluated at $X^{(k)}$. It is assumed that $J^{(k)}$ has an inverse during each iteration. Newton's method, as applied to a set of nonlinear equations reduces the problem to solving a set of linear equations in order to determine the values that improve the accuracy of the estimates. [1]

2.2.2 Gauss-Seidel Method

Gauss-Seidel method is also known as the method of successive displacements. To illustrate the technique, consider the solution of the nonlinear equation given by

$$F(x)=0 \quad (10)$$

Above function is rearrange and writes as

$$x=g(x) \quad (11)$$

If $x=(k)$ is an initial estimate of the variable x, the following iterative sequence is formed

$$X^{(k+1)} = g(x^{(k)}) \quad (12)$$

A solution is obtained when the difference between the absolute value of the successive iteration is less than a specified accuracy, i.e.,

$$| x^{(+k1)} - x^{(k)} | \leq \varepsilon \quad (13)$$

Where ε is the desire accuracy

The process is repeated until the change in variable is within the desired accuracy. So the Gauss-Seidel method needs much iteration to achieve the desired accuracy, and there is no guarantee for the convergence.[1]

2.2.3 Fast Decoupled Method

When solving large scale power transmission systems, an alternative strategy for improving computational efficiency and reducing computer storage requirements is the decoupled power flow method, which makes use of an approximate version of the Newton-Raphson procedure.

The Fast decoupled power flow solution requires more iterations than the Newton-Raphson method, but requires considerably less time per iteration and a power flow solution is obtained very rapidly. This technique is very useful in contingency analysis where numerous outages are to be simulated or a power flow solution is required for on-line control.[1]

For large scale power system, usually the transmission lines have a very high X/R ratio. For such a system, real power changes ΔP are less sensitive to changes in voltage magnitude and are most sensitive to changes in phase angle $\Delta\delta$. Similarly, reactive power is less sensitive to changes in angle and most sensitive on changes in voltage magnitude. Incorporate of these approximations into the Jacobian matrix in Newton-Raphson power flow solution makes the elements of the submatrices J_{12} and J_{21} zero.

We are then left with two separated systems of equations,

$$\begin{bmatrix} \frac{\partial P_2}{\partial \delta_2} & \cdots & \frac{\partial P_2}{\partial \delta_n} \\ \vdots & J_{11} & \vdots \\ \frac{\partial P_n}{\partial \delta_2} & \cdots & \frac{\partial P_n}{\partial \delta_n} \end{bmatrix} \begin{bmatrix} \Delta\delta_2 \\ \vdots \\ \Delta\delta_n \end{bmatrix} = \begin{bmatrix} \Delta P_2 \\ \vdots \\ \Delta P_n \end{bmatrix} \tag{14}$$

$$\begin{bmatrix} |V_2|\frac{\partial Q_2}{\partial |V_2|} & \cdots & |V_n|\frac{\partial Q_2}{\partial |V_n|} \\ \vdots & J_{22} & \vdots \\ |V_2|\frac{\partial Q_n}{\partial |V_2|} & \cdots & |V_n|\frac{\partial Q_n}{\partial |V_n|} \end{bmatrix} \begin{bmatrix} \frac{\Delta|V_2|}{|V_2|} \\ \vdots \\ \frac{\Delta|V_n|}{|V_n|} \end{bmatrix} = \begin{bmatrix} \Delta Q_2 \\ \vdots \\ \Delta Q_n \end{bmatrix} \tag{15}$$

In well-designed and properly operated power transmission system:

i) Angular differences between typical buses of the system are usually so small.

$\delta_{ij} = (\delta_i - \delta_j)$ very small that results,

$$\cos\delta_{ij} \approx 1$$

$$\sin\delta_{ij} \approx 0.0$$

ii) The line susceptances B_{ij} are many times larger than the line conductances G_{ij} so that $G_{ij}\sin\delta_{ij} << B_{ij}$.

iii) The reactive power Q_i injected into any bus i of the system during normal operation is much less than the reactive power which would flow if all lines from that bus were short circuited to reference.

