DEVELOPMENT OF GRAPHICAL USER INTERFACES IN CONTROL SYSTEMS FOR EDUCATIONAL LABORATORY WORK IN THE MATLAB® ENVIRONMENT

PD PRETORIUS

VAAL UNIVERSTY OF TECHNOLOGY
PRIVATE BAG X021
VANDERBIJLPARK
GAUTENG
SOUTH-AFRICA
1900
+27 (016)-9509437
pd@tritek.ac.za

Abstract: The objective of this research was the design and implementation of reliable user-friendly Graphical User Interfaces in the MATLAB® environment, that could easily and effectively be used to solve various control and design problems. These Graphical User Interfaces have enhanced teaching and learning in the control systems laboratory.

Keywords: Graphical User Interface, Automatic Control Systems, Time- and Frequency Domain Analysis, Educational Tool.

1. INTRODUCTION

The focus of this research was aimed at developing user-friendly Graphical User Interfaces (GUIs) for second year students (users) in the control laboratory to improve their knowledge in Automatic Control Systems. The aim was to provide an added visual dimension of understanding control systems through GUIs within MATLAB® 5.3.

The GUIs developed are interactive, windows-based, graphical user interfaces created to aid in the understanding of control systems as a subject and to assist the user as a tool in the design of certain control systems. These GUIs provide the user with all of the numerical and graphical information necessary to make reliable observations, practical comparisons and conclusions on certain control problems.

It should be noted that these GUIs, although providing accurate numerical and graphical data, do not solve all control problems completely, but provide just enough information on which the user may draw conclusions and interact with the GUIs. These GUIs are essentially user-friendly design tools.

MATLAB® 5.3 is one of the most powerful mathematical software programs on the market and many industries and educational institutions are currently using this software package globally. For analysis and design in Automatic Control Systems this is one of the leading software packages available on the market. It is a Windows based program but with a DOS/C++ command foundation. The only drawback of this software is that it is not very user friendly, and not all users are sufficient computer literate for this software. By developing these GUIs a user with minimal computer skills and control systems background information can utilise this software to its highest potential.

2. GUI DEVELOPMENT (The Mathworks Inc. 1997; Glaze 1998)

The GUI development process was undertaken in two phases, design followed by implementation. The design phase was aimed at establishing an overall layout for the tool such as window placement, colour scheme, etc, whereas the implementation phase linked the various interface components with external codes, thus making it operational. In some cases, the design and implementation phases were performed simultaneously to meet a desired goal or satisfy a specified requirement of the design tool.

The Graphical User Interface Development Environment Tool (GUIDE) within MATLAB® was used to develop and design the GUIs. In fact, GUIDE generated the code for the entire layout. The GUI design tools provided a straightforward approach to the implementation phase of development. During the implementation phase, they provided a method of independently programming each object in the GUIs, allowed direct manipulation of the GUI appearance, and presented a graphical representation of the resulting layout without having to compile and execute any code. Therefore, MATLAB® provided a

simplistic, efficient means of developing computer-based design tools.

Unexpected problems occurred during the implementation phase of the GUIs. Regardless of the care taken to avoid them, they are usually responsible for adding undesired time and complexity to the implementation process. In the area of GUI implementation, most unexpected problems are generally attributed to coding mistakes, or are considered a direct result of misinterpreting the motivations behind development. The main problem that arose was to make the GUIs user proof. To overcome this problem, message boxes were created and added to the M-file code to guide or inform the user on errors made using the GUI. Figure 1 presents an example of these message boxes.

