Int. J. Engng Ed. Vol. 16, No. 6, pp. 544–552, 2000
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Menu-driven Graphical Interface for MATLAB Control Design*

E. K. ONG and F. L. TAN
School Of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798.
E-mail: mfltan@ntu.edu.sg

This paper describes a method of integrating various approaches to controller design into a single menu-driven graphical software interface for MATLAB. For convenience sake, the graphical interface is referred to as CADICS (Computer-Aided Design Integration of Control System). The CADICS application package was developed under a Window-based environment to assist engineering students in designing control systems using MATLAB, but who do not have detailed knowledge of MATLAB commands. Two methods of the control system design that are normally taught in engineering undergraduate control courses are incorporated into the program. In the first method (Classical Control), the plant can be compensated by PID, lead or lag compensators. Users choose the type of compensator required and the compensator parameters are automatically determined by the system so as to fulfil the design specification. In the second method (State Space Regulator), control is effected by state feedback, with or without a full-order observer. The paper highlights the program environment and the data storage structure. A case study is described using the developed software (CADICS). The software helps to relieve the load on students to be familiar with MATLAB commands. When used in a laboratory session to illustrate control principles, students will thus have more time to concentrate on the control theory rather than on the design tool, i.e. MATLAB.

SUMMARY OF EDUCATIONAL ASPECTS OF THE PAPER

1. The paper discusses materials/software for a course in dynamics and control.
2. Third-year university students of the Mechanical Engineering Department are taught in this course.
3. The mode of presentation is in the form of a regular laboratory session.
4. The hours required to cover the material is 3 hours in laboratory with 6 hours homework.
5. The novel aspects presented in this paper relate to the formulated concept and approach in devising an autonomous intelligent controller designer.
6. The standard text recommended in the course, in addition to author's notes, is *Dynamics and Control Systems* by Nise, which covers the text described.

INTRODUCTION

THE EXCELLENT control system design tool, MATLAB [1] has enabled users to solve challenging problems and compare different design techniques [2]. MATLAB commands are easy to learn. However, with the introduction of more and more new toolboxes in MATLAB, e.g., the LQG toolbox, the H_∞ toolbox, etc., there have been complaints from students in laboratory design sessions that too much time has been spent in going through the manuals in order to select the appropriate commands. As a result, they are not able to devote sufficient time to understanding the more important design principles and theories.

As an example, to illustrate the tedious design process, consider the Ziegler-Nichols method of designing a PID controller. MATLAB can design the controller for a system represented by a transfer function whose numerator and denominator are denoted by *ng* and *dg* respectively. If K_m is the gain at which the close loop system starts to oscillate, and ω_m is the oscillation frequency, then the individual P-I-D gains can be computed as follows [3]:

$$k_p = 0.6k_m, \quad k_d = \frac{k_p\pi}{4\omega_m} \quad \text{and} \quad k_i = \frac{k_p\omega_m}{\pi}$$

The result can be achieved through the following MATLAB commands:

```
>> rlocus(ng,dg),grid,
>> [km,pole]=rlocfind(ng,dg);
>> pause
>> pole2=pole(2,1);
>> wm=imag(pole2);
>> kp=o.6*km;kd=kp*pi/
(4*wm):ki=kp*wm/pi;
>> nk=[kd kp ki]
>> dk=[1 0];
```

As can be seen, the procedure requires a good understanding of the command syntax of MATLAB. If a user decides to examine the step responses or re-tune the design, the process will become more time consuming. Those unfamiliar

* Accepted 29 October 1999.

with MATLAB commands have to spend lots of time going through the manuals.

To overcome these problems, Visual Basic programming is used to create an attractive and useful program that fully exploits the graphical user interface. The existing MATLAB toolboxes are automated and integrated into one user-friendly environment that allows students to experiment with the various control systems design algorithms.

MATLAB control toolboxes allow various methods to be employed. They include:

- classical control;
- state space regulator;
- digital control;
- algebraic design (RST controller);
- linear quadratic regulator and H_∞ control.

In many engineering undergraduate control courses, however, only classical control and state space regulator methods are included. This paper therefore focuses on these methods as a foundation for future development to expand into other design methods.

PROGRAM DEVELOPMENT

This section describes the concepts in the development and highlights the features available in the CADICS program.

The overview

Visual Basic 3.0 is used to develop a Window-based environment which serves as an interface between user and MATLAB control system toolbox. Communications between the two are carried out via '.m' files which are machine independent.