That is $Q_i << |V_i|^2 B_{ii}$.

$$\frac{\partial P_i}{\partial \delta_j} = -|Y_{ij}V_iV_j|\sin(\theta_{ij} + \delta_j - \delta_i) \tag{16}$$

$$|V_j|\frac{\partial Q_i}{\partial V_j} = -|V_j||Y_{ij}V_i|\sin(\theta_{ij} + \delta_j - \delta_i) = \frac{\partial P_i}{\partial \delta_j} \tag{17}$$

In Eq.(16) and Eq.(17), the off diagonal elements of J_{11} and J_{22} are given by

$$\left|V_j\right|\frac{\partial Q_i}{\partial V_j} = -\left|V_j\right|\left|Y_{ij}V_i\right|\sin\left(\theta_{ij}+\delta_j-\delta_i\right) = \frac{\partial P_i}{\partial \delta_j} \tag{18}$$

Using the identity $\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$ in Eq.(18) gives us

$$\frac{\partial P_i}{\partial \delta_j} = \left|V_j\right|\frac{\partial Q_i}{\partial \left|V_j\right|} = -\left|V_iV_j\right|B_{ij}\cos\left(\delta_j-\delta_i\right) + G_{ij}\sin\left(\delta_j-\delta_i\right) \tag{19}$$

The approximation listed above then yield the off diagonal elements

$$\frac{\partial P_i}{\partial \delta_j} = \left|V_j\right|\frac{\partial Q_i}{\partial \left|V_j\right|} = -\left|V_iV_j\right|B_{ij} \tag{20}$$

$$\frac{\partial P_i}{\partial \delta_i} = \sum_{\substack{j=1\\ j\neq i}}^{n}\left|Y_{ij}V_iV_j\right|\sin\left(\theta_{ij}+\delta_j-\delta_i\right) \tag{21}$$

$$\left|V_i\right|\frac{\partial Q_i}{\partial V_i} = -\frac{\partial P_i}{\partial \delta_i} - 2\left|V_i\right|^2 B_{ii} = Q_i - \left|V_i\right|^2 B_{ii} \tag{22}$$

The diagonal elements of J_{11} and J_{22} are shown in Eq. (8) and Eq. (9) respectively. Applying the inequality $Q_i << \left|V_i\right|^2 B_{ii}$ to those expressions yields

$$\frac{\partial P_i}{\partial \delta_i} \cong \left|V_i\right|\frac{\partial Q_i}{\partial \delta_i} \cong -\left|V_i\right|^2 B_{ii} \tag{23}$$

Substitute Eq. (19) and Eq. (20) into Eq. (14) and Eq. (15), we obtain

$$\begin{bmatrix} -\left|V_2V_2\right|B_{22} & -\left|V_2V_3\right|B_{23} & \cdots & -\left|V_2V_n\right|B_{2n} \\ -\left|V_2V_3\right|B_{32} & -\left|V_3V_3\right|B_{33} & \cdots & -\left|V_3V_n\right|B_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ -\left|V_2V_n\right|B_{n2} & -\left|V_3V_n\right|B_{n3} & \cdots & -\left|V_nV_n\right|B_{nn} \end{bmatrix}\begin{bmatrix} \Delta\delta_2 \\ \Delta\delta_3 \\ \vdots \\ \Delta\delta_n \end{bmatrix} = \begin{bmatrix} \Delta P_2 \\ \Delta P_3 \\ \vdots \\ \Delta P_n \end{bmatrix} \tag{24}$$

$$\begin{bmatrix} -|V_2V_2|B_{22} & -|V_2V_3|B_{23} & \cdots & -|V_2V_n|B_{2n} \\ -|V_2V_3|B_{32} & -|V_3V_3|B_{33} & \cdots & -|V_3V_n|B_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ -|V_2V_n|B_{n2} & -|V_3V_n|B_{n3} & \cdots & -|V_nV_n|B_{nn} \end{bmatrix} \begin{bmatrix} \frac{\Delta|V_2|}{|V_2|} \\ \frac{\Delta|V_3|}{|V_3|} \\ \vdots \\ \frac{\Delta|V_n|}{|V_n|} \end{bmatrix} = \begin{bmatrix} \Delta Q_2 \\ \Delta Q_3 \\ \vdots \\ \Delta Q_n \end{bmatrix} \tag{25}$$

We can also modify Eq. (24) and Eq. (25) to two decoupled systems of equations for n-bus network.

$$\underbrace{\begin{bmatrix} -B_{22} & -B_{23} & \cdots & -B_{2n} \\ -B_{32} & -B_{33} & \cdots & -B_{34} \\ \vdots & \vdots & \vdots & \vdots \\ -B_{n2} & -B_{n3} & \cdots & -B_{nn} \end{bmatrix}}_{B'} \begin{bmatrix} \Delta\delta_2 \\ \Delta\delta_3 \\ \vdots \\ \Delta\delta_n \end{bmatrix} = \begin{bmatrix} \frac{\Delta P_2}{|V_2|} \\ \frac{\Delta P_3}{|V_3|} \\ \vdots \\ \frac{\Delta P_n}{|V_n|} \end{bmatrix} \tag{26}$$

or

$$\begin{bmatrix} \Delta\delta_2 \\ \Delta\delta_3 \\ \vdots \\ \Delta\delta_n \end{bmatrix} = -[B']^{-1} \begin{bmatrix} \frac{\Delta P_2}{|V_2|} \\ \frac{\Delta P_3}{|V_3|} \\ \vdots \\ \frac{\Delta P_n}{|V_n|} \end{bmatrix} \tag{27}$$