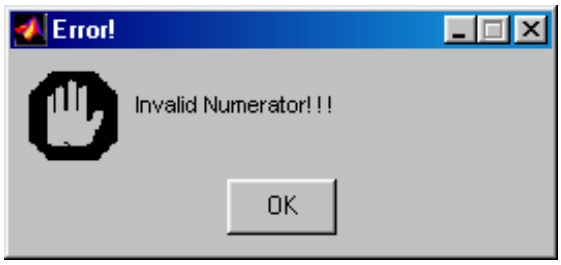

Figure 1: Message Box

2.1 Design Phase

The functionality of the GUIs was assessed during this phase. Assessment involved an iterative process because the overall functionality of any GUI is greatly dependent on the ability of the user to understand the design layout. The usefulness of the GUIs was then addressed giving special attention to the overall purpose of each GUI.

Factors that were considered in the overall design of each GUI included; required user inputs, loading and/or saving necessary data during operation, easily understood outputs, and accounting for the possibility of user error. The layout had to account for all of these conditions while effectively accomplishing the purposes of the GUIs.

Each GUI design was laid out as follows:

- Definition of the Task | Drawing the GUI
- Drawing the GUI | Testing the GUI
- Writing the Code | Testing the Code

2.2 Implementation Phase

The implementation of a graphical user interface requires fundamental knowledge of the subject being addressed. Secondary to this is the requirement that the overall design architecture be well understood. The techniques applied to make the MATLAB® GUIs operational design tools are explained in Mathworks Inc (1997). The functions that MATLAB® Method uses for graphical user interface manipulation (i.e., editing figure window and user interface control (uicontrol) properties during operation) were used extensively throughout the implementation process.

Linking the many uicontrols required the ability to locate an object or change its properties during execution. These properties are also known as MATLAB® Handle Graphics. For this, there were several MATLAB® commands. These commands were used in conjunction with "graphic handles" which are identifiers associated with graphical objects in the MATLAB environment.

Two MATLAB® Handle Graphics commands gcf (get current figure) and gcbf (get callback figure) were used specifically to locate figure windows. The gcf command was used to determine the handle of the current figure window; that is, the window where the results of commands such as plot would appear. The gcbf command was used to determine the handle of the figure window containing the object whose callback was running at the time. For example, when a push-button was pressed causing callback execution, during which its figure window needed locating, the gcbf command was issued. Each time a new figure window was opened, it became the "current figure" and as such was the handle returned by gcf.

To locate objects other than figure windows, the more generalized findobj command was used. Although this particular command has many forms, the one used throughout the implementation process discussed here was written as h = findobj ('PropertyName', 'PropertyValue') where the 'PropertyName' was the handle of all graphic objects having the property 'PropertyName', set to the value 'PropertyValue' and 'h' was the handle. To ensure proper object location using this command, the most commonly used 'PropertyName' was 'Tag', and the 'PropertyValue' was then the unique name given to that object during the design phase.

Once an object handle was determined, its properties were determined and/or modified. The two MATLAB® commands used to accomplish this are appropriately named get and set. In its simplest form, get(object), the get command was used to provide a listing of all the properties associated with an object and their current values. To determine the value of a specific object property, the get command was appended causing it to search exclusively for that property; get(object, 'propertyname'). Similarly, the set command has multiple uses; it was used to modify single sets of properties; set(object,'propertyname',new value).

3. RESULTS

The following Graphical User Interfaces were developed for the control systems laboratory:

- Poles, Zeros and Partial-Fraction Expansion GUI
- Complex Block Diagram GUI
- Routh-Hurwitz Criterion GUI

- Time and Frequency-Domain Analysis GUI

The most powerful GUI developed during this study was the Time and Frequency Domain-Analysis GUI which will be discussed here as an example.

3.1 Time and Frequency Domain Analysis GUI

This GUI provides the user with time and frequency attributes of any second order system, but its main objective is to assist the users in understanding the concept of time and frequency domain analysis of certain control systems. The user is evaluated on a variety of control problems listed in the pull-down menu of the GUI.

Use of this GUI in the control systems laboratory requires that the user understands the principles and methods used to conduct time and frequency domain analysis of certain control systems. Normally these analyses require a massive load of mathematical calculations before responses can be plotted. However GUIs facilitate simultaneous visualisation, thus stimulating the user's knowledge on time and frequency domain analysis.

