Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.6, No.5, 2015
www.iiste.org

A Study on Automatic Control Principle Courseware Based on MATLAB

Shouxi Zhu[1, 2*] Wenlai Ma[1, 2]
1.Flying college, Binzhou University, NO.391 of Huanghe 5th Road, Binzhou, Shandong, China
2.Aviation Information Technology R&D Center, Binzhou University, NO.391 of Huanghe 5th Road, Binzhou, Shandong, China
* E-mail of the corresponding author: zhushouxi@163.com

Abstract
The course of automatic control principle needs to draw a lot of curves, but actually it is difficult to achieve. In order to solve the problems that the diagrams of automatic control principle is difficult to draw and the knowledge is difficult to understand, this paper presents a kind of automatic control principle courseware based on VISUAL C++ and MATLAB hybrid programming. The hybrid programming methods of VISUAL C++ and MATLAB are discussed in this paper, then analyzes the concrete realization method of VISUAL C++ calling MATLAB engine. In order to get a friendly user interface, the courseware using VISUAL C++ to write GUI(Graphical User Interface) and related data processing utilizing the MATLAB control system toolbox. Through MATLAB engine, this courseware can easy to draw the Bode diagram, Nyquist curve, Root Locus diagram and so on. The courseware make full use of the advantages of VISUAL C++ and MATLAB, has a friendly GUI and basically achieved all functions of MATLAB, which are convenient for teaching. The courseware designed in this paper can help the student to study the principle of automatic control and can improve the study effect to a certain extent.
Keywords: VISUAL C++, MATLAB engine, Hybrid programming, Courseware

1. Introduction

Automatic control principle is an important specialized course for engineering majors such as Electrical engineering, Automation, Transportation, etc. It is an important foundation for further specialty study [1]. Through learning of this course, students should understand the basic concept of automatic control, the basic composition and working principle of the automatic control system, and need to master the commonly methods of design and analysis of control system [2]. Due to the nature of the course, principle of automatic control needs to draw a lot of curves, such as bode diagram, nyquist diagram, root Locus diagram and so on, etc. Under the traditional teaching mode, the diagrams are gradually drawn on the blackboard by teachers, so the diagrams are crude, not accurate and requires a lot of time, for this reason, students are not easy to understand. From the above, in actual teaching, the course of automatic control principle tends to have the problems that rather difficult to understand and abstract, students lack of interest and so on. So how to solve these problems, make the abstract problems specific, improve students' interest in learning and enhance the teaching effect has become a challenge for the principle of automatic control teaching in most colleges.

MATLAB is an essential numerical calculation and graphics processing software by MathWorks company launched in 1984 [3], and also one of the world's most widely used scientific computing software nowadays. It has a powerful scientific computing and data visualization function, integrates a variety of toolboxs for different areas [4], currently has been widely used in the field of control, communication, signal processing, and so on. The integrated control system toolbox of MATLAB is an effective way to design and analyze all kinds of complicated control system. Therefore, using MATLAB simulation becomes one of the effective ways to solve the above problems. But, because of the limitations on MATLAB development platform, the program will not run out of the MATLAB environment. The GUI is relatively simple, and thus appear insufficient flexibility in dealing with some practical applications [5]. VISUAL C++ is a windows program development platform launched by Microsoft, programs developed by VISUAL C++ has the following advantages, such as maintenance friendly, user-friendly, efficient code execution and so on. But in terms of engineering calculation and image processing, VISUAL C++ is powerless.

In this paper, we integrate MATLAB into VISUAL C++ through the MATLAB engine, and design an automatic control principle courseware. The complicated data processing runs in the background by MATLAB, and VISUAL C++ is responsible for GUI [6]. So, VISUAL C++ can make full use of the data processing function of MATLAB, reduce programming work and difficulty. This courseware mainly uses MATLAB 7.0 and VC++ 6.0, design of MATLAB 6.5 and the higher version matlab is basically the same. This courseware makes full use of the advantages of MATLAB and VISUAL C++, can solve the problems of automatic control theory teaching, such as boring, poor flexibility, too abstract and so on, can improve the students' independence and learning consciousness. To a certain extent, improve the teaching effect.

2. Summary of VISUAL C++ and MATLAB Hybrid Programming Methods

MATLAB application program interface is a very important part of MATLAB. Through the interface, the user can easily realize the interaction between MATLAB and external environment. In order to make MATLAB application program interface work more perfect and convenient, MathWorks made a lot of work. Due to the universality of MATLAB and VISUAL C++, the program interface between them becomes one of the most important MATLAB application program interfaces.

2.1 MATLAB Engine

MATLAB engine function library is a group of interface functions of MATLAB. It allows users call MATLAB functions in their own applications. VISUAL C++ can utilize MATLAB engine, adopts the Client/server model, through ActiveX connecting to MATLAB. This is equivalent to use MATLAB as a calculation engine, let it run in the background. MATLAB engine function library set up the exchange bridge of data between a user program and MATLAB process, to complete the data exchange and transmission of commands.

Through MATLAB engine, almost all MATLAB functions can realize [7], and can call other MATLAB engine through network. But this method cannot run independently without MATLAB environment, and execution efficiency is low. Therefore it cannot be used in the software development and more suitable for personal use or demonstration experiment.

2.2 Call MATLAB Math Function Library Directly

As an important part of MATLAB extension, MATLAB contains about 400 mathematical functions written in C/C++ language which involved in linear algebra, numerical analysis, Fourier transform and polynomial calculation, etc., and the function library provides a large number of matrix operation functions. Users can call functions easily in C or C++ applications according to certain rules, then easily achieve the function of complex operation in C language environment, greatly reduce programming workload. But this approach requires the designer very familiar with C/C++ language, and cannot call the MATLAB functions, especially unable to make full use of many drawing functions provided by MATLAB.