And

$$\underbrace{\begin{bmatrix} -B_{22} & -B_{23} & \cdots & -B_{2n} \\ -B_{32} & -B_{33} & \cdots & -B_{34} \\ \vdots & \vdots & \vdots & \vdots \\ -B_{n2} & -B_{n3} & \cdots & -B_{nn} \end{bmatrix}}_{B'} \begin{bmatrix} \Delta|V_2| \\ \Delta|V_3| \\ \vdots \\ \Delta|V_n| \end{bmatrix} = \begin{bmatrix} \frac{\Delta Q_2}{|V_2|} \\ \frac{\Delta Q_3}{|V_3|} \\ \vdots \\ \frac{\Delta Q_n}{|V_n|} \end{bmatrix} \tag{28}$$

or

$$\begin{bmatrix} \Delta|V_2| \\ \Delta|V_3| \\ \vdots \\ \Delta|V_n| \end{bmatrix} = -[B']^{-1} \begin{bmatrix} \frac{\Delta Q_2}{|V_2|} \\ \frac{\Delta Q_3}{|V_3|} \\ \vdots \\ \frac{\Delta Q_n}{|V_n|} \end{bmatrix}$$

B_{ij} are the imaginary parts of the corresponding Y_{bus} elements.

2.3 Line Flow and Losses Calculation

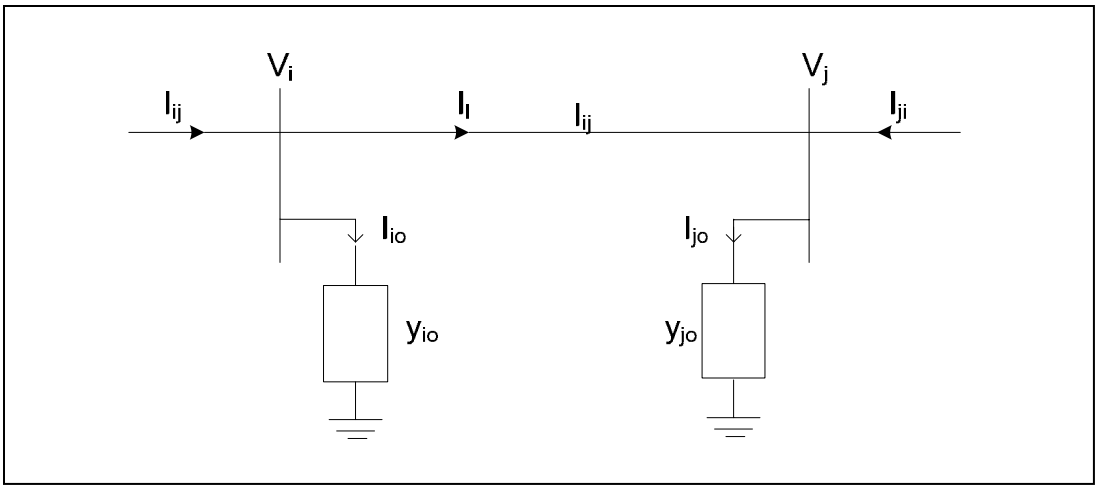

Figure 2.1: Transmission line model for calculating line flows

$i \rightarrow j$ = is given by

$$I_{ij} = I_l + I_{io} = y_{ij} (V_i - V_j) + y_{io} V_i \quad (29)$$

Similarly, the line current I_{ij} measured at bus j and defined positive in the direction j → i is given by

$$I_{ij} = -I_l + I_{io} = y_{ij} (V_j - V_i) + y_{jo} V_j \quad (30)$$

The complex power S_{ij} from bus I to j and S_{ji} from bus j to i are

$$S_{ij} = V_i I^*_{ij} \quad (31)$$

$$S_{ji} = V_I {}^*_{ji} \quad (32)$$

The power loss in line i – j is the algebraic sum of the power flows determined from (6.40) and (6.41), i.e.,

$$S_{Lij} = S_{ij} + S_{ji} \quad (33)$$

2.4 MATLAB GUI

A graphical user interface (GUI) is a pictorial interface to a program. A good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth.[6][7][8]. The GUI should behave in an understandable and predictable manner, so that a user knows what to expect when he or she performs an action. For example, when a mouse click occurs on pushbutton, the GUI should initiate the action described on the label of the button. This chapter introduces the basic elements of the MATLAB GUIs [6] [7] [8]. The chapter does not contain a complete description of components or GUI features, but it does provide the basics required to create functional GUIs for your programs [6].

Applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work [7] [8]. The action that results from a particular user action can be made clear by the design of the interface [6] [7] [8].

A graphical user interface provides the user with a familiar environment in which to work. This environment contains pushbuttons, toggle buttons, lists, menus, text boxes, and so forth [5] [6] [7] [8]. All of which are already familiar to the user, so that he or she can concentrate on using the application rather than on the mechanics involved in doing things. However, GUIs are harder for the programmer because a GUI-based program must be prepared for mouse clicks (or possibly keyboard input) for any GUI element at any time [5] [6] [7].

Such inputs are known as events, and a program that responds to events is said to be *event driven.* The three principal elements required to create a MATLAB Graphical User Interfaces are [6]:-

i. Components. Each item on a MATLAB GUI (pushbuttons, labels, edit boxes, etc.) is a graphical component. The types of components include graphical controls (pushbuttons, edit boxes, lists, sliders, etc.), static elements (frames and text strings), menus, and axes.
   Graphical controls and static elements are created by the function uicontrol, and menus are created by the functions uimenu and uicontextmenu. Axes, which are used to display graphical data, are created by the function axes [5] [6] [7] [8].

ii. Figures. The components of a GUI must be arranged within a figure, which is a window on the computer screen. In the past, figures have been created automatically whenever we have plotted data. However, empty figures can be created with the function figure and can be used to hold any combination of components [6].

iii. Callbacks. Finally, there must be some way to perform an action if a user clicks a mouse on a button or types information on a keyboard. A mouse click or a key press is an event, and the MATLAB program must respond to each event if the program is to perform its function. For example, if a user clicks on a button, that event must cause the MATLAB code that implements the function of the button to be executed. The code executed in response to an event is known as a call back. There must be a callback to implement the function of each graphical component on the GUI [6] [7].

2.5 Similar software in Market

A Windows-based Interactive and Graphic Package - For the Education and Training of Power System Analysis and operation. The authors of this software package are Joong-Rin Shin, Wook-Hwa Lee and Dong-hae Im from Department of Electrical Engineering (Kon-Kuk University, Seoul). The software represents a Windows-based Interactive Graphic Package developed for the education and training of the power system modeling, analysis and operation with user-friendly Graphical User Interface (GUI) and Visual Data Base Management System (VDBMS). Both GUI and VDBMS have been developed by the use of Borland C++ Builder and the application programs are written in C++. The application programs in this package include the Power Flow (PF) calculation, the Transient Stability Analysis (TSA), the Fault Analysis (FA), the Economic Dispatch (ED), and the Automatic Load-Frequency Control (ALFC) [2].

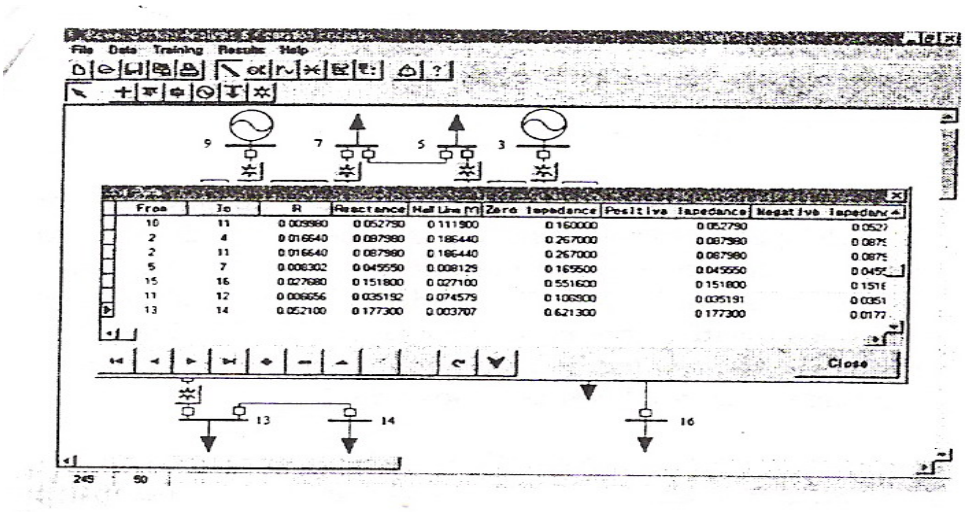

Fig. 2.2: A Windows-based Interactive and Graphic Package.