3.2 Basic GUI Operation

Basic operation of the 'Time and Frequency Domain Analysis GUI' is straightforward. The six basic steps are listed below:

1. Run MATLAB®.
2. Change the present working directory to: C:\MATLABR11\toolbox\MyGuis\EIBEH2\Pr6.
3. Type IPDATA_GUI at the MATLAB® command prompt.
4. Enter the transfer function or design value in the appropriated edit boxes.
5. Observe the time and frequency attributes and interact with the GUI on the observations made if required by the control problem in question.
6. Submit the user interactive data.

3.2 Systematic Example

A certain time response is given to the user to conduct Time and Frequency Domain Analysis. This analysis is shown in the example below.

Step 1

Run MATLAB® 5.3

Step 2

Change the present working directory by typing the following at the MATLAB® command prompt:

» cd C:\MATLABR11\toolbox\MyGuis\EIBEH2\Pr6↵

»

Step 3

Load IPDATA_GUI.m by typing the following at the MATLAB® command prompt:

» IPDATA_GUI↵

»

Step 4

Enter the percentage maximum over shoot found from the time response given in the '%MO' edit box and the time when the maximum value was reached in the 'Tmax' edit box. After entering these values, the GUI will respond as follows:

- Calculates all time attributes and a certain amount of frequency attributes
- Plots a unit step response of the system entered in the LTI-Viewer

Figure 2 represents the status of the GUI after entering the design values. The LTI-Viewer with the unit step response of the system entered is shown in Figure 3.

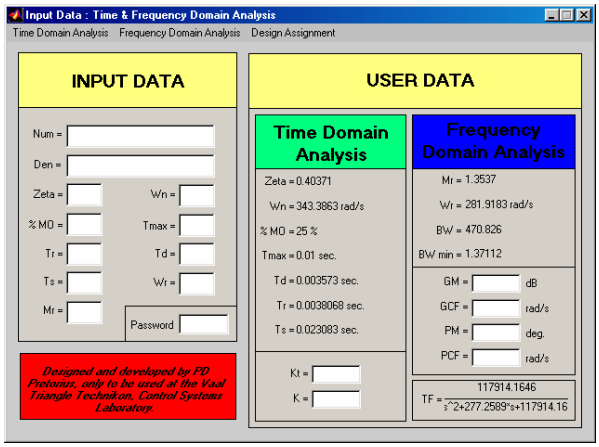

Figure 2: Step 4; Time and Frequency Domain Analysis GUI

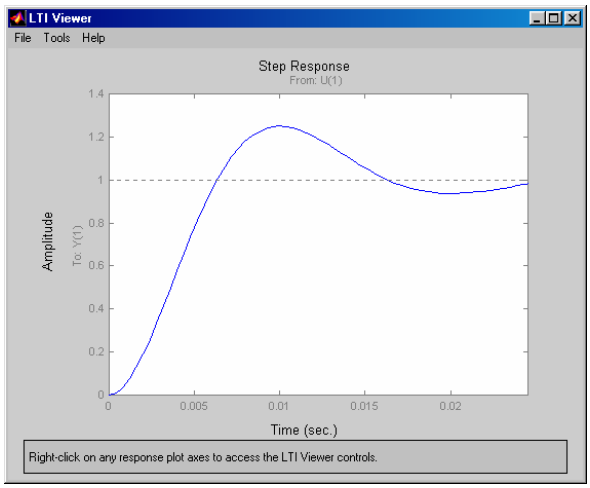

Figure 3: Step 4, LTI-Viewer with Time response

Step 5

The user data now found can be submitted via the 'Time Domain Analysis' pull-down menu. Click on the menu and check the appropriated number as shown in Figure 4. The user data found are submitted as 'Exercise 7.12' in the example. The 'Kt' and 'K' edit boxes are only used for certain control problems in which the user will enter the design values found depending on the control problem given. After submitting the control problem the user can continue with the next problem and repeat the procedures used in steps 4 to 5.