2.3 Utilizing MATLAB Compiler Supplied by MATLAB

The role of MATLAB Compiler is transforming M-file to C/C++ code (what is otherwise known as MCC command), and using C/C++ compiler compile to a standalone application [8]. The C/C++ files generated during the M-file transformation can be called by other C/C++ code in principle. Through set the MCC command options, we can compile M-file to dynamic link library files, C/C++ files, executable file and so on.

After MATLAB version 6.0, the MATLAB Compiler is upgraded, can support more data types, and has better optimization functions. In MATLAB version 7.0, the compiler version is 4.x. Compared with previous version, the MATLAB Compiler 4.x no longer compiles all M functions, and only makes the necessary interface function which be called for other functions. Those functions which have not been compiled will join to component technology file after encrypted. The MATLAB Compiler will no longer provide math library and graphics library, using MATLAB Component Runtime (MCR) environment instead.

2.4 Utilizing Matcom Software

Matcom is a third-party control provided by Mathworks. The tool can easily transform M script files and m functions to C/C++ files with the same functions, which is more powerful than MCC command. It can generate dynamic link library files (dll files) and executable files (exe files) easily, not only can convert separate script files, but also can convert nested script files, after setting the environment, you also can use MATLAB toolbox functions.

Deficiencies of Matcom are cannot support struct class, can part of the plot statements can not be executed or can not get accurate images, especially three-dimensional images. Furthermore, this method will produce a lot of redundant code in the code conversion process, and the code readability is poor, the actual implementation of efficiency cannot be improved much. Therefore, it is suitable for the conversion without three-dimensional images or small M-files.

2.5 Utilizing COM Components

COM is short for Component Object Module. It's a software development technique based on component unit provided by Microsoft which can make different codes developed in different language platforms work together [9]. In other words, COM is a Client/Server standard, provides a kind of application program interface, allowing any standards-compliant applications access. The Combuilder from MATLAB 6.5 is using the MATLAB Compiler to convert the MATLAB programs to C/C++ source programs, and then call external C/C++ compiler to compile and generate COM components which can be called for other languages. This method is simple, versatile, and almost can use any MATLAB functions (note: does not support script files, script files should be

changed to function files). So we recommend using this method in the case that the application is large, calling toolbox functions or calling more functions. MATLAB Combuilder is a new tool from MATLAB 6.5, after MATLAB 7.0, this tool is renamed MATLAB Builder for COM [10], version 1.1.

The hybrid programming method based on COM is recommended by Mathworks. MATLAB COM Builder has the same operating principle with MCC command, but it provides a more friendly GUI for users, and can generate independent application that does not rely on MATLAB environment, thus can obtain the fastest speed, and don't need to do a code transformation, makes the programming style consistent and good readability.

2.6 Comparison of the Interface Methods

The interface methods for VISUAL C++ and MATLAB is shown in table 1. We can see that the MATLAB engine can almost use all MATLAB functions and of more simple operating. So if we do not consider the running environment and execution efficiency, the MATLAB engine is a good choice.

Table 1. Comparison of the Interface Methods

Methods	Items compared			limitations
	Run independently	Programming difficulty	Execution efficiency	
MATLAB engine	No	Easy	Low	No
Call directly	Yes	Hard	High	Can call a few function, does not support drawing functions
MATLAB Compiler	Yes	Medium	Relatively low	Does not support some functions
Matcom	Yes	Easy	Medium	Does not support struct class and drawing function, especially the 3D drawing is poor
MATLAB COM Builder	Yes	Medium	Relatively high	Does not support script files

3. Concrete Implementation Method of VISUAL C++ Calling MATLAB Engine

Although MATLAB engine cannot run independently without MATLAB environment, and with low execution efficiency, this courseware still adopt MATLAB engine way, mainly based on the following aspects:

- Compared with other interfaces, MATLAB engine can support almost all of the MATLAB functions.
- The courseware is mainly used for teaching, students' and the school's lab computers are equipped with MATLAB, and also there is no requirement for execution efficiency.
- The process of MATLAB engine will show their icon in the task bar, open the process, we can see the MATLAB running process by way of engine control, and also can enter any MATLAB command in it.

3.1 MATLAB Engine Functions

MATLAB provides related function library for C language calling MATLAB engine. By calling these functions, we can realize the control and operation of MATLAB engine in C/C++ program, including engine open and close, data transmission, etc. MATLAB engine library contains a series of MATLAB engine functions that can be called from an external application. The MATLAB engine functions for C language are shown in table 2, all these functions use eng as a prefix.

Table 2. MATLAB Engine Functions

Engine function	Functional description
engOpen	Open MATLAB engine
engGetArray	Get MATLAB array from MATLAB engine for data exchange
engPutArray	Send MATLAB array to MATLAB engine from an application program
engEvalString	Execute a MATLAB command
engOutputBuffer	Create a character buffer for MATLAB text output
engOpenSingleuse	Open a separate unshared MATLAB engine
engClose	Close MATLAB engine
engSetVisible	Set MATLAB visible or invisible
engGetVisible	Return MATLAB visible
engGetVarible	Get a variable from MATLAB workspace
engPutVarible	Put the specified MATLAB variable in the MATLAB workspace

3.2 Concrete Steps

Assuming that the MATLAB installation path is E:/ma.

Step one: add header file of MATLAB engine library and the path of library functions

Open the menu "tools, Options" to choose the "Directories" option. Select "Include files" in the combo box of "Show directories for", add "E:\ma\\EXTERN\Include" (as shown in figure 1).