Figure 2 above shows screen shot for power flow calculation. The programming of power flow solution, the authors of this software choose to use full Jacobian Newton-Raphson method the most wide used method.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter presents the methodology of this project. It describes on how the project is organized and the flow of the steps in order to complete this project. The methodology is diverged in two parts, which is developing the programming for MATLAB and build graphical user interface in MATLAB GUIDE.

There are three mains method in order to develop this project. Before the project is developing using MATLAB, it is needed to do the study on MATLAB GUIDE and MATLAB program. The flowchart in Figure 3.1 illustrated the sequence of steps for this project. The first method is build MATLAB program of power flow solving. Secondly is run simulation and analysis using MATLAB. Lastly is developing GUI in MATLAB and programs every GUI component.

3.2 Project Flow Chart

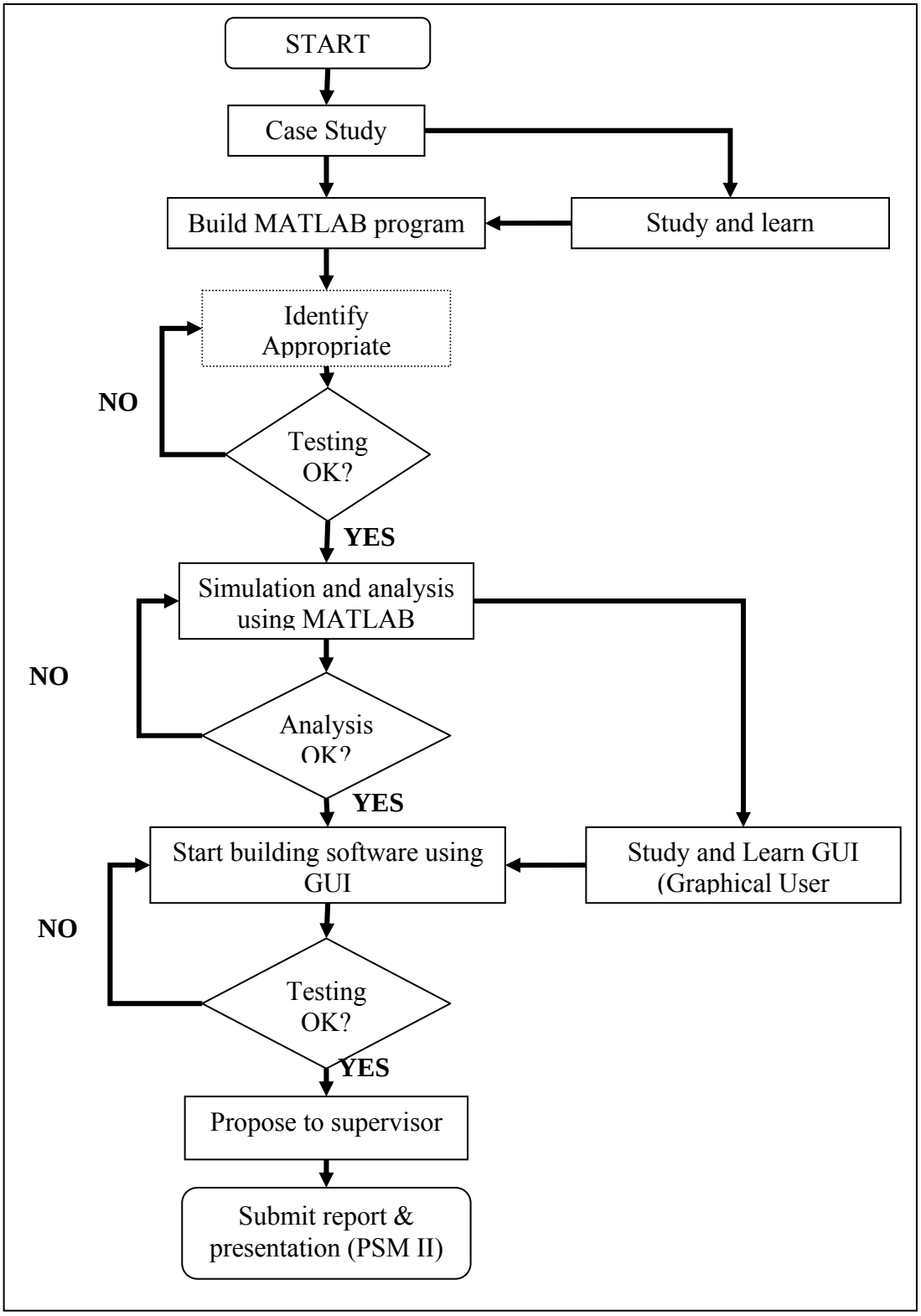

Figure 3.1: Flowchart for Whole Project

Figure 3.1 show the flow of the whole project. The project begins with doing case study about the project. First, power flow analysis programming develops using MATLAB. The second part is to run simulation of developed program in MATLAB for small, medium and large systems. After that design graphical for the software using MATLAB GUI and interface with the developed power flow analysis program. The main contribution of this project is to interface MATLAB GUI with the power flow analysis program.

