Open Journal of Applied Sciences, **2013, 3, 61-64**
doi:10.4236/ojapps.2013.32B012 Published Online June 2013 (http://www.scirp.org/journal/ojapps)

Real-Time Debugging and Testing a Control System Using Matlab

Zenghui Wang
Department of Electrical and Mining Engineering, University of South Africa, Florida, South Africa
Email: wangzengh@gmail.com

Received 2013

ABSTRACT

In this paper two methods for real-time debugging and testing of a control system are proposed. The basic instruments used are personal computers, a Visual C++ compiler and MATLAB including the GUI Design Environment, Simulink, real-time workshop, xPC target, and some relevant hardware. For the first method, MATLAB functions are used to build a control system debugging and testing environment. This method is flexible and only one RS-232 serial cable is used. Limited programming is used for the second method and ready-made blocks in MATLAB/Simulink are used to build the simulation environment and communication channel. In both methods, the parameters of the emulation system can be modified online, important graphs can be drawn in real time and relevant data can be easily saved for the later analysis. As can be seen from the presented examples, both techniques are easily realized.

Keywords: Control System; Matlab; Simulink; GUI; Visual C++

1. Introduction

It is a practical necessity to debug and test a controller before deploying it in a physical environment. A physical simulation system can be constructed [1], or the controller can be directly put into the field to test its performance. However, these methods have excessive time and cost implications and debugging and testing are also constraint by the set-up and physical environment. It is moreover difficult to gather data on controller performance, especially when the amount of data is large and changes quickly. However, if the simulation environment is software-defined, the process becomes much easier.

There is software available to control a PC's I/O interface. MATLAB, which is widely used, also has this capability. In Refs [2]-[4], MATLAB, supported by a C++ compiler and relevant MATLAB toolboxes, is used to realize a control algorithm and communication channel and to debug and test the controller.

In this paper, two methods using MATLAB and relevant hardware are proposed to debug and test a controller. MATLAB is used to set up a flexible software environment. A communication channel is established to facilitate the exchange of information between the controller and the PC for debugging and performance assessment.

2. First Real-time Debugging and Testing System

In order to debug and test a physical controller, the hardware must first be established. The first method limited hardware and a single computer. The hardware is employed to sample, convert and transfer the date between the controller and the computer. Sometimes, the data is not used to compute the control signals but for later analysis. The conversion changes the data format according to the transmission protocol. A PC with MATLAB is used to establish the debugging and testing environment. As most plants can be mathematically modeled, MATLAB can easily be used to describe the plant characteristics. The general system architecture for real-time debugging and testing system is shown in **Figure 1**.

In **Figure 1**, the middle block is the hardware mentioned. It is the bridge between the controller and the model. Control commands and the output of the controller are transferred via the bridge between the control hardware and the simulation software. **Figure 2** shows the system architecture from a data flow viewpoint

Since all data is exchanged through the bridge, the bridge plays an important role. Considering the flexibility, generality and cost any Microprogrammable Control

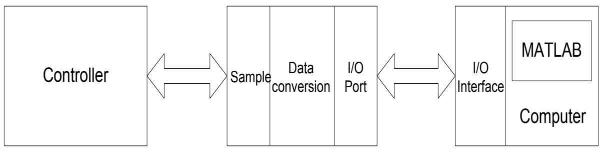


Figure 1. System architecture (1).

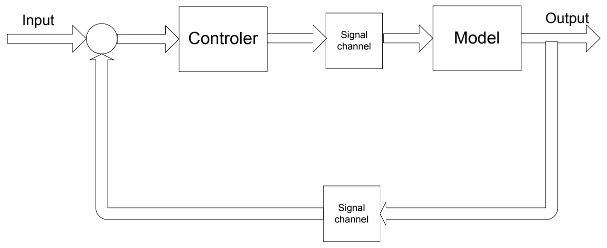


Figure 2. Block diagram.

Unit (MCU) which has a USART module to communicate with the computer and a multi-channel Analog-to-Digital converter to sample and enough digital I/O port, can act as a bridge

2) In this method, the information channel is a serial port (RS-232, RS-422 or RS-485). Among these interface standards, the most widely used interface to connect computers with peripheral devices is RS-232. Many MCUs have USART modules which can support RS-232, making it a popular choice. As MATLAB provides functionality to directly access peripheral devices, it is easy to exchange information between the controller and the model.

