Developing a MATLAB-Based Control System Design and Analysis Tool for Enhanced Learning Environment in Control System Education

Frank S. Cheng and Lin Zhao

Industrial and Engineering Technology Department
Central Michigan University
Mount Pleasant, MI 48859

Abstract

This paper presents the development of a MATLAB-based control system design and analysis (CSDA) tool for aiding engineering students to learn feedback control system theories and design techniques. As a result, the developed CSDA tool provides users with user-friendly MATLAB graphical user interfaces (GUIs) that integrate the existing functions of Simulink and SISO Design Tool for enhanced functions in control system analysis and controller design.

1. Introduction

In control theory, a single input-single output (SISO) closed-loop feedback control system consists of five basic components. They are the input, output, controller, plant, and feedback which are connected in a structure as shown in Figure 1. Given an input, a good SISO control system is able to generate the system output that meets the desired control specifications. Usually, the time response of a unit step input to the standard second order system is used to define the control specifications. The common control specifications are the overshot (*OS* %), settling time T_s, and steady-state error e_{ss}.[1] The design and analysis of SISO feedback control systems play the critical roles in control system education. The study topics include control system modeling, system characteristics analysis, and controller function design. Teaching and learning practice in control system education has shown that the effectiveness of the study relies on the various functions of the control tools such as Simulink and SISO Design Tool.[1, 2, 3, 4]

The Simulink software is a modeling and simulation tool for control systems. The software allows users to model the control system components with transfer functions, and simulate the system's time response for the given input. However, one problem in using Simulink software is the lack of available Simulink functions in data measurement and acquisition. Often users have to develop additional MATLAB programs to have these functions for supporting their control analysis and design.

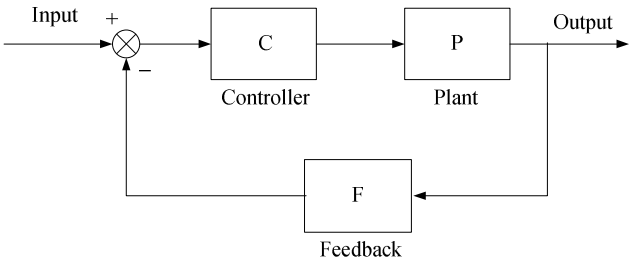


Figure 1 Block diagram of SISO closed-loop control system

The SISO Design Tool is a graphical-user interface (GUI) that allows user to use plot techniques such as root-locus and Bode diagram to design controllers for a SISO control system. To use the SISO Design Tool, however, the transfer functions of the components in the control system have to be developed using Simulink LTI System blocks. This requirement often results in inconvenience for Simulink model development, management, and analysis.

For enhancing the existing learning environment provided by Simulink and SISO Design Tool, this paper introduces the development of a MATLAB-based control system design and analysis (CSDA) tool that provides users with user graphical interfaces (GUIs). These interfaces integrate the existing functions of Simulink and SISO Design tool for improved functions in model conversion, data manipulation, system analysis and design. Figure 2 illustrates the interconnections of the developed CSDA GUIs.

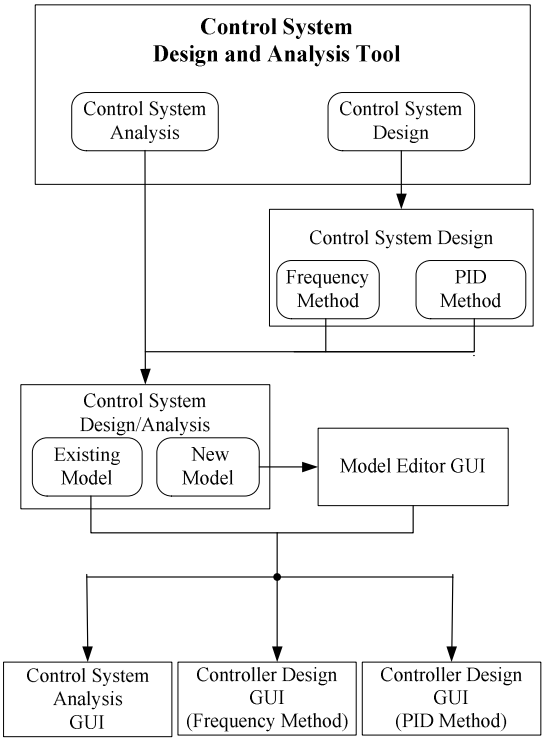
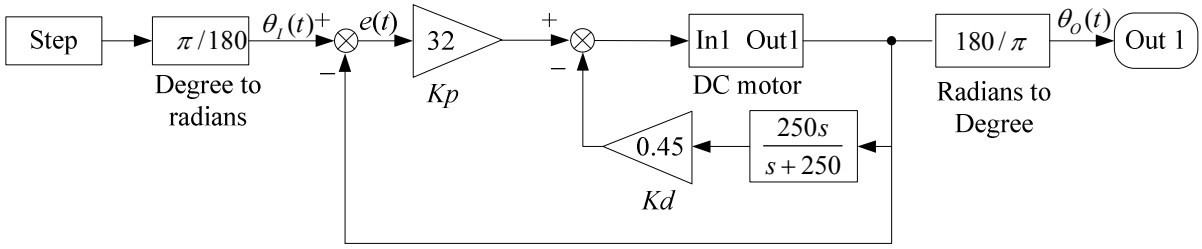