3.3 Developing MATLAB GUI using MATLAB GUIDE

GUIDE, the MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs. This tool allows a programmer to layout the GUI, selecting and aligning the GUI components to be placed in.

There are 5 steps in build the MATLAB GUI. Firstly the GUI's layout is used by placing the components in Layout Editor in MATLAB GUIDE. Figure 3.2 show MATLAB GUIDE Layout Editor. The basic component of the MATLAB GUI is shown in Table 3.1.

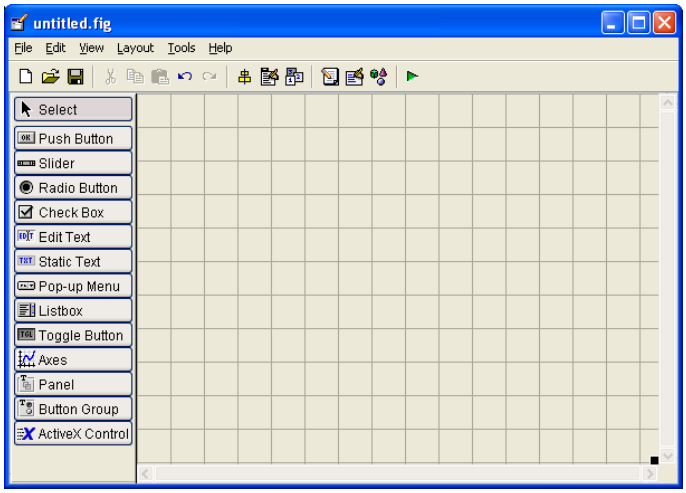

Figure 3.2: MATLAB GUIDE Layout Editor

Secondly, the MATLAB tool called the Property Inspector (built into guide) is used to give each component a name (a "tag") and to set the characteristics of each component, such as its color, the text it displays, and so on. Then, the figure is saved. When the figure is saved, two files will be created on disk with the same name but different extents. The fig file contains the actual GUI that has been created, and the M-file contains the code to load the figure and skeleton call backs for each GUI element.

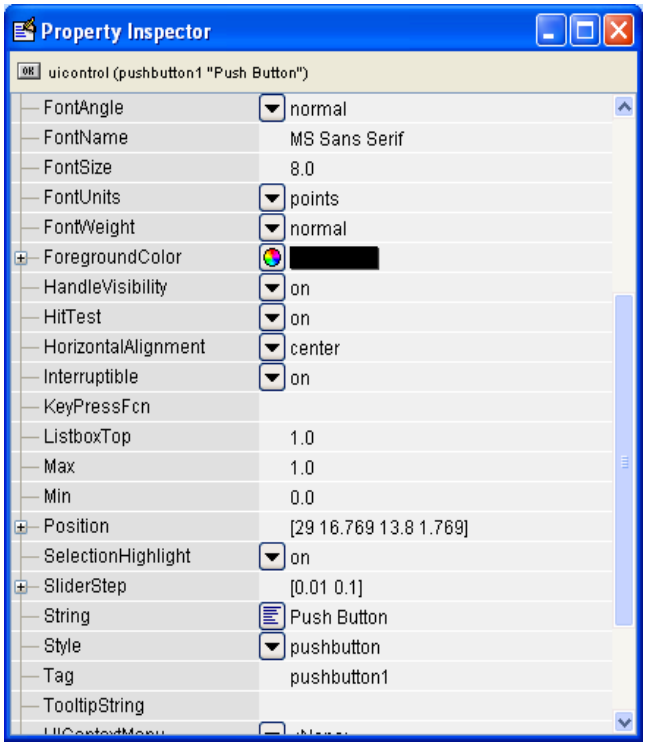

Figure 3.3: Property Inspector

Table 3.1: Basic MATLAB GUI Component [7]