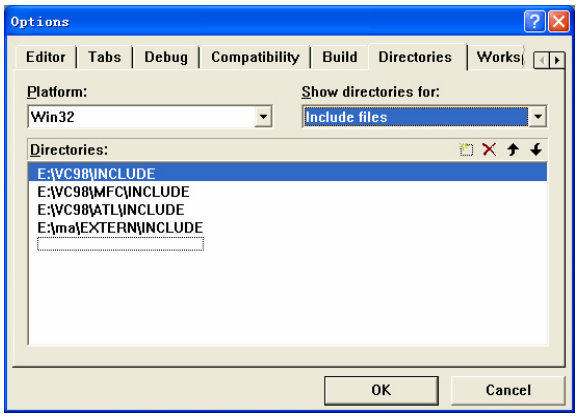

Figure 1. Add "E:\ma\\EXTERN\Include"

Then select "Library files" in the combo box of "Show directories for", add "E:\ma\EXTERN\LIB\WIN32\MICROSOFT" (under the MATLAB installation path, as shown in figure 2), the dynamic link library used by engine library is in this directory. However, in MATLAB 6.5 and lower version, the path is "E:\ma\EXTERN\LIB\WIN32\MICROSOFT\MSVISUAL C++60", the compilation environment needs to be set up only once.

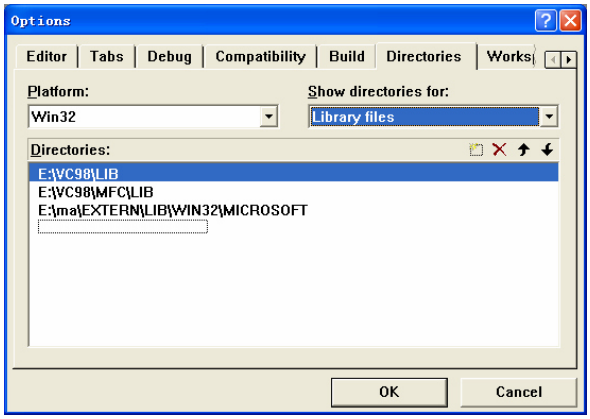

Figure 2. Add "E:\ma\EXTERN\LIB\WIN32\MICROSOFT"

Step 2: Import the corresponding dynamic link library of MATLAB engine.

Set up the project, through the menu "Projects Settings" which in the "Object/Library Modules" box, add the library file name: libmx.Lib, libmat.Lib, libeng.Lib. Note that every new project file needs to add the library files separately.

Step 3: in the program header files which using engine functions contain the following two header files:

```
#include "engine.h"
#include <stdlib.h>
```

Step 4: add the following application code that can call MATLAB engine:

```
Engine*ep;
If(!(ep=engOpen(" "))) //Open MATLAB engine to connection with local MATLAB
{
fprintf(stderr, "n  Can't start MATLAB engine”);
exit(-1);
}
```

After finish the above basic steps, we can use engEvalString function of MATLAB engine library to invoke the built-in functions of MATLAB.

4. General Design of the Courseware

In this design, we use VISUAL C++ writing the GUI of courseware for inputting data and setting up the experimental parameters. The MATLAB will process the data in the background. The communication between VISUAL C++ and MATLAB is using MATLAB engine. General design of the courseware is shown in figure 3. By calling the algorithm of MATLAB control toolbox, we can easily calculation and graph drawing. This method not only avoids the complicated programming process of control algorithm, also avoid the disadvantages that using MATLAB simulation is not completely intuitive. This method can shorten the program design cycle, also facilitate change in the future.

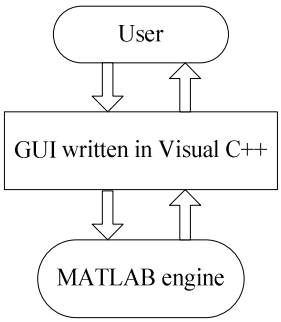

Figure 3. General Design of the Courseware

4.1 Software Structure

In the courseware, teachers can input parameters through VISUAL C++ GUI, by using MATLAB engine can realize the connection between MATLAB and VISUAL C++, call the corresponding functions in MATLAB, draw the corresponding graphics and display the results. The software structure is shown in figure 4.

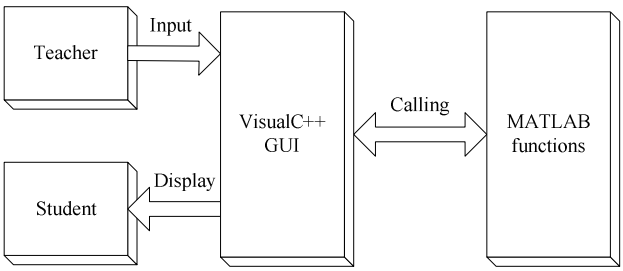

Figure 4. Software Structure

4.2 Design of GUI

4.2.1 Working Principle of GUI

The GUI through dialogs resources, dialogs dialogue and document object to realize data exchange between user and application. Its main function is following.

- Accept the input parameters.
- Sent the input parameters to MATLAB.
- Show the operation result.

The working principle of GUI is shown in figure 5.

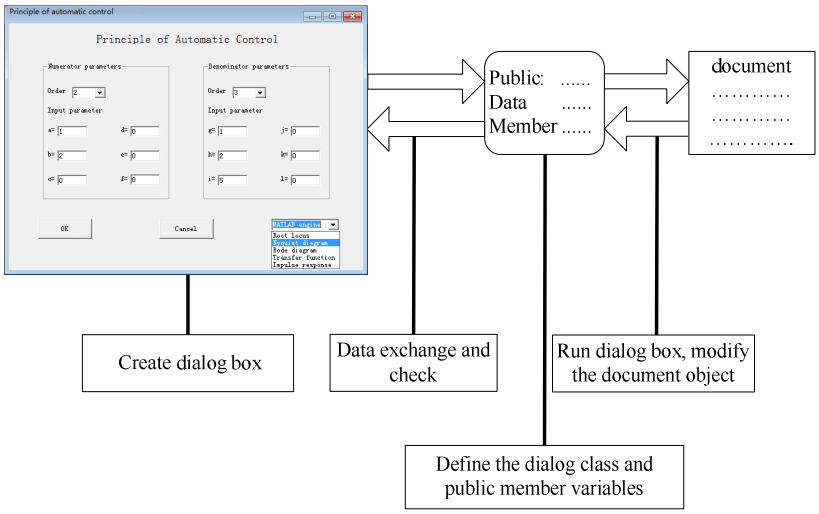

Figure 5. Working Principle of GUI

4.2.2 Brief Description of the GUI

Teachers can input parameters in the corresponding box, the maximum order of the denominator and numerator can be inputted is 5, after inputting the denominator and numerator parameters, all remaining box enters 0. After inputting the parameters, then click on the button of MATLAB engine, we can call the corresponding function in MATLAB, and the results will be displayed. Instruction of GUI is shown in figure 6.