Figure 2 CSDA user graphical interfaces

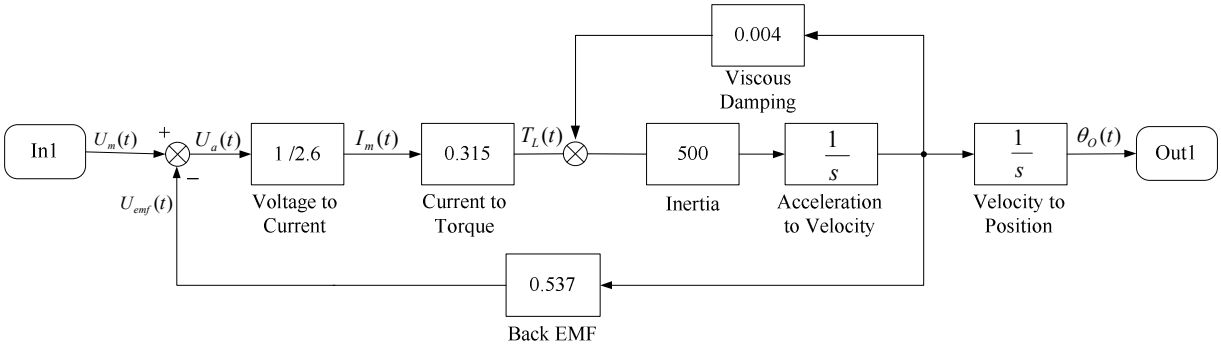
In Figure 2, the square block represents a GUI and the rounded-corner block represents a button on the GUI; the arrows indicate the transitions of the GUIs after clicking the button. Through clicking GUI buttons, users are able to manage Simulink models in CSDA tool by using the Model Editor, analyze a control model by using Control System Analysis GUIs, and design the controller by using either the Frequency Design or PID Design GUIs. In Section 2 and 3, the Simulink model types and data files used in CSDA tool are described. The functions of the CSDA GUIs are discussed in Section 4. This includes the Model Editor GUI, Control Analysis GUI, Frequency Design GUI, and PID Design GUI. The DC motor position and speed control systems are used to illustrate the applications of the GUI functions. Section 5 presents the conclusions.

2. CSDA Model Types

The CSDA tool uses three types of Simulink models in supporting control analysis and design. They are the 'normal', 'standard', and 'PID' models. A 'normal' Simulink model consists of one step input, one output block, and numerous functional blocks and sub-functional blocks. Figure 3 shows a 'normal' model developed for a DC motor position control system. In CSDA, the Control Analysis GUI uses a 'normal' model for analyzing the parameter effects to the control system output. To do that, the functional block where the parameter is to be analyzed must have a block name. For example, K_p and K_d are the names for the two gain blocks in Figure 3, respectively.



(a) A closed-loop position control system block diagram



(b) A DC motor subsystem in the closed-loop position control system

Figure 3 A 'normal' Simulink model for a DC motor position control system

The 'Standard' and 'PID' Simulink models in CSDA are two special cases of the 'normal' model. They are used by the Frequency Design GUI and PID Design GUI, respectively. Like a 'normal' model, both 'standard' and 'PID' models have one step input block and one output block. However, there are restrictions on the models' functional blocks. A 'standard' model have three functional blocks named as 'plant', 'feedback', and 'controller'. Figure 4 shows a 'standard' model developed for a DC motor speed control system. In a standard model, the transfer functions of the plant and feedback blocks must be in polynomial forms, and the controller transfer function must be in zero/pole form. The initial transfer functions of the control block can be '1'. A 'PID' model have a controller block named as 'PID controller' that has P-I-D- functions defined by gain parameters K_p, K_i, and K_d, respectively. The initial values of the PID parameters K_p, K_i, and K_d can be set as 1, 0, 0, respectively. There is no restriction on the transfer functions of the plant and feedback blocks in the PID model. Figure 5 shows an initial 'PID' model used by the current version of the PID Design GUI, where the plant is a generic first order system and the feedback is 1.