Although this method is simple and flexible, it has a delay-time limitation. The main reasons for this limitation are A/D conversion, data transformation and the transmission of the data between the controller and the computer. The use of this set-up is therefore limited. Nevertheless, the following method can overcome this limitation: Among the factor contributing to the delay, the RS-232 transmission time is the largest. If the MCU is fast enough, the sampling time will not be more than 50μs and the data processing time to prepare for transmission will be minimal. Although a normal desktop computer serial port can support very high baud rates, the error rate and cable length will influence data quality. The sample time and data processing time can be ignored when compared with the RS-232 transfer time. The delay time of the simulation can almost be deduced from the data transfer time which will determine whether this method suits a particular controller. To use this method, the following condition

$$T_{\max} > \frac{N_b}{f_{baud}}(8+2) + N_{AD}T_{sample} + T_p + T_{error} \quad (1)$$

must therefore be met. Here $T_{\max}$ is the maximum delay time the system can beartolerate; N_b is the number of bytes to transmit between the controller and computer; v is the RS-232 transmission baud rate; (8+2) indicates 8 bits of data and a start bit and a stop bit; N_{AD} is the number of analog-to-digital conversion channels; T_{sample} is the time for analog-to-digital conversion; T_p is the data processing time; T_{error} is the error checking time. Note that all time durations refer to a single cycle.

As mentioned

$$\frac{N_b}{f_{baud}}(8+2) + T_{error} >> N_{AD}T_{sample} + T_p \quad (2)$$

The condition can be simplified to

$$T_{\max} > \frac{N_b}{f_{baud}}(8+2) + T_{error}. \quad (3)$$

For example, if two single floating point numbers are exchanged between the controller and computer, f_{baud} is set to 19200 and there is no error checking, then the minimum delay time which the system can handle should not be less than 4.167 ms.

This method depends heavily on MATLAB programming and using the MATLAB GUI environment aids debugging and testing the controller. The following section illustrates the use of this method.

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the custom paper size (21 cm * 28.5 cm).

3. Experiment One

In this experiment, a multiple variable predictive controller is debugged and tested. The core of the control system is a PIC18F458 which is a 40-Pin high performance, enhanced FLASH microcontrollers with CAN. The control arithmetic is realized using this MCU. As this is a digital controller, it is not robust enough for real time applications and the first method can be used to debug and test this controller. As the PIC18F458 has a USART module, the hardware of middle module in **Figure 1** is not needed and its function can be realized by the MCU. In this application, the only additional hardware are a serial cable and a desktop or notebook PC. MATLAB 6.5 was used along with the instrument control toolbox, the real-time workshop and version6 GUI.

Several models were used to debug and test the controller in MATLAB. To test the robustness of the controller, some noise was added to the models. In order to analyze the control system performance, the expected trajectory is transmitted to the controller by MATLAB. The tracing trajectories are drawn in real time to make the results more clear, and the control signals and the model outputs are saved for the later analysis.

Since data is the core of the whole system, the flow of the data must be carefully arranged according to the plant configuration. To control the plant, the control system must first be given the output and the set point values followed by the control commands. **Figure 3(a)** shows the control flow chart in the MCU.

After initialization, the start of control system is triggered by the MATLAB through a protocol/trigger chosen by the designer. The time after the data is exported depends on the plant. The exported data includes not only the control information but also some important parameters used to analyze the control performance. When receiving the data, MATLAB uses the control variables to control the model and plot the output curve. Considering the whole system, the flow is shown in **Figure 3(b)**.

Since the plant model is running on the PC, it's output is almost instantly available. Generally, PC should wait for the control information. During this time, the important figures can be drawn to observe the performance of the controller.

To make the operation simple, the GUI Development Environment can be used to build a graphic interface as shown in **Figure 4**. From the graphic interface, the controller is easily controlled and its performance is directly