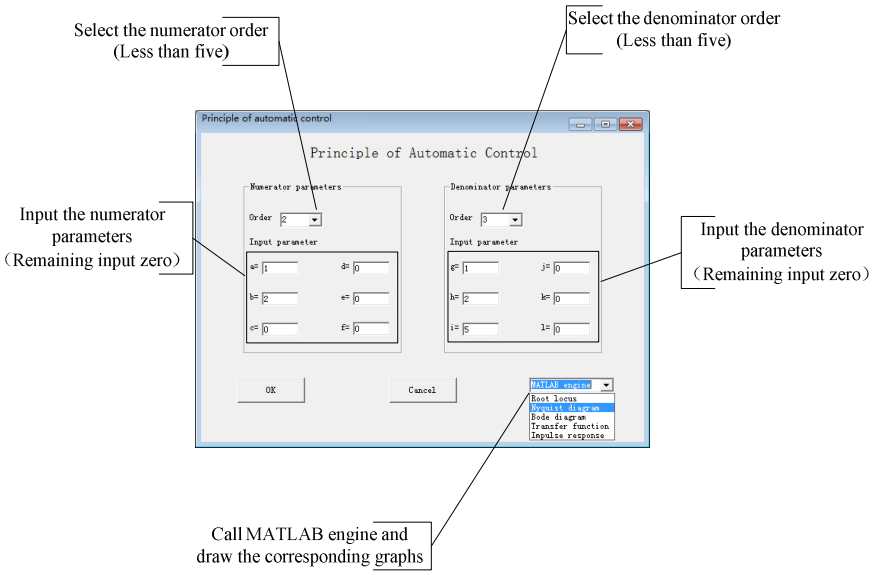

Figure 6. Instructions of GUI

4.2.3 Key Issue of Interface Input

For MATLAB engine method, the function of engEvalString (Engine*ep and const char*string) is the key to call MATLAB function, its basic function is transferring the MATLAB command from interface to background program. Among them, the ep as a pointer to the object of engine variables, defined by the keywords Engine*, the string is a pointer to the string constant variables, defined by the keywords char*. The string content that the string points to is the MATLAB command that will be sent.

The usage of the function is very simple, for example, the command line engEvalString(ep, "s=tf([a,b,c,d,e,f],[g,h,i,j,k,l]);"); can transfer the parameters of s=tf([a,b,c,d,e,f],[g,h,i,j,k,l]) of MATLAB transfer function.

The characteristics of this function brought difficulties to interface design, such as the parameters "a", "b", "c", etc., we expect it as a variable which value is decided by the user input. For EngEvalString function characteristics, however, limit it is constant, so we can not change its value directly.

In order to solve the above problem, this design adopts the following method. Set the input to CString type, so the user input values will be handled as character strings in the program [11]. By changing the values of

character strings we can change the variable values in the MATLAB program. For example, when input the value of parameter "g", we can use the following method.
First, output type the parameter "g" should be set to CString type, as show in figure 7.

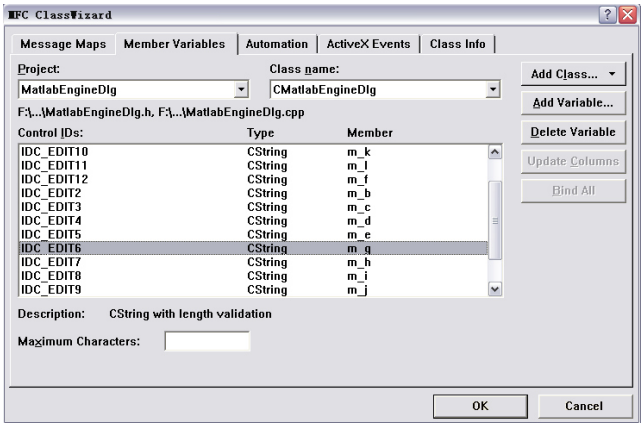

Figure 7. Set Parameter "g"to CString Type

In the message processing function, the following procedure will combine the "g" values to a complete MATLAB assignment command statements:

```
UpdateData(true);
CString g2=m_g;
CString g1="g=";
CString g3=";";
CString g4=g1+g2+g3;
char* g=(char*)(LPCTSTR)g4;
```

The command line CString g4=g1+g2+g3 can complete the combination of the assignment statements, the command line char* g=(char*)(LPCTSTR)g4 will convert CString* type to char* type, prepare for the usage of engEvalString function.
The following program fragment finally completes the full MATLAB procedure calls:

```
engEvalString(ep, g);
engEvalString(ep, "s=tf([a,b,c,d,e,f],[g,h,i,j,k,l]);");
engEvalString(ep, "figure(1),rlocus(s);");
engEvalString(ep, "sgrid");
```

The function engEvalString(ep, g) will assign the value of parameter "g", which is decided by the user's input. By this method, we can respectively complete other parameters setting, thus to solve the input problem of parameter values.

4.3 LTI Object in Control Toolbox

The LTI object in MATLAB control toolbox contains the following three subobject: ss subobject, tf subobject and zpk subobject which correspond to the state space model, transfer function model and the zero-pole-gain model, respectively.

4.3.1 Creation of LTI Model

All the LTI model can be created through the corresponding function [12]. There are five this function, in this design, we use two functions, as shown in table 3.

Table 3. The functions which can create LTI model

Function name	Function
tf(num，den，……)	Create or convert other model to transfer function model
zpk(z，p，k，……)	Create or convert other model to zero-pole-gain model

4.3.2 Functions for Drawing the Root Locus

The functions for drawing root locus in MATLAB control toolbox as shown in table 4. In these functions which use sys as input variable, can apply to both continuous and discrete systems, also can be used in multi-input multi-output system.