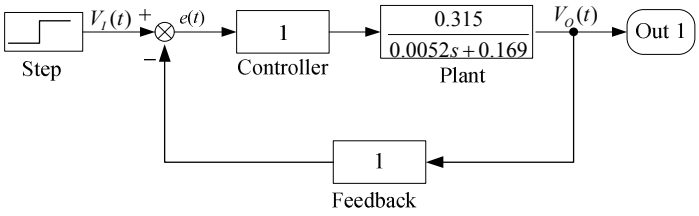


Figure 4 An initial 'standard' model of a DC motor speed control system

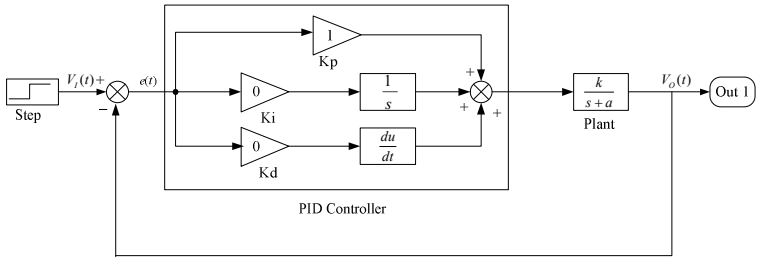


Figure 5 An initial 'PID' model for the first order system

3. CSDA Data Files

The data files are created for the data storage and exchange in the CSDA tool as shown in Figure 6. In MATLAB environment, the most important approach to importing and exporting data involves the use of MATLAB MAT-files. MAT-files provide a convenient mechanism that allows the CSDA tool to handle the model data required by different GUIs. In a CSDA data file, the required model data is handled through MATLAB array variables such as *structure* that has named "data container" called fields. The fields of a structure can contain any kind of data. The key in designing CSDA data files with structures is how to build arrays that holds the structures, and how to break up the structure fields.

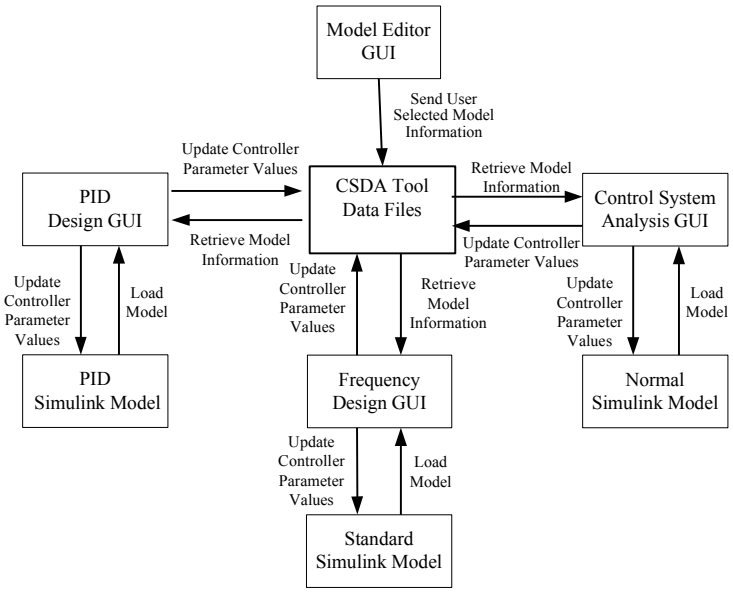


Figure 6 Data exchange in CSDA

In CSDA, a selected Sinmulink model may be identified through its model name and model type. The model name is the file name of the Simulink model, and the model type refers to the CSDA 'normal', 'stamdard', and 'PID' models. A parameter in the model is identified through the block name, parameter name, and parameter index. For a named block that has a constant number, the parameter name is the name assigned to the number. For example, for the values of 'K_p', 'K_d' and 'Inertia' blocks in the 'normal' model shown in Figure 3, the corresponding parameter names are the assigned names called 'Gain' in the model. However, for a named block that has a transfer function in polynomial form, the parameter name must be specified as 'numerator' or 'denominator'. In this case, the parameter index describes the order of the corresponding coefficient in the numerator or denominator. For example, for the plant block in the 'standard' model shown in Figure 4, the parameter_index1 and parameter_index2 in the denominator are 0.0052 and 0.169, respectively.

In the design of the CSD data files, the model name, model type and associated model parameters are organized through the fields of the created data structures as shown in Table 1.

Table 1 The Variables in CSDA Data File

Variable Name	**Type**	**Fields Name**		**Type**	**Data**
Model_ data	Structure	Model_name		String	Model name
		Model_type		Number	Model type
		Parameter_number		Number	Parameter number
		Parameter i (i =1,2,3,…)	Block_name_i	String	Block name
			Param_name_i	String	Parameter name
			Param_Index_i	Number	Parameter index

String arrays are also used in the data file for storing a group of model names used by the CSDA tool. Four data files are used to support the functions of control system analysis and design in CSDA. They are called modeldata.mat, modelparam.mat, sys1.mat, and sys2.mat. The key data file in CSDA is modeldata.mat that is created in advance with three data structures and three spring arrays. All three structures in the data file have the same fields as shown in Table 1, and are used to store the three types of the model information, respectively. The three cell arrays are used in the data file to store the model names for each model type.