Element	Created By	Description
Graphical Controls		
Pushbutton	`uicontrol`	A graphical component that implements a pushbutton. It triggers a callback when clicked with a mouse.
Toggle button	`uicontrol`	A graphical component that implements a toggle button. A toggle button is either "on" or "off," and it changes state each time that it is clicked. Each mouse button click also triggers a callback.
Radio button	`uicontrol`	A radio button is a type of toggle button that appears as a small circle with a dot in the middle when it is "on." Groups of radio buttons are used to implement mutually exclusive choices. Each mouse click on a radio button triggers a callback.
Check box	`uicontrol`	A check box is a type of toggle button that appears as a small square with a check mark in it when it is "on." Each mouse click on a check box triggers a callback.
Edit box	`uicontrol`	An edit box displays a text string and allows the user to modify the information displayed. A callback is triggered when the user presses the `Enter` key.
List box	`uicontrol`	A list box is a graphical control that displays a series of text strings. A user can select one of the text strings by single- or double-clicking on it. A callback is triggered when the user selects a string.
Popup menus	`uicontrol`	A popup menu is a graphical control that displays a series of text strings in response to a mouse click. When the popup menu is not clicked on, only the currently selected string is visible.
Slider	`uicontrol`	A slider is a graphical control to adjust a value in a smooth, continuous fashion by dragging the control with a mouse. Each slider change triggers a callback.
Static Elements		
Frame	`uicontrol`	Creates a frame, which is a rectangular box within a figure. Frames are used to group sets of controls together. Frames never trigger callbacks.
Text field	`uicontrol`	Creates a label, which is a text string located at a point on the figure. Text fields never trigger callbacks.
Menus and Axes		
Menu items	`uimenu`	Creates a menu item. Menu items trigger a callback when a mouse button is released over them.
Context menus	`uicontextmenu`	Creates a context menu, which is a menu that appears over a graphical object when a user right-clicks the mouse on that object.
Axes	`axes`	Creates a new set of axes to display data on. Axes never trigger callbacks.

After laying out the GUI component and set the property, Figure 3.4 shows the GUI layout after laying out the GUI component and set the property.

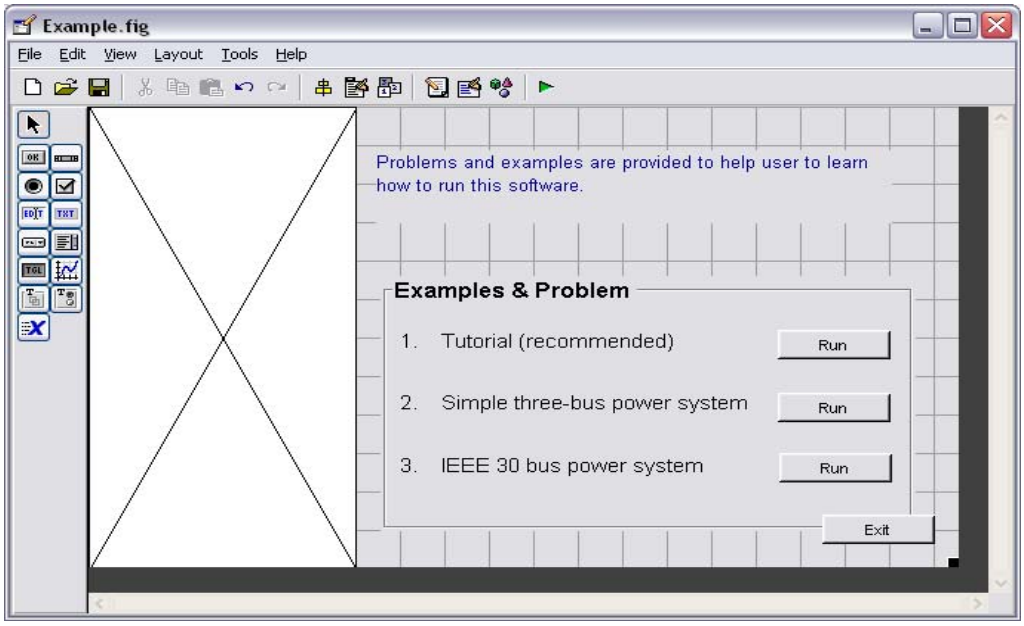

Figure 3.4: Example of GUI

Finally, the code is programmed to implement the behavior associated with each callback function in m-files as shown in figure 3.5. This last step is the most difficult part and has to make an extra reading on how to write the coding before the GUI component can perform some task.