Table 4. Functions for drawing root locus

Function name and call format	Function
pzmap(sys)[p, z]=pzmap(sys)	Draw the system zeros and poles
z=tzero(sys)	Get the system transmission zeros
rlocfind(sys)	According to the selected point on the root locus, calculate its gain and root value
rlocus(sys)	Calculation and draw root locus
sgrid	Draw damping and grid in root plane of continuous system
zgrid	Draw damping and grid in root plane of discrete system

4.3.3 Frequency Domain Analysis Functions

The frequency domain analysis functions in MATLAB control toolbox as shown in table 5. In these functions which use sys as input variable, can apply to both continuous and discrete systems, also can be used in multi-input multi-output system.

Table 5. Frequency domain analysis functions

Function name and call format	Function
bode(sys), [mag, phase, w]=bode(sys)	Draw bode
fres=evalfr(sys, f)	Get the system frequency response of a single complex frequency point
H=freqresp(sys, w)	Get the system frequency response in a given real frequency range
[Gm, Pm, wcg, wcp]=margin(sys)	Get the system gain and phase margin
ngrid	Draw nichols grid
nichols(sys)	Draw nichols grid
nyquist(sys)	Draw nyquist
sigma(sys)	Draw bode diagram of system singular values

5. Design Examples

5.1 Draw Root Locus Diagram

The transfer function is $\frac{1}{s^4+12s^3+30s^2+50s}$, draw root locus diagram through MATLAB engine and obtain the gain of critical point. Set T_s =0.5, make the system discrete, and then do the same work.

First, establish a LTI continuous model "s" of the system, use rlocus(s) function to draw its root locus, and then type rlocfind(s) function, use the mouse to select where root locus and imaginary axis intersect. Use Bilinear transform to establish an open-loop LTI discrete model "sd" and do the same work.

Input of the interface as shown in figure 8.

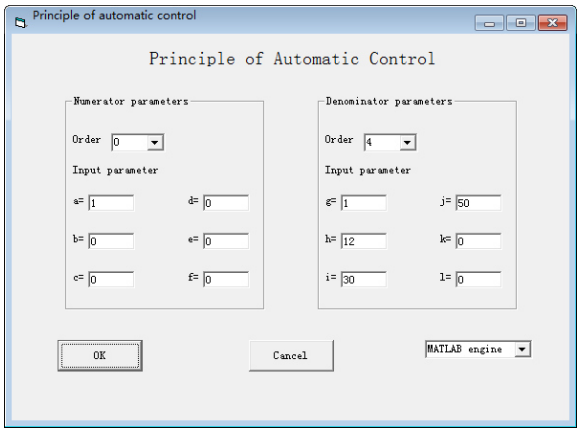

Figure 8. Input of The Interface

The VISUAL C++ program is following.

```
void CMATLABEngineDlg::OnMATLABEngine()
{
        // TODO: Add your control notification handler code here
        Engine *ep;          //Define MATLAB engine
        MessageBox("Press enter, Start MATLAB engine!", "MATLAB engine", MB_OK |
```



```
MB_ICONINFORMATION);
        if (!(ep=engOpen("\0")))  //Open MATLAB engine
        {
        fprintf(stderr, "\MATLAB engine can not start!\n");
        MessageBox("MATLAB engine can not start!","MATLAB engine ",
        MB_OK | MB_ICONERROR);
        exit(-1);
        }
        MessageBox("Press enter, hide MATLAB!","MATLAB engine", MB_OK |
MB_ICONINFORMATION);
        engSetVisible(ep, 0);   //Hide MATLAB window
        MessageBox("Press enter, redisplay MATLAB!", "MATLAB engine", MB_OK |
MB_ICONINFORMATION);
        engSetVisible(ep, 1);   //Redisplay MATLAB window
        mxArray *T = NULL;
        //Execute the MATLAB command through MATLAB engine, and the following code is MATLAB
command
        engEvalString(ep, "disp('First, the analysis of continuous system');");
        engEvalString(ep, "s=tf(1,[1,12,30,50,0]);");
        engEvalString(ep, "figure(1),rlocus(s);");
        engEvalString(ep, "sgrid");
        engEvalString(ep, "rlocfind(s);");
        engEvalString(ep, "disp('Second ,the analysis of discrete system');");
        engEvalString(ep, "sd=c2d(s,0.5,'t');");
        engEvalString(ep, "figure(2),rlocus(sd);");
        engEvalString(ep, "zgrid;");
        engEvalString(ep, "rlocfind(sd);");
        MessageBox("Close MATLAB engine, MATLAB will quit!","Root locus diagram", MB_OK |
MB_ICONINFORMATION);
engClose(ep);  //Close MATLAB engine and quit MATLAB
        }
```

After calling MATLAB engine, we can get the results. The root locus of continuous system is shown in figure 9, and root locus of the discrete system is shown in figure 10.

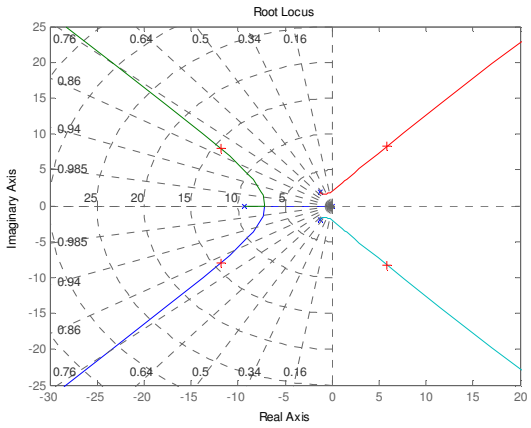

Figure 9. Root locus of the continuous system

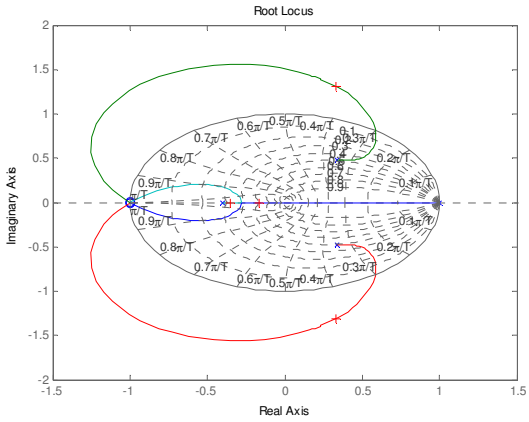

Figure10. Root locus of the discrete system

Because calling of MATLAB engine methods is similar, so in the following examples, we only provide the MATLAB program.