4. CSDA GUIs

Creating a GUI involves two basic tasks: laying out the GUI components and programming the GUI components. In MATLAB, these two tasks can be done under the GUIDE development environment. GUIDE primarily is a set of layout tools in which the MATLAB graphical control objects called 'uicontrol' objects can be created and programmed. The typical 'unicontrol' objects used in CSDA include push button, edit text, radio button, sliders, list box, axes and figures. These 'uicontrol' objects are programmed in CSDA GUIs via their Callback properties and Callback functions to accomplish the functions of data manipulation and procedure sequencing.

4.1 Model Editor GUI

Figure 7 shows the Model Editor GUI designed for users to input new model information to CSDA data file modeldata.mat. This is done through the sequence of using the 'unicontrol' objects in the GUI. After the GUI is lunched, the GUI first enables three radio buttons in the Function frame for users to select one of three 'model manipulation' functions by activating a button. The 'Add a model' function allows users to input model data of a Simulink model to CSDA data file, the 'Edit a Model' function allows users to modify the existing model data in CSDA data file, and the 'Delete a Model' function allows users to delete the existing model data from CSDA data file.

If the 'Add a Model' function button is selected, the GUI then enables three 'model type' radio buttons for user to specify one of three model types. If the 'standard' (SISO) (or 'PID') model type is selected, the GUI enables the 'model name' edit text for users to type the model name. After the model name is entered, the corresponding Simulink model appeares in the Simulink window and the GUI enables the OK button. Clicking OK button will launch the Frequency Design GUI (or PID Design GUI) and save the input 'standard' (or 'PID') model data to CSDA data file modeldata.mat. If the 'normal' model type is selected and the model name is entered, the GUI enables the four 'parameter' edit texts in Parameter frame for users to specify up to four parameters to be studied. With the 'normal' model opened in the Simulink window, users can easily specify the selected model parameter with respect to the block name, parameter name, and parameter index. For example, Figure 7 shows three parameter values, K_p, K_d, and Inertia, of the DC motor position control system shown in Figure 3. Clicking OK button will launch Control Analysis GUI and save the input 'normal' model data to CSDA data file modeldata.mat.

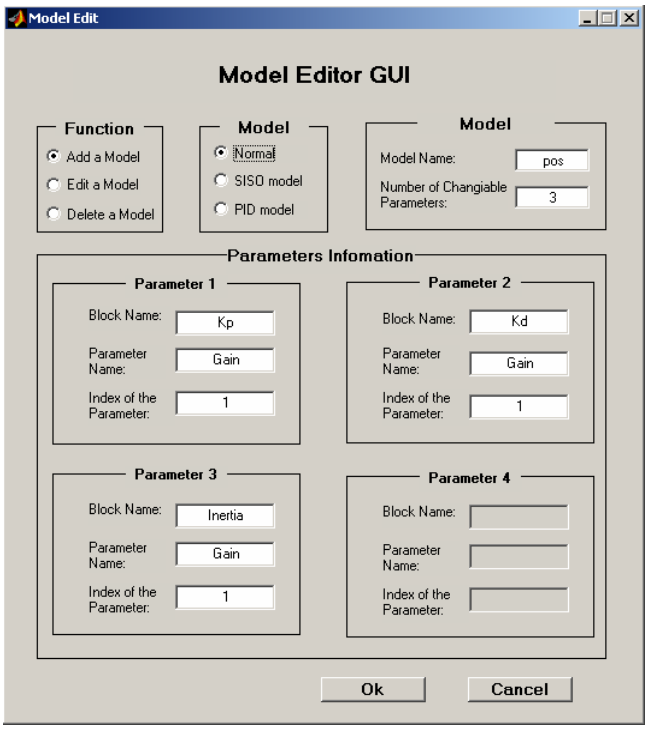


Figure 7 Model Editor GUI

If the 'Edit a Model' function button is selected, the GUI then enables the 'model name' edit text in the Model frame for users to enter the model name of an existing model in CSDA. Using the entered model name, the GUI retrieves the corresponding model type and parameters stored in data file modeldata.mat and displays them in the GUI. The selected model also appears in the Simulink Window. Users then might be able to modify the model in the Simulink Window, and input the new model type and/or new parameters on the GUI. According to the entered model type, the GUI will launch the corresponding GUI and save the input model data to data file modeldata.mat after the OK button is clicked.