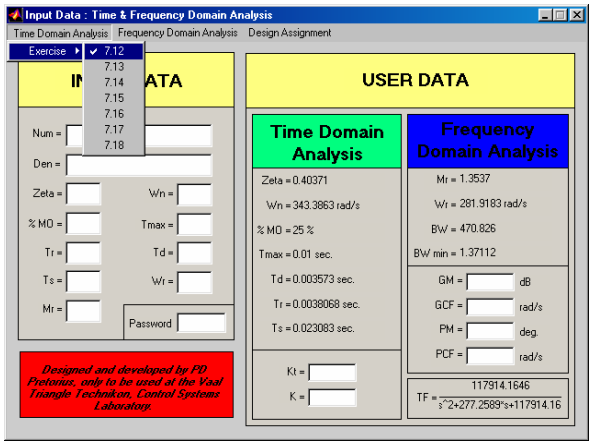

Figure 4: Step 5; Time and Frequency Domain Analysis GUI

Step 6

After entering the design values as in Step 4, the GUI responded as presented in Figure 2 and loaded the LTI-Viewer with a unit step response as shown in Figure 3. Right click with the mouse on the axis of the unit step response of the LTI-Viewer. A context menu appears with options to choose from, move the mouse pointer to the 'Plot Type' menu option and select the 'Bode' or 'Nichols' option as shown in Figure 5.

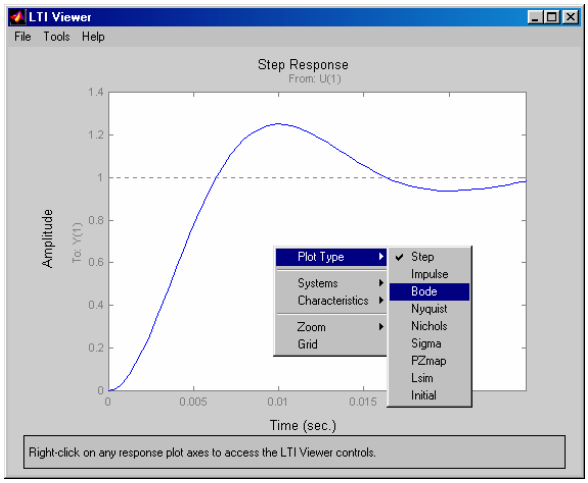

Figure 5: Step 6, selecting the 'Bode' plot using the Uicontext Menu

After clicking on the 'Bode' option the LTI-Viewer will plot the frequency response as shown in Figure 6.

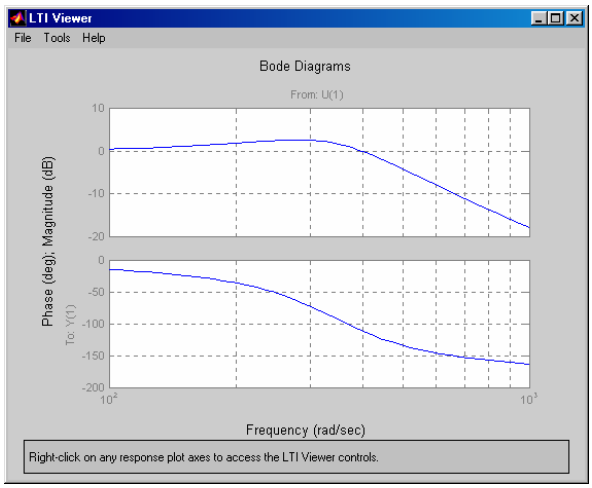

Figure 6: Step 6, LTI-Viewer with the Bode plot

Right click on any one of the axes to find attributes such as gain margin, phase margin, gain crossover frequency and phase crossover frequency. A context menu will appear, move the mouse pointer to 'Characteristics' menu option and click on the 'Stability Margins' as shown in Figure 7. The LTI-Viewer will mark the closed-loop attributes, gain margin and phase margin, with blue markers on which the user can click and by holding the mouse button, the values will be displayed as shown in Figure 8. These values are then entered in: 'GM', 'GCF', 'PM' and 'PCF' edit boxes. The values displayed in Figure 8 are as follows: Phase Margin = 69.6°, Frequency = 399 rad/sec. Using Frequency Domain Analysis theory the user can conclude that the frequency given at the point where the phase margin is measured to be the gain crossover frequency (Kuo 1995: 539-634).