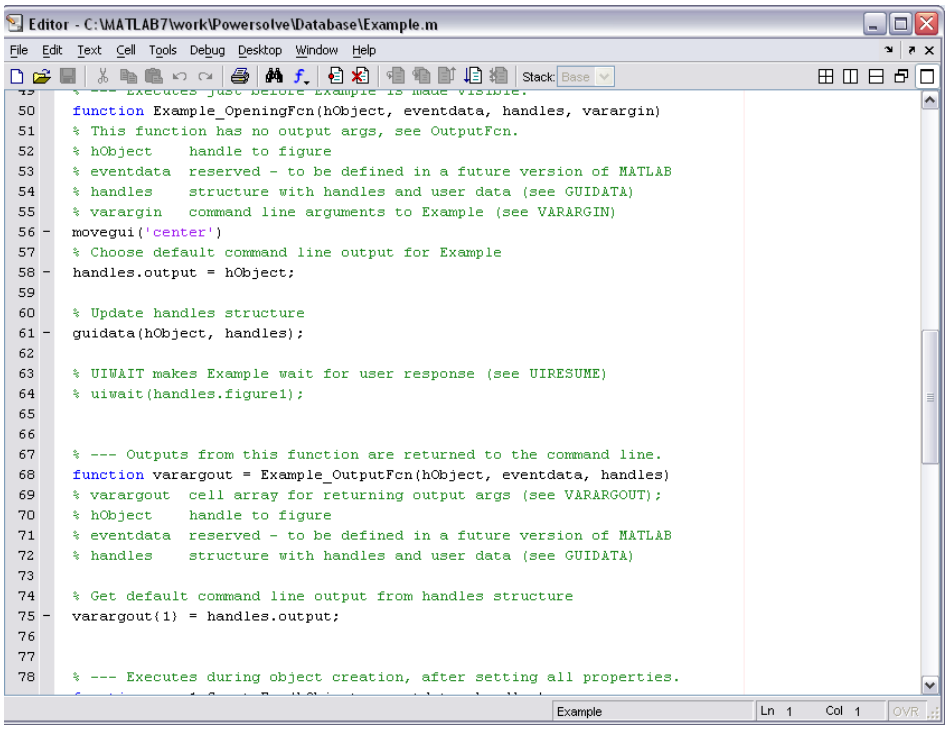

Figure 3.5: Example of M-files for GUI

3.4 Problems on Power System

3.4.1 Example of 3-Bus System

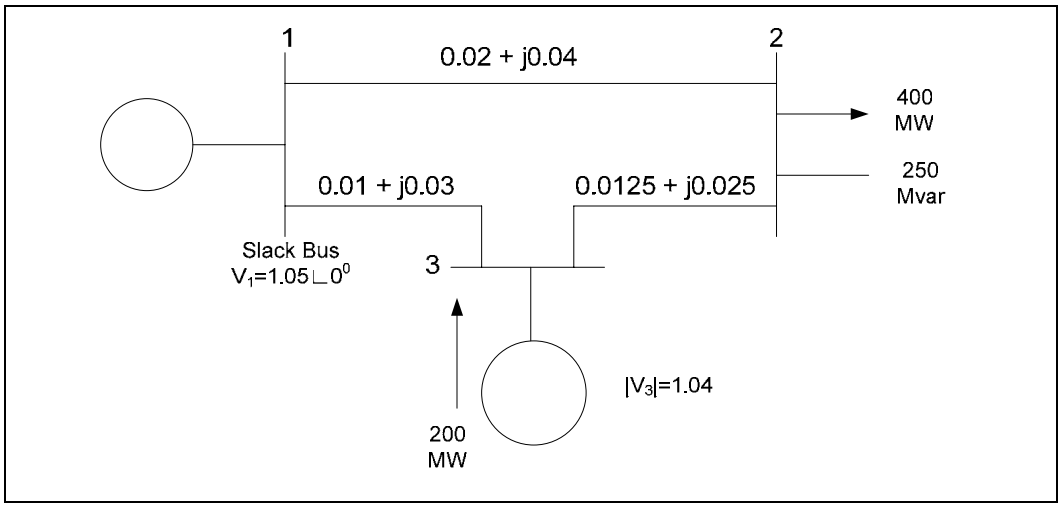

Figure 3.6: Example of 3 bus system

MVA base of system = 100MVA

Table 3.2: Bus data of 3-bus system

Bus No.	Bus Type	Voltage Magnitude	Angle Degree	load		Generator				Injected
				MW	Mvar	MW	Mvar	Qmin	Qmax	Mvar
1	Reference	1.05	0	0	0	0	0	0	0	0
2	Load	1	0	400	250	0	0	0	0	0
3	Generation	1.04	0	2.4	1.2	200	0	0	0	0

Table 3.3: Line data of 3-bus system

Bus From	Bus To	Resistance p.u	Reactance p.u	1/2 B p.u	Transformer Tap Setting Value
1	2	0.02	0.04	0	1
1	3	0.01	0.3	0	1
2	3	0.0125	0.025	0	1