4.2 Control Analysis GUI

The Control Analysis GUI as shown in Figure 8 is developed for users to analyze the parameters' effects to control system output. Users first select a 'normal' model existed in CSDA by using the popup manual 'Model Selection.' As the model is selected, the corresponding input parameters stored in data file modeldata.mat are displayed on the GUI's sliders. For example, the three sliders in Figure 8 show the current values of three parameters K_p, K_d, and Inertia in the DC motor position control model shown in Figure 3. The sliders in the GUI are designed to accept numeric input within a specific range and with required resolution. Prior to enabling the slider function for parameter adjustment, the GUI asks users to select either 'Display' or 'Record' radio button. If the 'Display' button is selected, the current values of sliders are to be displayed on the 'Result List' of the GUI without writing them into CSDA data file. However, when the 'Record' button is active, the current parameter values set by the sliders are to be saved in data file

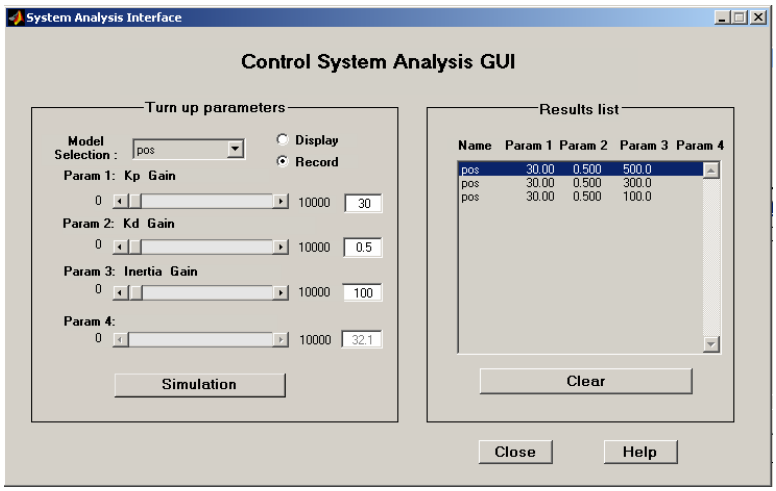


Figure 8 Control Analysis GUI

modeldata.mat, as well as in the Simulink model as the final analysis results. If the 'Display' button is selected, the GUI enables the 'Simulation' button. Clicking the button, the sliders are enabled and their values are loaded to the Simulink model for performing model simulation. The axis and figure 'uicontrol' objects in the GUI are designed to dynamically measure and display the time response of the system output as the parameters on the sliders are changed. Figure 9 and Table 2 show three different system responses for three different inertia valves in the DC motor position control system.

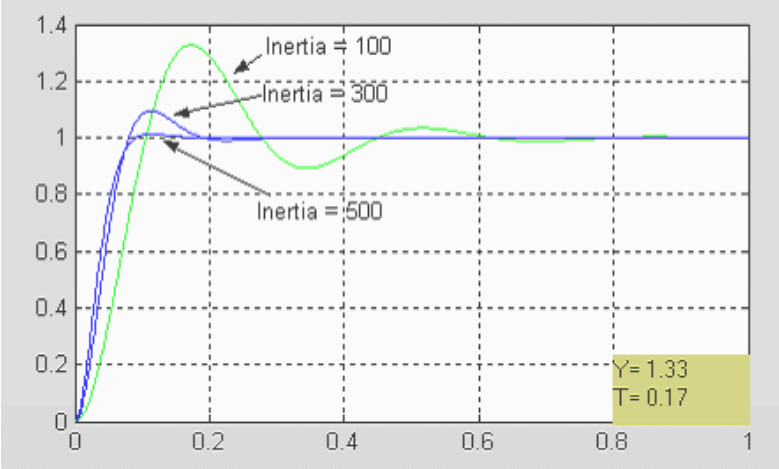


Figure 9 The effect of motor inertia in position control system

Table 2 The measured time response of the position control system in different motor inertia

Inertia ($1/kg \cdot m^2$)	Overshoot (OS %)	Settling Time (sec.)
100	32	0.58
300	9	0.22
500	2	0.11

Note: K_p=32, K_d=0.5

4.3 Frequency Design GUI

The Frequency Design GUI shown in Figure 10 allows users to conduct controller design through the frequency-response design procedures. In the GUI, users first select an existing 'standard' model from the popup manual, and then input steady-state error e_{ss}, overshot (SO %), and settling time T_s using the edit texts in the Time Response frame. As the user clicks the 'Finish Input' button, the GUI converts the input specifications into the values of phase margin Φ_M and bandwidth ω_{BW}, and displays the results in the edit texts of the Frequency Response frame. The relationships used in the conversion are derived based upon the second order closed-loop control system.[1] They are:

$$\xi = \frac{-\ln(OS\%/100)}{\sqrt{\pi^2 + \ln^2(OS\%/100)}}$$

$$\Phi_M = \arctan\left(\frac{2\xi}{\sqrt{-2\xi^2 + \sqrt{1+4\xi^4}}}\right)$$

$$\omega_{BW} = \frac{4}{T_s\xi}\sqrt{(1-2\xi^2) + \sqrt{4\xi^4 - 4\xi^2 + 2}}$$

where ξ is the system's damping ratio. Figure 10 shows the results of the control specification conversion for the DC motor speed control system in Figure 4. With the frequency response specifications available, users are able to lunch MATLAB SISO Design Tool by pressing