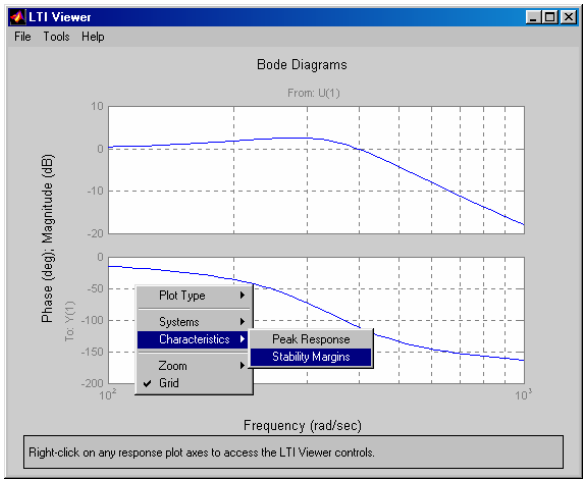

Figure 7: Step 6, selecting the 'Stability Margins' using the Uicontext Menu

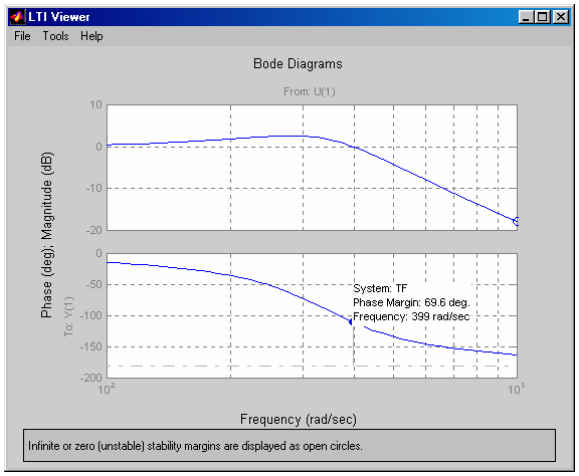

Figure 8: Step 6, stability margins

The user then enters the values found from the 'Bode' or 'Nichols' plot and then submits this user interactive data for evaluation.

The 'Kt' and 'K' edit boxes are only used for certain control problems in which the user will enter the design values found depending on the control problem given.

In this example, the user data are as presented in Table 1 and Figures 3 and 8.

Table 1 User Data

Time Domain Analysis	Frequency Domain Analysis
$Zeta = 0.40371$	$Mr = 1.3537$
$\omega n = 343.3863 rad/\sec$	$\omega r = 281.9183 rad/\sec$
$\%MO = 25$	$BW = 470.826$
$T_{\max} = 0.01 \sec$	$BW_{\min} = 1.37112$
$T_s = 0.023083$	$GM = \infty$
$T_d = 0.003573$	$GCF = 399 rad/\sec$
$T_r = 0.0038068$	$PM = 69.6^\circ$
	$PCF = \infty$

Step 7

This step is only for the use of the facilitator. Once the user has finished all control problems given, the facilitator enters a password in the 'Password' edit box and a message box will appear with the user's evaluation mark as shown in Figure 9. The square bracket associated to each "Exercise" (7.12-7-18, 9.3-9.6, 9.29, 9.30 and 9.47), indicates which specific problem has been submitted. Observing Figure 9 the facilitator concludes; that the users has submitted only Exercise 7.12 of the problems in "Exercise 7.12-7.18" and has 100 percent for that specific problem, and has done none of the other problems in this sections.