Either Borland C++ or Visual Basic can be used as the programming language for the graphical interface. For the present application, Visual Basic is deemed to be more suitable as it has the following features.

- Visual Basic objects include forms, controls, and special objects such as App, Clipboard, Debug, Printer and Screen. Each object has an associated set of properties, events and methods.
- The on-line help with built-in examples makes programming easier.
- Interfacing with other software, e.g. Mathworks' MATLAB, is also easier.

Figure 1 shows an overview of the CADICS program. The existing program is developed based on modular design and is menu-driven. It has three menus, namely File, Method, and Help. Under the Method menu, two control design methods have been implemented as individual modules, namely the Classical Control Design module and State Space Regulator Design module. More modules can be added in the future to expand the control design methods.

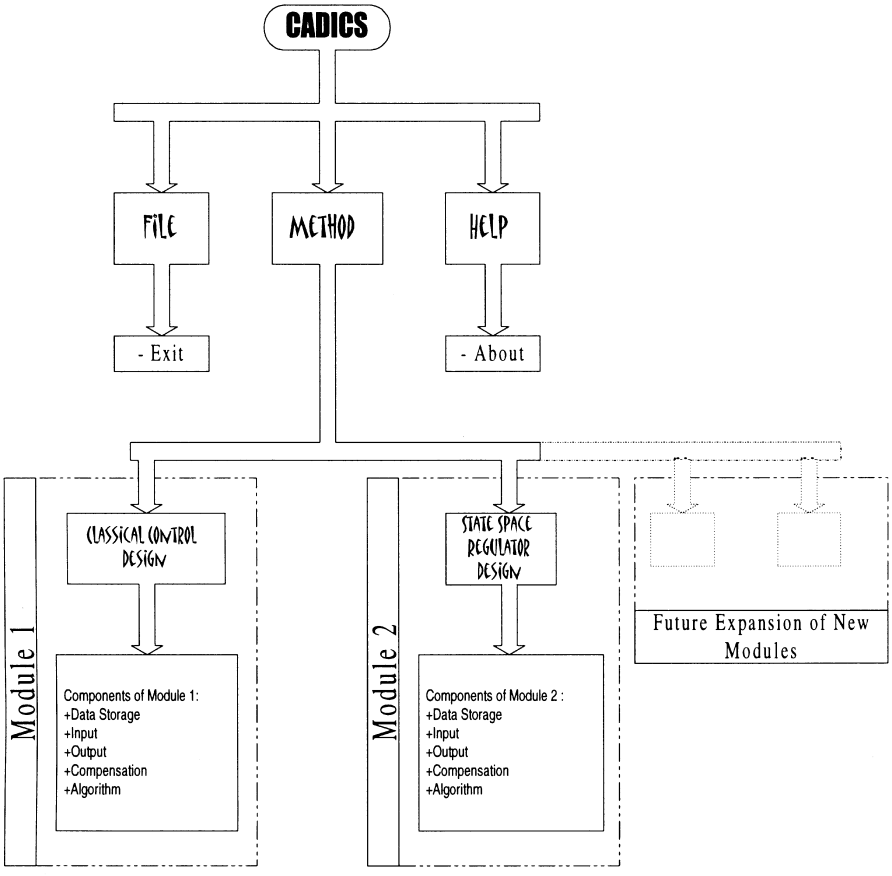

Fig. 1. An overview of the CADICS program.