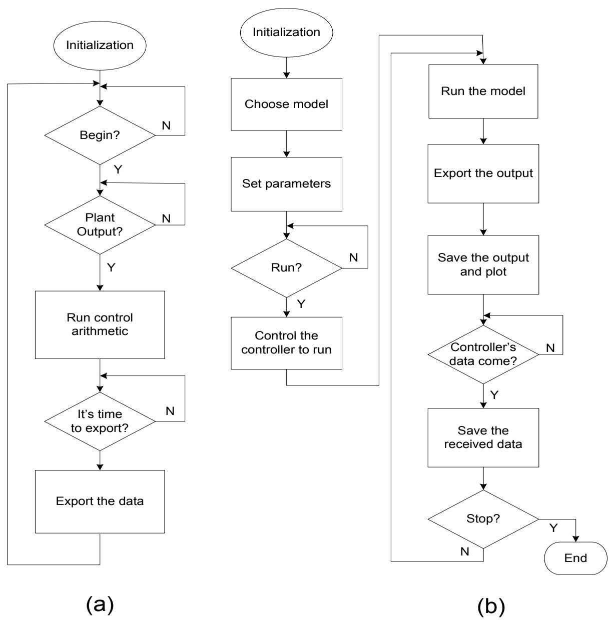
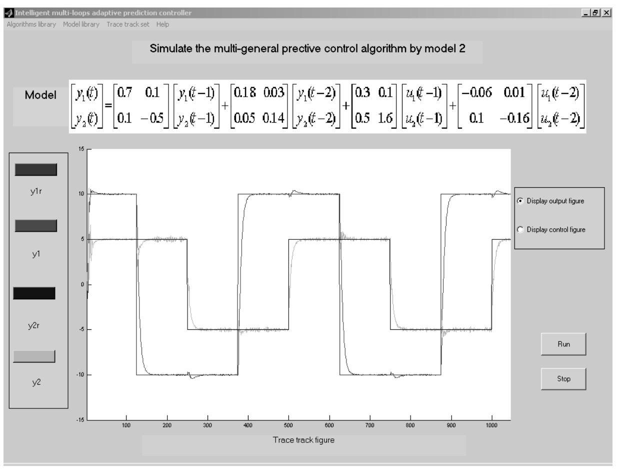


Figure 3. (a) Flow chart of the controller; (b) Flow chart in the simulating environment.


Intelligent multi-loops adaptive prediction controller
Algorithms library Model library Trace track set Help

Simulate the multi-general prective control algorithm by model 2

Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 0.7 & 0.1 \\ 0.1 & -0.5 \end{bmatrix} \begin{bmatrix} y_1(t-1) \\ y_2(t-1) \end{bmatrix} + \begin{bmatrix} 0.18 & 0.03 \\ 0.05 & 0.14 \end{bmatrix} \begin{bmatrix} y_1(t-2) \\ y_2(t-2) \end{bmatrix} + \begin{bmatrix} 0.3 & 0.1 \\ 0.5 & 1.6 \end{bmatrix} \begin{bmatrix} u_1(t-1) \\ u_2(t-1) \end{bmatrix} + \begin{bmatrix} -0.06 & 0.01 \\ 0.1 & -0.16 \end{bmatrix} \begin{bmatrix} u_1(t-2) \\ u_2(t-2) \end{bmatrix}$$

Figure 4. Simulating graphic interface.

observed. Our simulation had 3 models. Only the result of second model (model 2) is shown in **Figure 4**. In **Figure 4**, the horizontal and vertical axes indicated the time and plant outputs/expected trajectory, respectively.

4. Second Real-time Debugging and Testing System

For this method, the entire architecture, shown in **Figure 5**, is different from the first one method.Yet the flow diagram is almost the same as **Figure 2**. In MATLAB, Simulink and other blocks are used to construct the simulation environment. To model the plant, the designer can use the Simulink models. It is easily used, and little programming is needed. xPC Target is used to set up the connections between the computer and the controller. xPC Target provides many hardware board driver blocks which support analog, digital, CAN, GPIB, RS232, UDP, counters, timers, and signal conditioning. There are three types of embedded target applications. For simplicity, the BootFloppy was chosen as the xPC Target Embedded Option. Both a host PC and a target PC are therefore needed. The host PC can be a desktop or notebook PC on which MATLAB, Simulink, Real-time workshop, xPC Target, xPC Target Embedded Option and visual C++ compiler must be installed. Here a desktop computer is acted as the target PC which must has the resource of CPU, RAM, and serial port or network adapter. The network TCP/IP was chosen as the Host-Target connection, since it has more advantages than serial RS232.

This method is superior to trhe first method. Because a variety of data acquisition cards can be used to transfer data, the delay introduced can be minimised. As the whole environment is constructed using Simulink andblocksets, the connection of the whole system is more like the real system. There are many ready-made models that can be used to construct the system. Moreover, remote testing and debugging can be realize if network TCP/IP is chosen as the Host-Target connection.

The following section illustrates the use of this method.

5. Experiment Two

A single variable general predictive controller was tested using the second method. The communication channel between the controller and the target PC was also serial RS232. After the hardware was set up, the following steps were used to set up Simulink and xPC Target to

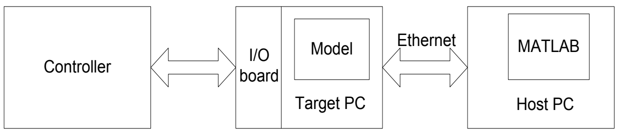


Figure 5. System Architecture (2).