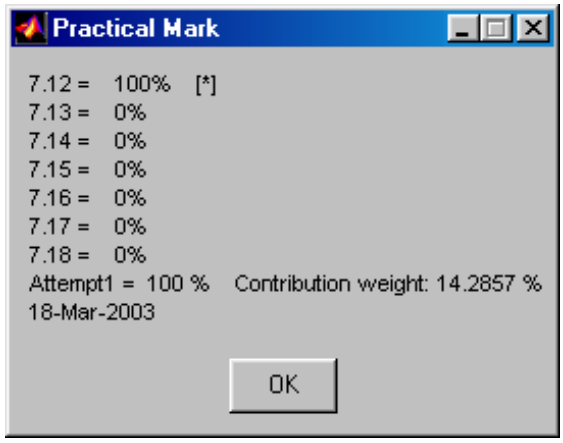

Figure 9: Step 7, Practical Mark; Time and Frequency Domain Analysis GUI

3.3 GUI Results

The interactive data found such as the time- and frequency-domain attributes, is found by means of observing the responses plotted by the LTI-Viewer and then entered via the GUI. The user's theoretical knowledge on time and frequency domain analysis is then assessed by means of the built-in self-assessment function of the GUI.

4. ADVANTAGES

Advantages of Using these GUIs Include:

- Reducing time spent on control problems.
- Minimising human error in certain mathematical calculations.
- Adding a 3rd dimension to control systems.
- Improves the users understanding of control systems.
- Provides experience of using graphical user interfaces.

Moreover there are two main advantages to the facilitator:

- Automation of the control laboratory.
- Specific assessment of the user's knowledge.

5. CONCLUSION

The development and implementation of graphical user interfaces in the MATLAB® environment has contributed to improved teaching and learning in control systems laboratories, by streamlining calculations and evaluation procedures. Moreover, students' understanding of control systems is enhanced by real-time visual (graphic) display.

REFERENCES

Cavallo A., Setola R., & Vasca F. (1996). *Using MATLAB, SIMULINK and Control Systems Toolbox*. Hertfordshire, Europe: Prentice Hall.

Glaze M.L. (1998). *The Design and Implementation of a GUI-Based Control Allocation Toolbox in the MATLAB® Environment.* Masters dissertation: Virginia State University.

Hanselam D.C. & Kuo B.C. (1995). *MATLAB Tools for Control System Analysis and Design, Second Edition.* Englewood Cliffs: Prentice Hall.

Hanselam D.C. & Littlefield B. (1996). *Mastering MATLAB®.* Upper Saddle River, New Jersey: Prentice Hall.

Kuo B.C. (1995). *Automatic Control Systems, Seventh Edition*. Englewood Cliffs: Prentice Hall.

Nakamura S. (1996). *Numerical Analysis and Graphic Visualization with MATLAB®.* Upper Saddle River, New Jersey: Prentice Hall.

Nise N.S. (2000). *Control Systems Engineering, Third Edition*. New York: Wiley & Sons.

Ogata K. (1997). *Modern Control Engineering*. Upper Saddle River, New Jersey: Prentice Hall.

The Mathworks Inc (1996). - Using MATLAB Graphics - Version 5, *MATLAB, the Language of Technical Computing*. Natick, Massachusetts: The Mathworks.

The Mathworks Inc (1997). - Building GUIs with MATLAB - Version 5, *MATLAB, the Language of Technical Computing*. Natick, Massachusetts: The Mathworks.

The Mathworks Inc (1998). - User's Guide - Version 4, *Control System Toolbox User's GUIDE*. Natick, Massachusetts: The Mathworks.

The Mathworks Inc (1999). - Using MATLAB - Version 5, *MATLAB, the Language of Technical Computing*. Natick, Massachusetts: The Mathworks.

Umez-eronini E. (1998). *System Dynamics & Control*. Pacific Grove: PWS Publishing.