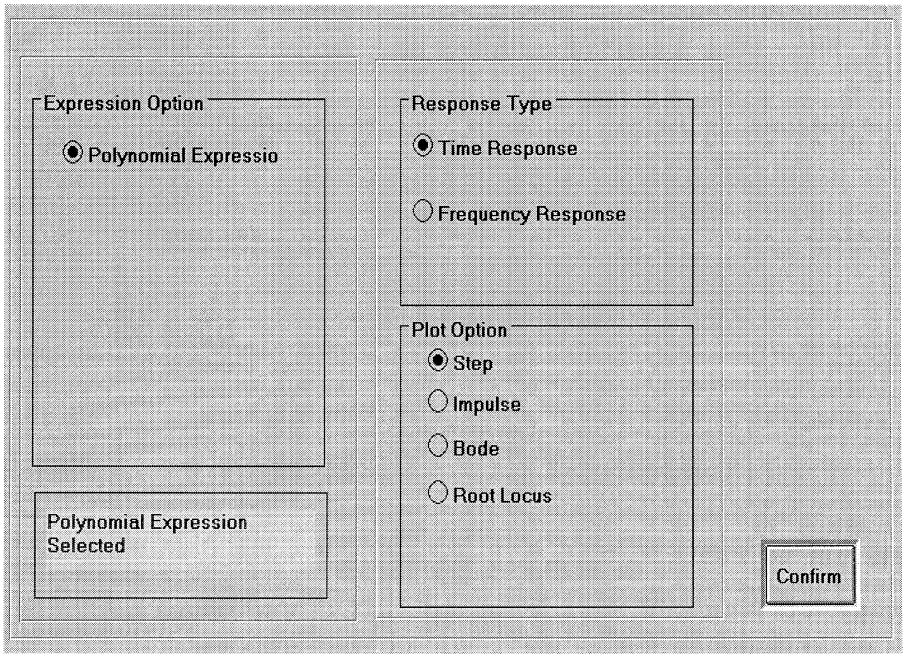

Fig. 2. Classical Control menu.

Special considerations that are taken into account during the development process are described below.

Firstly, the program is 'menu-driven' as opposed to direct keying of commands into MATLAB. It simply requires a user to be able to understand or to recognize the available options, instead of having to learn and to recall the MATLAB command and argument structure. The menu-driven feature guides the user step-by-step by suggesting viable options.

Secondly, the program is built up by modules. The program has a set of options, displayed on the screen, where the selection and execution of the options results in a change in the state of the interface. It consists of a File menu, a Help menu and a 'Method' menu. The 'Method' menu caters for different controller design algorithms. Each method is classified under one module and includes various components that have specific purposes in the program. The modular structure allows flexibility for future inclusion of new modules (methods of controller design) without affecting the operation of the existing program.

The module structure

For ease of programming and further expansion, every module is built with the same structure:

(a) *Data storage* is a buffer created for data to be stored and retrieved for processing. Input from the user to the program is performed through input boxes. Once the user has entered the necessary input data, the program will write and save the data into a file. The data is then loaded to MATLAB via '.m' files for processing.

(b) The *input format* comes in two forms. They are the 'input boxes' form which accepts numerical values and the 'input by option' form which allows the user to select one of the several options that are available in the program.

(c) Different design algorithms can be coded into the *controller design techniques* section, such as Classical Control, State Space Regulator Control method and LQG.

(d) *Output format* refers to the intermediate or final results obtained in running the application. It exists in two display modes: the graphical display mode which includes step response, root locus plot, bode plot, and the numerical values display mode which shows system parameters and the transfer function.

CADICS GRAPHICAL INTERFACE

Various functions are built into the CADICS program. As shown in Fig. 1, two control method modules were included; the Classical Control module and the State Space Regulator Control module. Upon selecting the Classical Control menu, the classical menu would appear as shown in Fig. 2.

The menu allows the user to choose between time and frequency response. The input is in polynomial expression form as shown in Fig. 3.

The Proceed button opens the input boxes for the user to enter numerical values based on the format of the expression given as shown in Fig. 4.

The parameter button only appears when the program has detected a second-order system input.

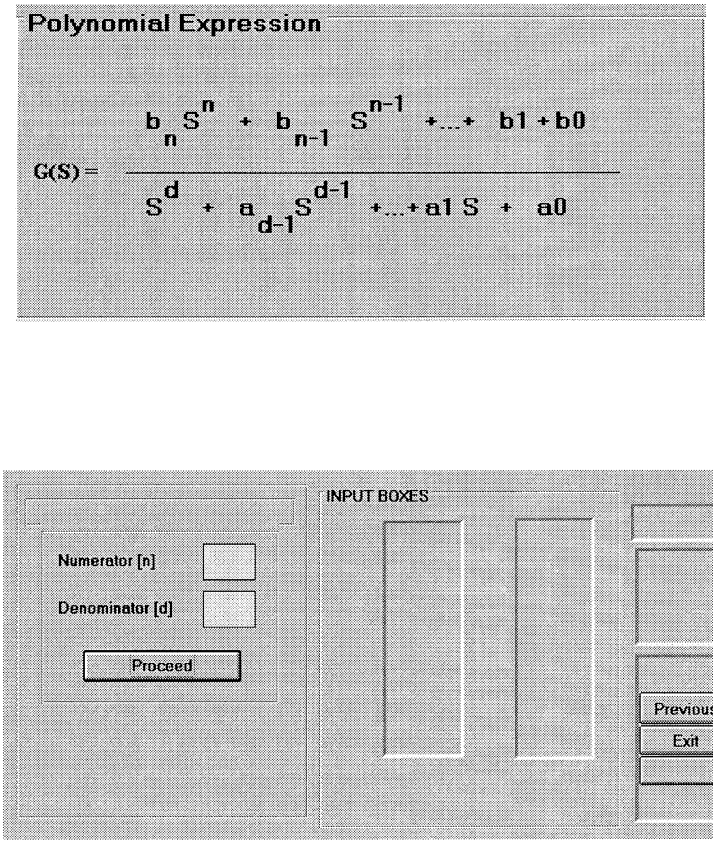

Fig. 3. Polynomial expression form.