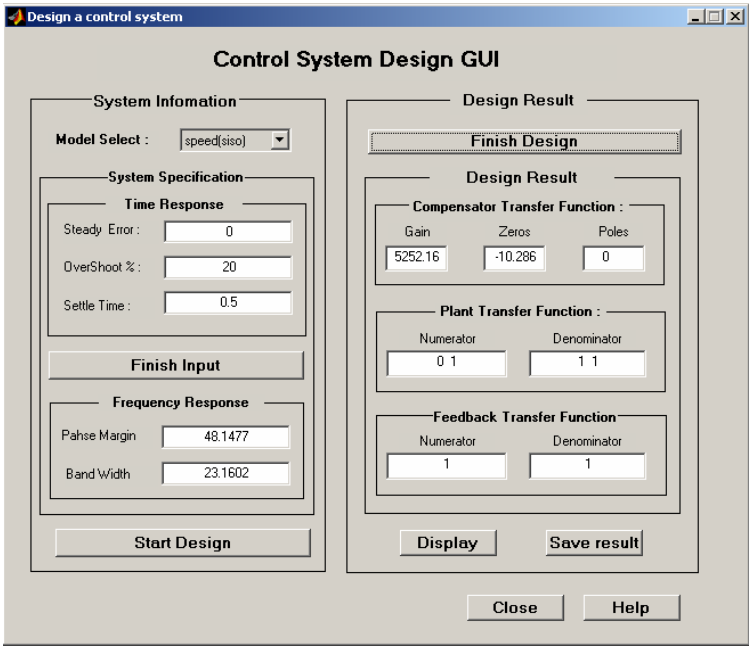


Figure 10 Frequency Design GUI

the 'Start Design' button on the GUI. During the design with SISO Design Tool, users are able to observe the system open-loop Bode plot while placing designed zeros and poles. The SISO Design Tool also displays the values of the synthesized zeros and poles as the controller design solution. Figure 12 shows the Bode plot where the synthesized zero and pole values constitute the controller design solution for the DC motor speed control system in Figure 4. After users close the SISO Design Tool, the Frequency Design GUI automatically displays and updates the design results in the GUI, CSDA data files, and Simulink model as shown in Figure 10 and Figure 11, respectively. The 'Display' button allows users to display the system time response. For the designed DC motor speed control system in Figure 11, the time response shown in Figure 13 shows that the system time response meets the given control specifications shown in Figure 10.

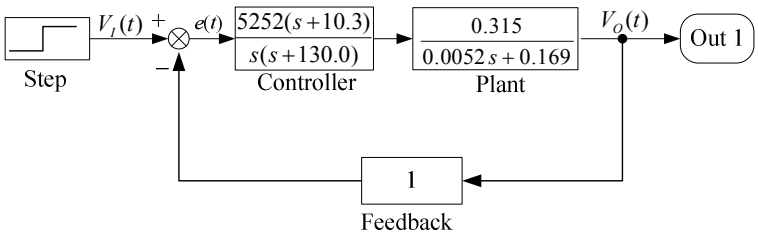


Figure 11 The solution model of DC motor speed control system after the frequency design