5.2 Draw Nyquist Diagram

The system transfer function is $\mathrm{H}(s)=\dfrac{50}{(s-1.2)(s+1)(s+6)}$, which is an open-loop unstable system. Draw its nyquist diagram and predict the closed-loop stability, then verify it by other MATLAB functions. Add a zero, and then do the same work.

First, establish LTI model "s1" with zpk function, use nyquist function to draw its nyquist diagram, and then use feedback function to get its closed loop transfer function. Predict its stability using impulse function. After adding a zero, do the same work.

MATLAB program:

```
clear
s1=zpk([],[-6,-1,1.2],50);
figure(1)
subplot(2,2,1),nyquist(s1),grid
sb1=feedback(s1,1)
subplot(2,2,2),impulse(s1),grid
subplot(2,2,3),impulse(s1),grid
subplot(2,2,4),impulse(sb1),grid
s2=zpk([-.5],[-6,-1,1.2],50);
figure(2)
subplot(2,2,1),nyquist(s2),grid
sb2=feedback(s2,1)
subplot(2,2,2),impulse(s2),grid
subplot(2,2,3),impulse(s2),grid
subplot(2,2,4),impulse(sb2),grid
```

Results:

The nyquist diagram as shown in figure 11 and figure 12. From the denominator of the two closed-loop model, we can see that system 1 is unstable and system 2 is stable, and as shown in the fourth subgraph, through the closed loop impulse response, we can get the same conclusion. Impulse response of the second subgraph indicates that the system is unstable when it is open loop.