This activates a system parameter menu as shown in Fig. 5.

Clicking the PID button activates the PID compensator menu as shown in Fig. 6. There are two design options: Ziegler Nichols and Analytical method. Upon clicking MATLAB Link button, the MATLAB Command Window would be activated followed by a display of the output plots.

Similarly, the lead/lag compensator sub-menu is designed following a 'user-friendly approach'. The procedures for a lead compensator design are highlighted as shown in Fig. 7.

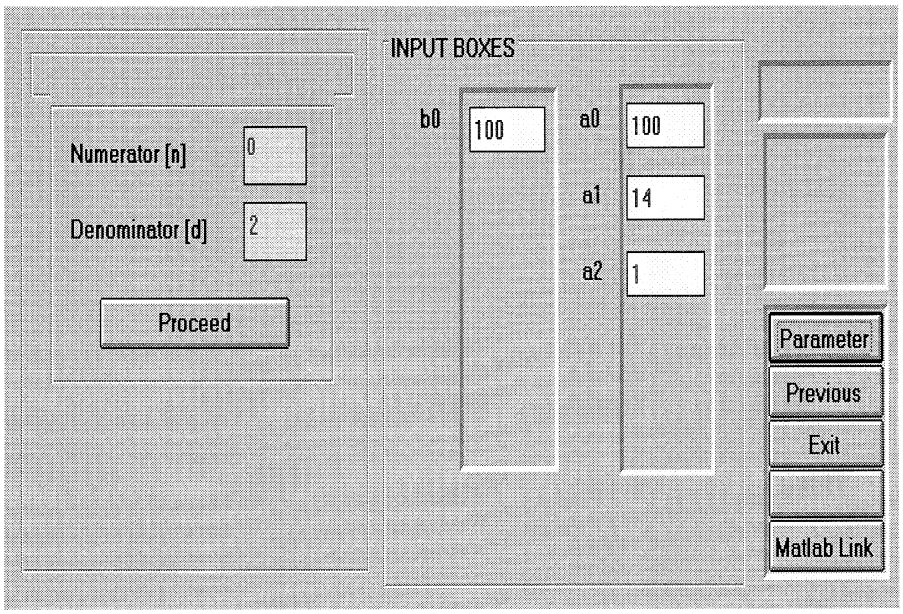

Fig. 4. Input boxes.

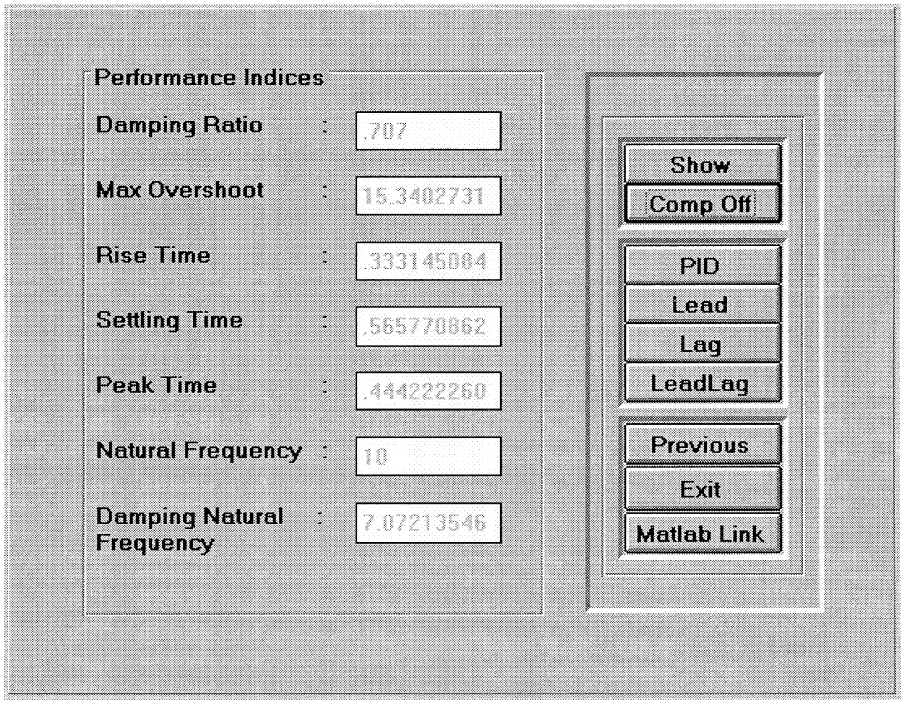

Fig. 5. System Parameter menu.

- Select the compensation method from Root Locus Design section.
- Select the required output plot from the Plot Option section.
- Enter the design specifications
- Select MATLAB Link.
- Graphs will be plotted and parameters will be displayed.

Selecting the State Space Regulator Design Menu will bring up Fig. 8. Plant parameters can be entered either as a transfer function polynomial or as a state space regulator equation.

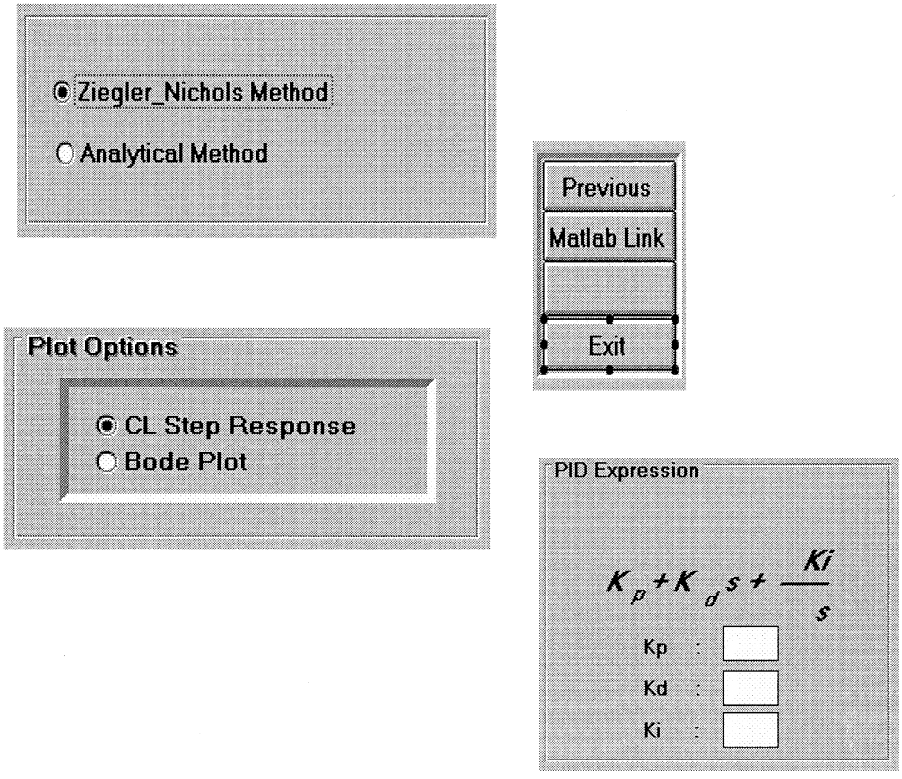

Fig. 6. PID compensator menu.