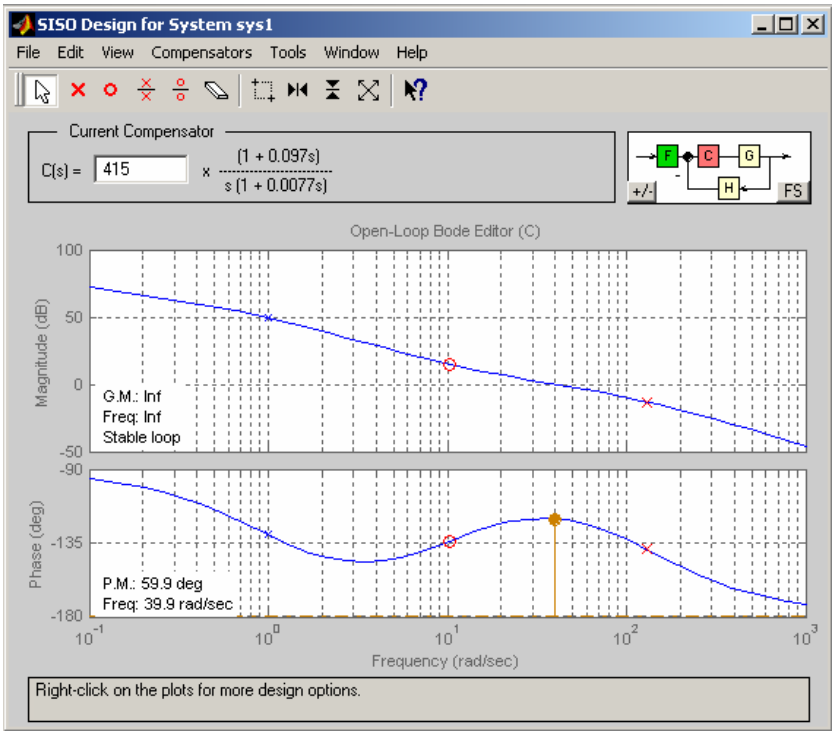


Figure 12 The Bode plot of the closed-loop DC motor speed control system

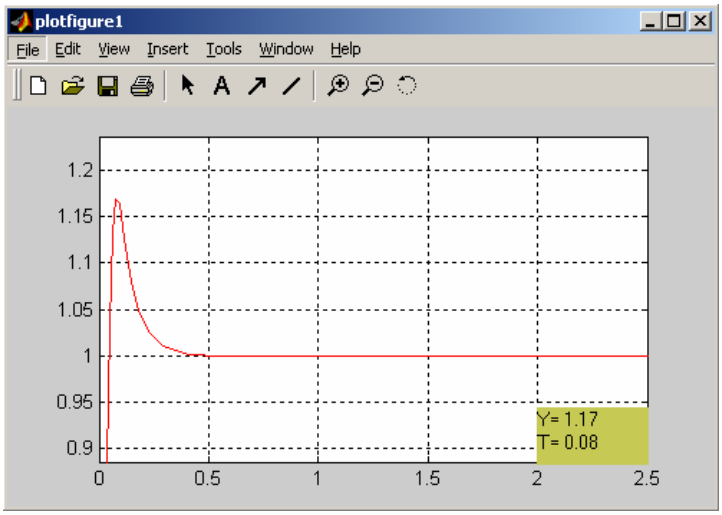


Figure 13 The step time-response of designed DC motor speed control system

4.4 PID Design GUI

PID controllers are widely used in SISO feedback control systems. This is because the PID functions can be used to improve both the transient response and the steady-state response of a SISO control system. The 'PID tune-up' is a practical approach for PID controller design solutions. This is done through manually or automatically adjusting the PID parameters in the control system in order for the system to meet the required control specifications. However, in this approach, users must be able to measure the system's output performance as adjusting PID parameter, and the user's PID design experience plays a critical role for finding a feasible set of PID parameters. To aid users to conduct the 'PID tune-up' for a SISO control system, the PID Design GUI is developed as shown in Figure 14, where users are able to perform the 'manual' and 'auto' tune-up functions for a 'PID' model.

4.4.1 Manual PID Tune-Up Function

Under the 'manual' tune-up function, the plant block in the PID model can be any Simulink blocks or subsystems representing any order control system. In the GUI, users are able to adjust PID parameters K_p, K_i, and K_d through three sliders, and observe the change of the system output displayed in the built-in time response window. For each output response, the values of the overshot (OS %), settling time T_s, and steady-state error e_{ss} are automatically measured and displayed for users to compare them with the expected specifications. With such a dynamic interaction between the controller's parameters and the system's response, users are able to quickly find a set of PID parameter values that allows the system to generate required output.

4.4.2 Auto PID Tune-Up Function

Under the 'auto' tune-up function, the GUI is able to automatically determine the values of PID parameters for the given system to meet the specified control specifications. In the current version of the PID GUI, the 'auto' function is developed to tune up only the K_i and K_p

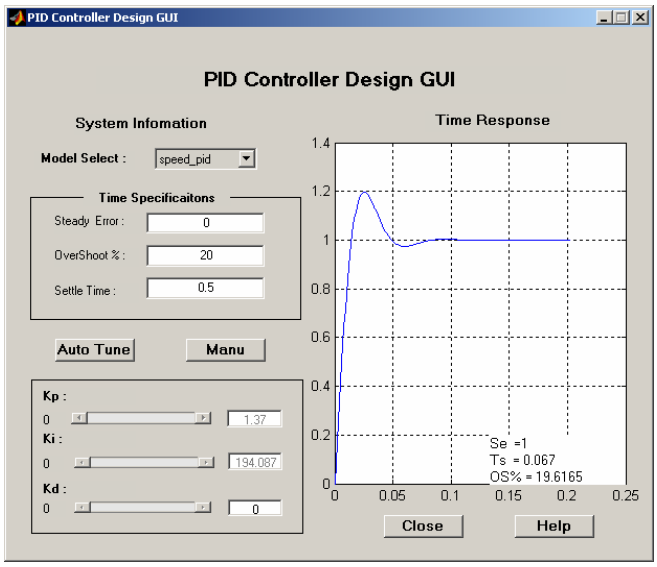


Figure14 PID Design GUI

parameters for the 'PID' model that has the first order plant as shown in Figure 5. In this case, the 'auto' function works by first estimating the values of two parameters, *a* and *K*, in the first-order plant block according to the given expected control specifications. The analytical solutions of the first order system show that *a* and *K* value can be determined as:

$$K = \frac{4e_{ss}}{T_s} \quad \text{and} \quad a = K\left(\frac{1-e_{ss}}{e_{ss}}\right)$$

where T_s is settling time and e_{ss} is the steady-state error. With the estimated plant parameters K and a, the 'auto' function is able to determine the initial values of K_i and K_p prior to the adjustment, and then use an established strategy to adjust the K_i and K_p according to the actual system performance. In the GUI, this is achieved through applying the analytical solutions of second order systems.

For the system that has a first order plant and a PI controller as shown in Figure 5, the closed-loop system transfer function of the system is a second order system, and the corresponding open-loop transfer function is:

$$G_o = \frac{KK_p\left(s + \frac{K_i}{K_p}\right)}{s(s+a)}$$

where pole *a* and zero K_i/K_p are brought by the plant and controller, respectively. The 'auto' tune-up function uses two facts about the pole and zero in the open-loop transfer function, which are indicated in the open loop Bode plot.

The first fact used by the 'auto' function is that if the controller zero K_i/K_p is far away from the plant pole a, the open-loop transfer function can be approximated as:

$$G_o = \frac{KK_p}{s(s+a)}$$

In this case, the system closed-loop transfer function becomes a standard second order system. The analytical solutions show that the step response of a standard second order system is a function of ξ and ω_n; and the settling time T_s can be determined as $k/(\xi\,\omega_n)$, where k represents percentage of steady error e_{ss}. In design of such a system, only one characteristic of the system ξ and ω_n, or a single function of the two can be set.[1] Based upon these solutions, the 'auto' function in the GUI establishes the single function of ξ and ω_n as:

$$\omega_c = \xi\,\omega_n = 3/T_s$$

where ω_c is the frequency at which the magnitude of the open-loop frequency response equals 1. This relationship allows the 'auto' function to express K_p as a function of T_s:

$$K_p = \frac{\sqrt{\left(\frac{1}{T_s}\right)^4 + a^2\left(\frac{1}{T_s}\right)^2}}{K}$$

The 'auto' function uses this equation to determine the initial value of K_p prior to the parameter adjustment.

The second fact used by the 'auto' function is that the phase characteristic of the open-loop transfer function remains unchanged if zero K_i/K_p remains a constant value; and the increment of K_p increases the system bandwidth, resulting in increasing the speed of system response.[1] Using this fact, the 'auto' function establishes the strategy for adjusting K_i and K_p. The strategy is to keep a constant zero value while increasing K_p 20 percent each time until the actual system specifications meet the expected system specifications. The 'auto' function in the GUI sets the constant zero value as:

$$K_i/K_p = 3a$$

where a is the estimated plant pole. The PID GUI 'auto' function was tested with the DC motor speed control system as shown in Figure 14. The results show that the developed 'auto' tune-up method works for the 'PID' model that has a first order plant.

5. Conclusion

The development of the CSDA tool is a very useful practice in MATLAB programming and GUI design. The test results show that the developed CSDA tool successfully integrates Simulink and SISO Design Tool though CSDA model types, data files, and GUI functions. With the CSDA

tool, users are able to use SISO Simulink models to analyze model's parameter effects to the system output, and design the controller's functions based upon the given control system specifications. The GUI design method used in this project also demonstrates the feasibilities for implementing other techniques in SISO control study using MATLAB-based programming and tools.

Bibliography

1. Phillips, C. L., Feedback Control Systems, 4th ed. Prentice Hall, 2000.
2. The Mathworks, Manual of Simulink, Version 5, 2002.
3. The Mathworks, Manual of Control Tool Box, 2002.
4. The Mathworks, Manual of MATLAB, 2002

Biography

FRANK S. CHENG is an associate professor in the Department of Industrial and Engineering Technology at Central Michigan University. He received a M.S. degree and a Ph.D. degree from the University of Cincinnati. Dr. Cheng has taught courses in robotics and industrial automation. His research interests include robotics, control systems, and automated manufacturing. He is a member of the American Society for Engineering Education (ASEE) and the Society of Manufacturing Engineers (SME).

LIN ZHAO received a B.S. degree in 1997 and a M.S. degree in 2000 from Haerbin Engineering University, Haerbin, China. He is currently a graduate student at Central Michigan University. His research interests include control systems, robotics, and automation.