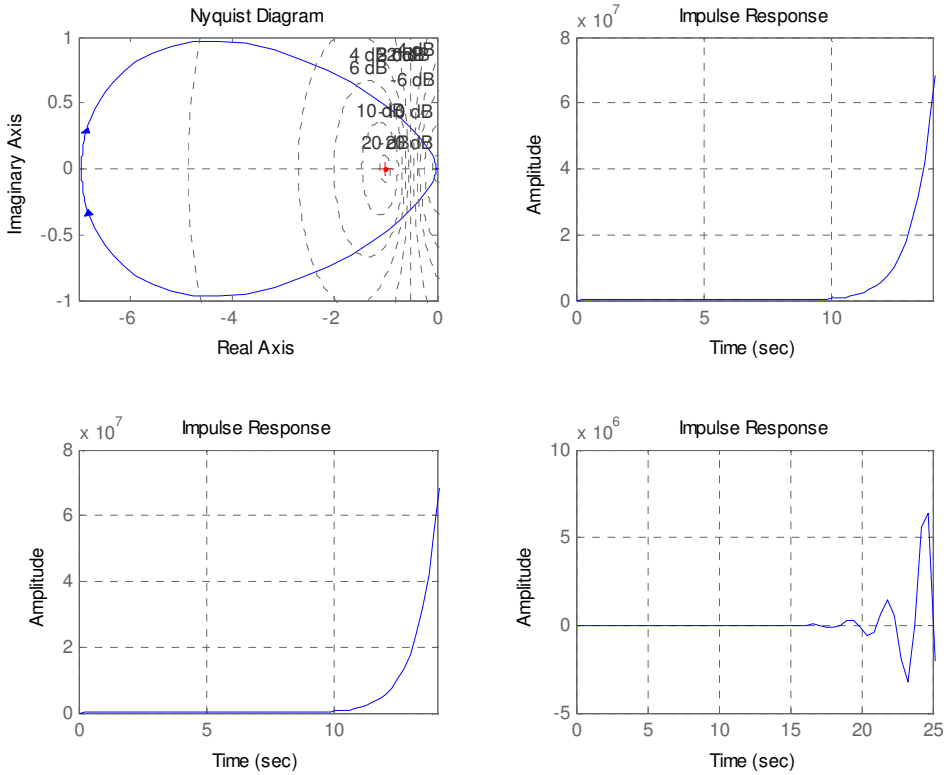

Figure 11. Nyquist diagram of system 1

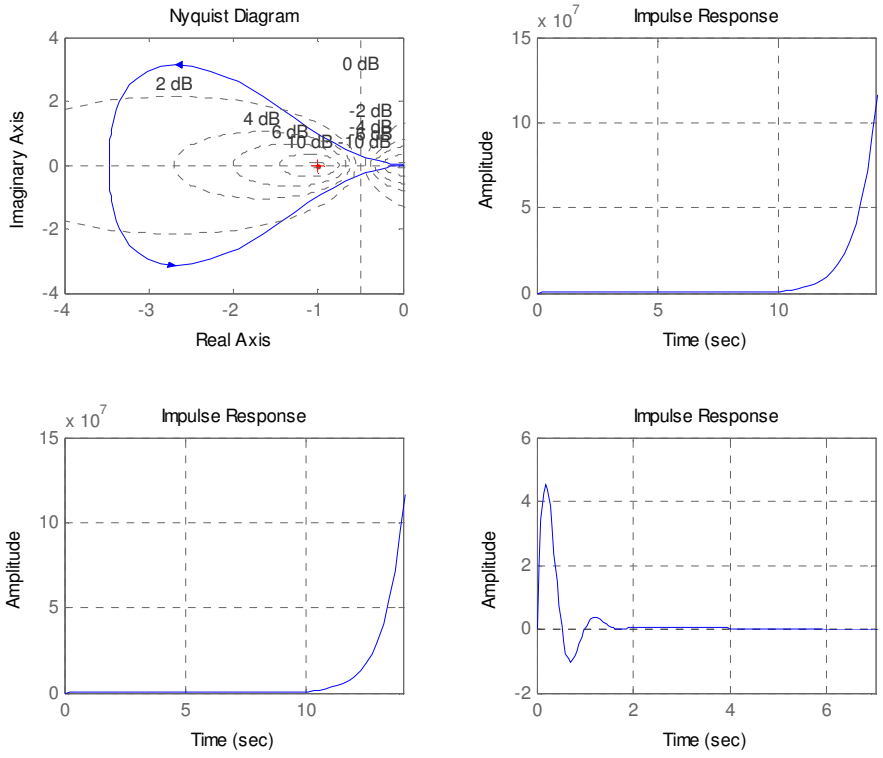

Figure 12. Nyquist diagram of system 2

5.3 Draw Bode Diagram

The system transfer function is $H(s)=\frac{200(s+6)}{s(s+1)(s+10)^2}$, draw its bode diagram, make the system discrete under the condition T_s=0.1, and then do the same work.

First, establish LTI model "s" with zpk function, use c2d function converting it to discrete LTI model "sd", then get closed-loop transfer functions of the two system by feedback function, at last, draw logarithmic frequency characteristic curve through bode function.

MATLAB program:

```
clear
Ts=0.1
s=zpk(-6,[0,-1,-10,-10],200)
sd=c2d(s,Ts)
sb=feedback(s,1)
sbd=feedback(sd,1)
figure(1),bode(s,'--',sb,'.-')
figure(2),bode(sd,'--',sbd,'.-')
damp(sb)
damp(sbd)
[Gm,Pm,wcg,wcp]=margin(s)
[Gmd,Pmd,wcgd,wcpd]=margin(sd)
```

Results:

The bode diagram of four order continuous system and discrete system as shown in figure 13 and figure 14 respectively. From the results, the continuous system is stable, but stability margin is small，and the corresponding discrete system is unstable. We also can use margin function predicting stability of system.

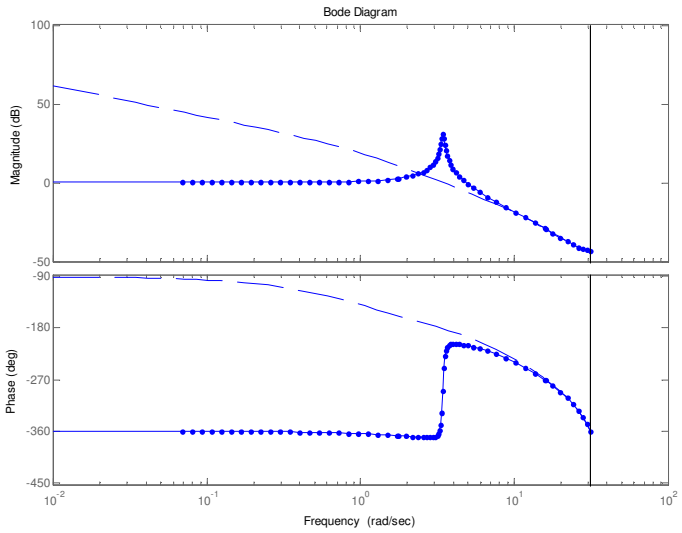

Figure 13. Bode diagram of four order continuous system

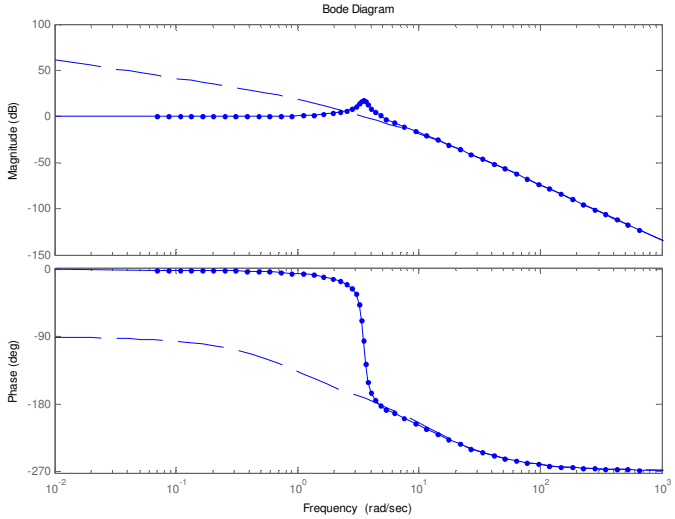

Figure 14. Bode diagram of four order discrete system

5.4 Solve the Transfer Function

If the transfer function of continuous system is $H(s)=\frac{-4s+1}{s^2+2s+10}$, sampling period is 0.2s, use the two methods of zero-order holder and bilinear transformation to solve its discrete transfer function.

MATLAB program:

```
format compact
f=[-4,1];g=[1,7,12];ts=0.2;
sc=tf(f,g)
disp('zero-order holder')
sd1=c2d(sc,ts)
disp('bilinear transformation')
sd3=c2d(sc,ts,'t')
```

Results:

```
Transfer function:
 -4 s + 1
--------------
s^2 + 7 s + 12
zero-order holder
Transfer function:
-0.3852 z + 0.4059
-----------------------
z^2 - 0.9981 z + 0.2466
 Sampling time: 0.2
bilinear transformation
Transfer function:
-0.2143 z^2 + 0.01099 z + 0.2253
--------------------------------
z^2 - 0.967 z + 0.2308
Sampling time: 0.2
```

5.5 Draw Impulse Response Function Curve

Set transfer function of a two-order system as $H(s)=\frac{1}{s^2+2\zeta\omega_n s+\omega_n{}^2}$, let its natural frequency ω_n=10, draw its impulse response function curves which damping factor ζ=[0.1,0.3,0.7,1] respectively. Set T_s=0.1, make the system discrete, and then do the same work.