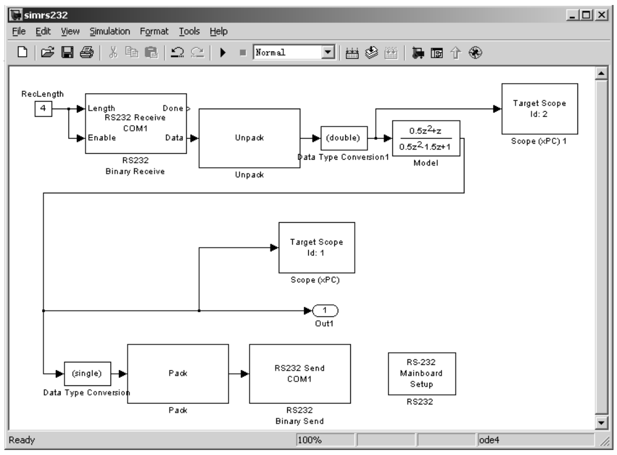


Figure 6. Simulink Diagram based on xPC Target.

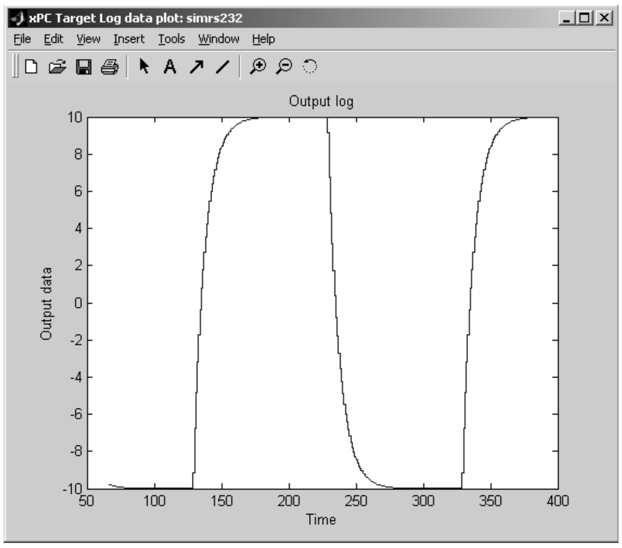


Figure 7. Output Figure.

debug and test the controller.

(1) Build the simulink model based on the xPC Target as shown in **Figure 6**.

(2) Setup the xPC working environment and create a target boot disk with the Graphical User Interface.

(3) Enter the Real-Time Workshop Parameters and Boot the Target PC.

(4) Use the Real-Time Workshop and the Visual C++ compiler to create the target application.

(5) Run the Target Application

There are three ways to control the target PC to change the target model parameters, add xPC Target scopes and to select and log signals in the target PC. The controller performance is shown in **Figure 7**.

6. Conclusions

The two methods have their own advantages and disadvantages. Although the first method is fairly flexible and can debug and test a controller using a single serial RS232 channel, it has some delay limitations. The second method is more flexible, but numerous data acquisition cards might be needed for multiple loop controllers. We have used a desktop computer to simulate the plant and for debugging and testing a controller. When the controller is put into the field, it runs very well after several control parameters were small adjusted. The methods proposed in this paper are easily realized, can reduce the cost and improve the efficiency of controller development.

7. Acknowledgements

This work was supported by China/South Africa Research Cooperation Programme (No. 78673 & CS06-L02), South African National Research Foundation Incentive Grant (No. 81705), SDUST Research Fund (No. 2010KYTD101) and Key scientific support program of Qingdao City (No. 11-2-3-51-nsh).

REFERENCES

[1] P. Xu, C. N. Huang and Y. T. Wang, "Studying A Physical Simulation System for Characteristics in Targets and Background of the Remote Sensor on Spacetorne," *Journal of System Simulation*, Vol. 15, No. 12, 2003, pp. 1763-1765.

[2] A. Davari and D. Y. Shen, "On-line Control of A Real System with Matlab/Simulink," *Proceedings of 30th Southeastern Symposium*, 1998, pp. 7-9.

[3] F. C. TENG, "Real-time Control Using Matlab Simulink," *IEEE International Conference*, 2000, pp. 2697-2706.

[4] P. S. Shiakolas and D. Piyabongkarn, "On the Development of A Real-Time Digital Control System Using Xpc-Target and A Magnetic Levitation Device," *Proceedings of the 40th IEEE Conference on Decision and Control*, 2001, pp. 1348-1351.

[5] L. Fu and H. H. Jin, "Research and Design of An Emulating Environment Based on Real-time and Testing for Embedded Software," *Computer Engineering and Applications,* Vol. 39, No. 3, 2003, pp. 117-119.