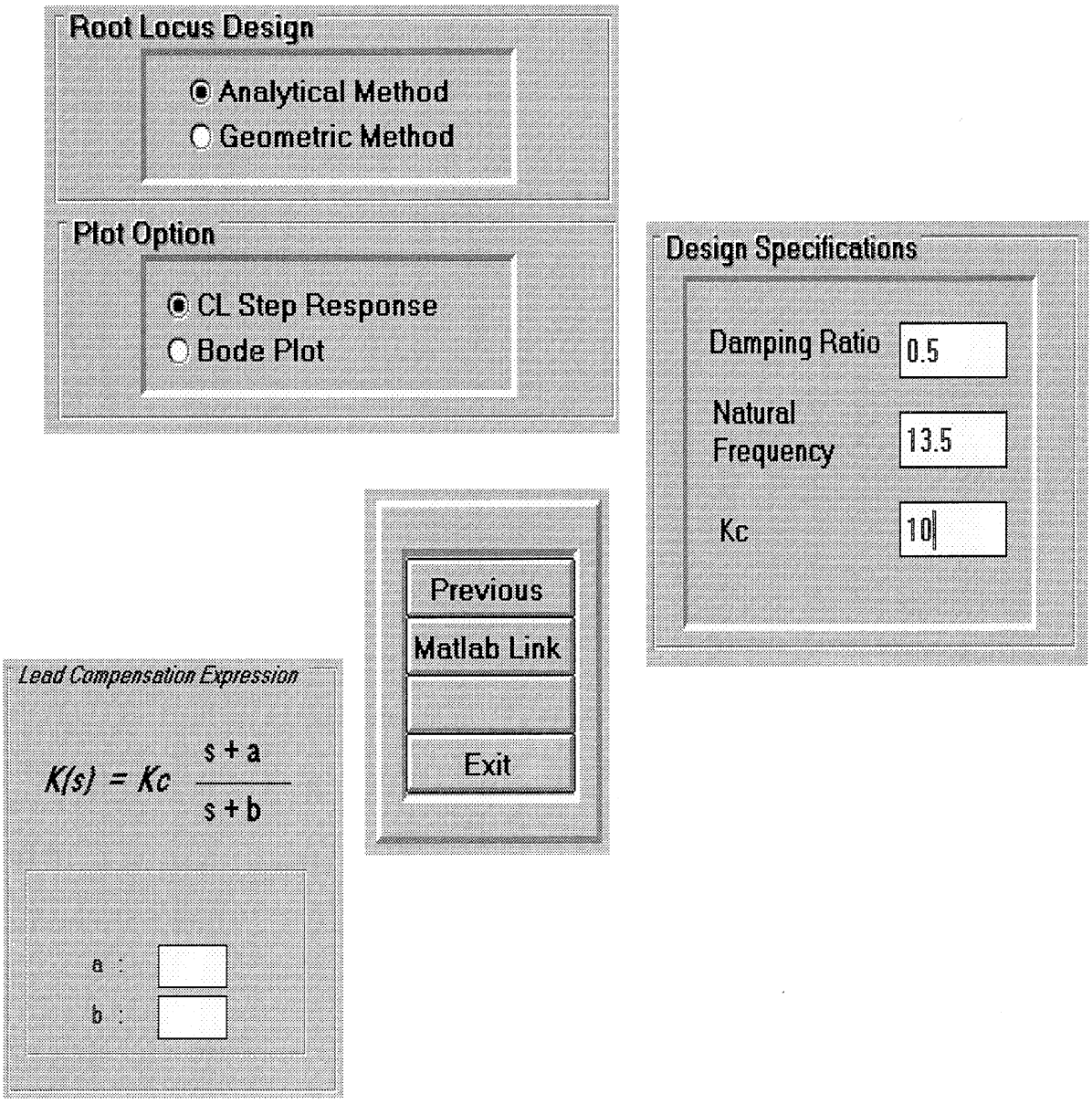

Fig. 7. Lead/lag compensator menu.

If an observer is required, clicking the appropriate icon will lead to a full-order observer-based menu. The following menu shown in Fig. 9 requires the user to enter the compensator poles as well as observer poles.

The plot options in the program allow plotting of the root locus, Bode diagram and time response of compensated systems.

CASE STUDY

In order to illustrate the usefulness of CADICS in the learning process, a comparison was made between design using MATLAB and design using CADICS for the following plant:

$$G(s) = \frac{400}{s(s^2 + 3s + 200)}$$

To compute and plot the step response from $t = 0$ to $t = 10$ s, the commands to be entered in MATLAB are as follows:

```
>> ng = 400; nk = [1 3 200 0];
>> t = [0:0.1:10]'; y = step(ng,nk,t);
plot(t,y)
```

For frequency response calculation, the syntax for the bode command is:

```
>> bode(ng,nk)
```

However, if the user wishes to view the root locus plot, the following command is needed:

```
>> rlocus(ng,nk)
```

Upon obtaining the response of the system, if the plant requires any compensation, the user will have to decide the type of compensator to be used (PID, lead or lag).

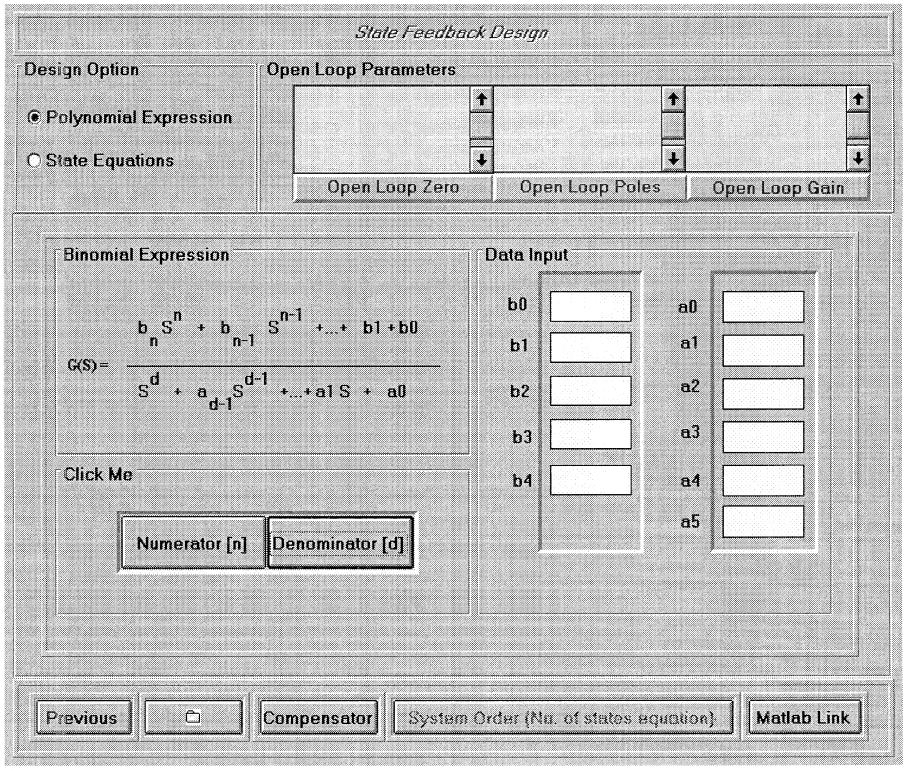

Fig. 8. State space regulator design menu.

In the Ziegler Nichols method, users can determine the gain parameters K_p, K_d and K_i using familiar equations, however, users are required to deduce the crossover gain (K_m) and crossover frequency (ω_m) first. This can be achieved by the `rlocus` and `rlocfind` commands as below:

```
>> rlocus(ng,nk);
>> [km,pole] = rlocfind(ng,nk);
>> pause
>> m = imag(pole);
```

The following commands are entered to calculate the gain parameters. The `conv` command is used to merge the transfer function with the compensator transfer function to obtain the compensated system:

```
>> kp = 0.6 x km; kd = kp x pi/(4 x (m);
ki = kp x (m/pi;
>> nk = [kd kp ki];
>> dk = [1 0];
>> num = conv(ng,nk);
>> den = conv(dg,dk);
```

The closed loop step response for the compensated plant can now be plotted using:

```
>> [ngc,dgc] = cloop(ng,dg,-1);
>> [A B C D] = tf2ss(ngc,dgc);
>> t = [0:0.01:2];
```

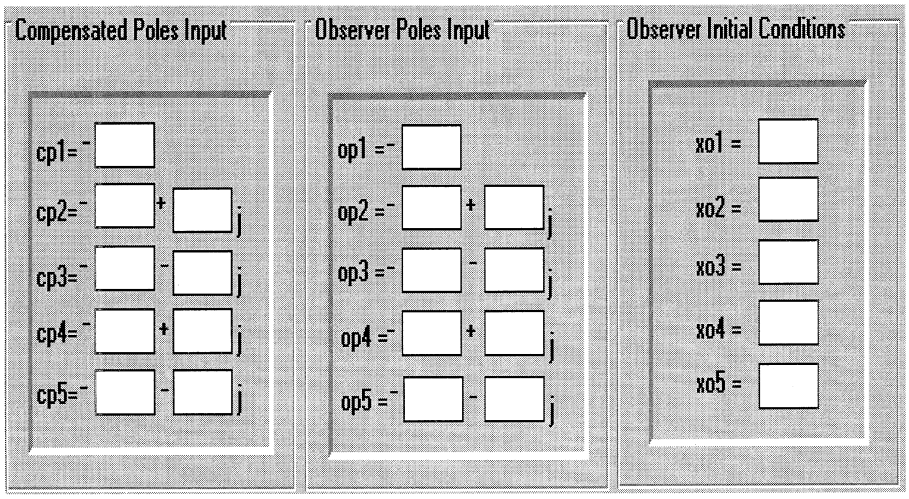

Fig. 9. Full-order observer-based menu.

```
>> Ct = step(A,B,C,D,1,t);
>> plot(t,Ct)
```

In the Analytical method, the calculations for the gain parameter are different. Users have to enter the equations to calculate the gain parameter. The syntax are as follows:

```
>> ngv = polyval(ng,j*wgc);
>> dgv = polyval(dg,j*wgc);
>> g = ngv/dgv;
>> thetar = (dpm - 180)*pi/180;
>> ejtheta = cos(ejtheta)
+j*sin(thetar);
>> eqn = (ejtheta/g)+j*(ki/wgc);
>> x = imag(eqn);
>> real(eqn)
>> kp = r;
>> kd = x/wgc;
>> if ki ~=0,
>> dk = [1 0]; nk = [kd kp ki];
>> else dk = 1; nk = [kd kp];
```

To perform lead/lag compensation, MATLAB commands and the equations have to be entered as follows:

```
>> [A B C D] = tf2ss(ngc,dgc);
>> Ct = step(A,B,C,D,1,t);
>> ngv = polyval(ng,s_1);
dgv = polyval(dg,s_1); g=ngv/dgv;
>> theta = angle(g);
>> if theta > 0; phi_c = pi-theta;end;
>> if theta < 0;phi_c = -theta;end;
>> phi = angle(s_1);
>> theta_z = real(phi+phi_c)/2;
>> theta_p = real(s_1)-imag(s_1)/
tan(theta_z);
>> z_c = real(s_1)-imag(s_1)/
tan(theta_z);
>> p_c = real(s_1)-imag(s_1)/
tan(theta_p);
>> nk = [1 -z_c];
>> dk = [1 -p_c];
>> num = conv(ng,nk);
>> [numc,denc] = cloop
(num_all,den_all);
```

From the above, it is obvious that users have to go through long and tedious procedures to obtain the compensated system. Further, the students have to be proficient with the MATLAB commands and require memorising the large number of commands. Very often, the students have to resort to flipping through the MATLAB manual to find the appropriate commands and syntaxes. Thus, this will slow down the students in a laboratory session. The same process, however, can be carried out through a few 'clicks' on the menus (similar to those menus shown in earlier section) in CADICS to achieve the same compensated system. As a result, students are able to spend more time investigating the characteristics of control systems without the burden of memorizing MATLAB Commands. The objective of the CADICS program is to emphasize the designing of a control system using MATLAB software and to investigate the characteristics of the designed system. The learning of the MATLAB commands will not be necessary through the use of the menu-driven graphical interface. The use of the CADICS interface to design the control system still require a good fundamental knowledge of the control system which is covered in the lectures of any engineering undergraduate course. The menu-driven interface will help to understand the detailed process in designing a control system.

CONCLUSION

A Window-based program (CADICS) that integrates various methods of controller design into one environment has been developed and described in this paper. The program can be used to speed up the process of controller design by the Classical Control or the State Space Regulator method. The completed program is menu-driven, which allows users to easily understand and recognize the various options. It also guides the user, step by step, suggesting viable options. These advantages, together with the ability of the program to generate different types of plots, make the menu-driven interface an even more attractive option compared to using the command-based MATLAB interface. The user time is thus more effectively spent on learning control system design rather than learning MATLAB commands. The program could be further developed by including other controller design methods such as LQR, LQG, Digital Control to the program.

REFERENCES

1. *The Mathworks Inc. MATLAB Reference's Guide,* MathWorks Inc. Natick, Mass. (1992).
2. Bahram Shahian and Michael Hassal, *Control System Design Using MATLAB,* New Jersey: Prentice-Hall (1993).
3. Norman D. Nise, *Control System Engineering,* Benjamin Cummings Publishing Company, Canada (1991).
4. *The Mathworks Inc, 1990 Control System ToolBox User's Guide,* MathWorks Inc. Natick, Mass. (1990).
5. *Microsoft Visual Basic Programmer's Guide,* Microsoft Corporation.

Eng Kian Ong was born in Singapore, 1959. He received his B.Eng in mechanical engineering from the University of Tokyo in 1984 and his MS degree in both mechanical engineering and electrical engineering from the University of Michigan, Ann Arbor in 1989. He is currently the principal of the Institutute of Technical Education, Dover, Singapore. Prior to his current post, he was a senior lecturer in Nanyang Technological University. His research interest is in the area of automation, dynamics and control and robotics.

Fock-Lai Tan was born in Singapore, 1959. He received his B.Eng in mechanical engineering from the National University of Singapore in 1984 and his MSME degree in mechanical engineering from Rensselaer Polytechnic Institute, Troy, New York in 1992. He is currently a Associate Professor at the School of Mechanical and Production Engineering in Nanyang Technological University, Singapore. His primary research interest is in the area of solidification and melting heat transfer. He has also been actively involved in the development of multimedia coursewares for university teaching and distance learning.