First, establish LTI model "s" with ord2 function, use c2d function converting it to discrete LTI model "sd", and then draw their impulse response function curves by impulse function.

MATLAB program:

```
clear,clf
wn=10;Ts=0.1;
for zeta=[0.1:0.3:1];
[num,den]=ord2(10,zeta);
s=tf(num,den);
sd=c2d(s,Ts);
figure(1),impulse(s,2),hold on
figure(2),impulse(sd,2),hold on
end
hold off
```

Results:

The impulse response function curves of continuous system and discrete system as shown in figure 15 and figure 16 respectively.

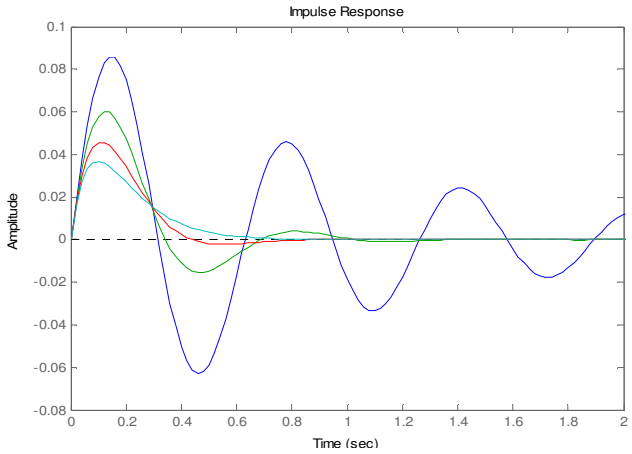

Figure 15. Impulse response function curves of continuous system

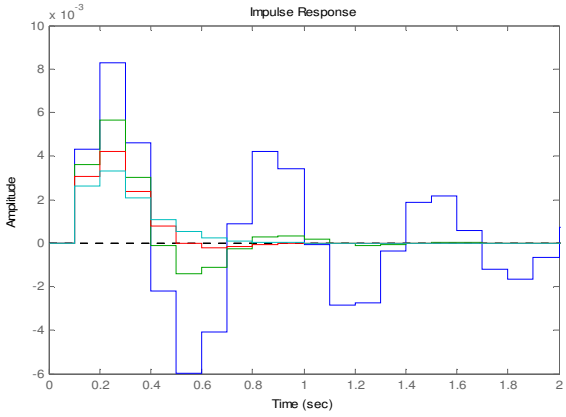

Figure 16. Impulse response function curves of discrete system

6. Conclusion

Using MATLAB engine, this paper presents an automatic control principle courseware based on VISUAL C++ and MATLAB hybrid programming. This paper studies the use of MATLAB engine to realize the drawing method of root locus diagram, nyquist diagram, bode diagram, and to solve the transfer function and impulse response curve. By the simulation results, we can see that the method of MATLAB engine can a make a good performance. The implementation is simple, and can make the students better understand the relevant knowledge by graphics. In addition, after calling MATLAB engine, we can open the MATLAB process. Using MATLAB platform, teachers can further explain the knowledge in the process of teaching. The practical application of this courseware in Binzhou University also proves the effectiveness of this courseware for improving the teaching effect. In further study, we can add more functions to the courseware, for example, add a description of the graphics after finish drawing. In addition, we also can use MATLAB COM Builder to design a courseware which can run without the MATLAB environment, that may be more convenient for teaching.

Acknowledgements
This research was financially supported by key experimental technology research project of Binzhou University (Grant NO.BZXYSYXM201401, BZXYSYXM201302).

References
[1] Li Yang, Discussions on Teaching Reform for "Automatic Control Principle" Course, *Lecture Notes in Management Science*, vol. 11, pp. 140-144, 2013.
[2] Zhang Lian, Hu Xiaoqian, Li Shan, Research and Practice of Integration of Information Technology with "Principle of Automatic Control", *Advances in Intelligent and Soft Computing*, vol. 108, pp. 339-344, 2011.
[3] Stormy Attaway, Matlab (Second Edition), *Butterworth-Heinemann*, 2011.
[4] María Jesús López Boada, Antonio Gauchía Babé, Beatriz López Boada and Vicente Díaz López, MATLAB-based educational software for vehicle's performance and longitudinal dynamics, *Computer Applications in Engineering Education*, vol. 15, no. 1, pp. 55-63, 2007.
[5] B Oakey, A virtual classroom approach to teaching circuit analysis, *IEEE Transactions on Education*, vol. 39, no. 3, pp. 287-296, 1996.
[6] MS Habib, Enhancing mechanical engineering deep learning approach by integrating MATLAB/Simulink, *International Journal of Engineering Education*, vol. 21, no. 5, pp. 906-914, 2005.
[7] Steven T Karris, Signals and Systems with MATLAB Computing and Simulink Modeling, *Orchard Publications*, 2008.
[8] MATLAB Compiler User' Guide Version 4. *The Mathworks Inc*, 2004.
[9] WS Gan, SM Kuo, Transition from Simulink to MATLAB in real-time digital signal processing education, *International Journal of Engineering Education*, vol. 21, no. 4, pp. 587-595, 2005.
[10] Liu Jie, Implementation of Hybrid Programming in Matlab/Simulink and VC++, *Science Technology and Engineering*, vol. 12, no. 16, pp. 4005-4007, 2012.
[11] Creating Graphical User Inter-faces, *The Mathworks Inc*, 2000.
[12] Alfonso Bachiller-Soler, Jose Rosendo-Macias, Antonio Gomez-Exposito, A Computer-Aided Teaching of State-Variable Formulation for LTI Circuits, *Computer Applications in Engineering Education*, vol. 21, no. 4, pp. 627-635, 2013.

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Academic conference: http://www.iiste.org/conference/upcoming-conferences-call-for-paper/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar