

# Open Source Ch Control System Toolkit and Web-Based Control System Design for Teaching Automatic Control of Linear Time-Invariant Systems

BO CHEN,<sup>1,2</sup> YU-CHENG CHOU,<sup>3</sup> HARRY H. CHENG<sup>4</sup>

<sup>1</sup>Department of Mechanical Engineering – Engineering Mechanics, Michigan Technological University, Houghton, Michigan

<sup>2</sup>Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, Michigan

<sup>3</sup>Department of Mechanical Engineering, Chung Yuan Christian University, Chungli, Taoyuan 32023, Taiwan

<sup>4</sup>Integration Engineering Laboratory, Department of Mechanical and Aerospace Engineering, University of California at Davis, Davis, California 95616

Received 4 August 2009; accepted 3 May 2010

**ABSTRACT:** The Ch Control System Toolkit (CCST) is a software package for the design and analysis of control systems. It is a C/C++ class with member functions for solving control problems in a user-friendly C/C++ interpreter, Ch. Based on the CCST, a Web-based Control System Design and Analysis System (WCSDAS), and a Web-based Controller/Compensator Design Module (WCCDM), have been developed. In this article, using the CCST, WCSDAS, and WCCDM for teaching automatic control of linear time-invariant systems is presented. With the CCST, students are able to solve control problems with only a few lines of C/C++ code. The CCST can also be used to develop various interactive utility programs that will assist students in learning control systems without any programming requirements. With the Web-based WCSDAS and WCCDM tools, students can interactively design and analyze control systems via a Web browser. The CCST, WCSDAS, and WCCDM are open source software packages. These software tools have been used for teaching undergraduate control courses at the University of California, Davis and Michigan Technological University. © 2010 Wiley Periodicals, Inc. *Comput Appl Eng Educ*; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.20454

**Keywords:** control systems; Web-based education

## INTRODUCTION

Automatic control has become a major field in almost every engineering subject, and its courses are part of the respective engineering curricula [1]. Due to the advances of mathematics and computing technology, the modern control engineering design and analysis methods have greatly expanded the range of problems that can be solved. The increasingly computational intensive methodologies call for the development of innovative teaching ware and laboratory experiments to improve the effectiveness

of control education. A number of software packages, such as MATLAB Control System Toolbox ([www.mathworks.com/products/control](http://www.mathworks.com/products/control)) and Mathematica's Control System Professional ([www.wolfram.com/products/applications/control/index.html](http://www.wolfram.com/products/applications/control/index.html)), have been developed and made commercially available for the purposes of computer-aided control system design and analysis.

In the past few decades, many interactive tools have also been developed for teaching and learning automatic control. The use of interactive tools would reinforce active participation of students in learning by directly manipulating graphical representations of systems and getting instant feedback on the effects [2,3]. Experiments have shown that a high degree of interactivity can speed up learning and obtain an intuitive feeling for control concepts

Correspondence to: B. Chen (bochen@mtu.edu).  
© 2010 Wiley Periodicals, Inc.

through dynamic visualization. The idea of “dynamic pictures” was introduced in [4]. Dynamic pictures are based on graphical user interface and built up as interactive modules that can be manipulated through a mouse. If changes are made in a dynamic picture, an immediate recalculation and presentation automatically begins. Also, each dynamic picture is tailored to illustrate a specific concept. This novel feature can be found in many educative tools, such as ICTools [3,5] and a suite containing PID-Basics, PID-Loop-Shaping, and PID-Windup [6,7]. Easy Java Simulations (Ejs), a Java-based tool that helps create interactive dynamic simulations, was introduced in [2]. Ejs can be used on its own to generate standalone Java applications or applets, or in conjunction with MATLAB/Simulink that acts the internal engine to describe and solve system models. A Sysquake-based tool was introduced in [8] for nonlinear control systems. The tool can help students understand fundamental concepts in nonlinear control, such as the behavior of piecewise linear systems, stable and unstable limit cycles and the describing function method. A two-dimensional time-domain field simulation software tool called MEFiSTo-2D Classic [9] was developed to illustrate fundamental electromagnetic concepts that are traditionally described by mathematical formulas. MEFiSTo-2D Classic transforms abstract electromagnetic concepts into realistic images on the screen. A LabVIEW-based tool was introduced in [10] to offer a set of comfortable, ready-to-use solutions for plant identification, loop-shaping, and LQR (Linear-Quadratic-Regulator)/LQG (Linear-Quadratic-Gaussian) controller design and implementation.

As the World Wide Web (WWW) widely spreads, Web-based control system design and analysis tools and virtual laboratories are emerging as a promising technology that could greatly improve the teaching and student learning of the control systems. These tools make students more actively involved in control courses and are effective for distance learning [11,12]. The Web-based interactive computing tools allow students to try out different solutions and explore new design strategies easily by observing results that is instantly generated on the Web. Web-based laboratories can be divided into two categories: virtual and remote. A virtual laboratory allows clients to continuously access a simulation process in a remote server. The simulation engine in the server could be MATLAB or any other control toolkit. A remote laboratory offers a physical experimental apparatus to remote users through the Internet. Most existing Web-based control laboratories use MATLAB as a computational engine.

Kypuros and Connolly [13] developed virtual systems for a System Dynamics and Controls course. Each virtual system is a user-configurable numerical simulation of a physical system and can generate animations for dynamic responses of that physical system. The virtual systems are used to aid students’ visualization and enhance students’ in-lab experience. Two virtual Web-based laboratories for the controller design experiments in a Control Systems course are presented in [14]. The first virtual laboratory provides predefined simulation problems to support computer-aided controller design. The second virtual laboratory allows students to create their controller design experiments by writing MATLAB programs to be executed via the MATLAB Web Server (MWS). A student survey showed that the reliability of the MWS needs to be improved. Students preferred the second unstructured MATLAB-like environment. A Java LAPACK-based virtual system for teaching an introductory Control Engineering course is presented in [15]. The system provides basic functions for time domain and frequency domain response analysis, stability, and PID controller and compensator design. A Java and MATLAB-based

remote laboratory for an automatic control course is presented in [16]. The system was used as a complementary activity to students’ on-site laboratory work. Compared to students who do not use the environment, students who use the environment have better analytical skills during the design and tuning phases of a control system. A Java-based system, acting as both virtual and remote laboratories, is presented in [17] for simulation and remote operation of Controller Area Network (CAN) devices.

Most of the developed virtual and remote laboratories are based on commercial computational engines, such as MATLAB. These Web systems need an additional application server program to connect the Web interface and the computational engine. In this paper, an open source Ch Control System Toolkit (CCST), Web-based Control System Design and Analysis System (WCSDAS), and Web-based Controller/Compensator Design Module (WCCDM), for teaching automatic control systems are presented. The CCST provides basic building blocks to model, analyze, and design control systems. Implemented in C/C++, it offers a unique feature for smoothly integrating with real-time control software. The open source nature of these three software packages allows instructors and students to develop new simulations and experiments as needed. In addition, students can read the source code and understand how the control principles and algorithms are implemented. Developed in the Ch environment, the Web-based applications based on the CCST can be easily implemented using the Common Gateway Interface (CGI) supported in Ch.

The rest of the paper is organized as follows. The second section introduces the CCST. The third section describes a WCSDAS. The fourth section presents an extension of the WCSDAS. The fifth section presents the integration of the CCST and WCSDAS into teaching and learning for automatic control courses. Examples are given in the fifth section to demonstrate utilizing the presented software tools to solve different kinds of control problems. An extension of the WCSDAS, a WCCDM for interactive controller design via the root locus, is also presented in the fifth section. The effectiveness of using these software packages for teaching and learning automatic control systems is assessed and the evaluation results are discussed in the sixth section. Finally, the conclusions are made in the seventh section.

## CH CONTROL SYSTEM TOOLKIT

The CCST ([www.softintegration.com/products/toolkit/control](http://www.softintegration.com/products/toolkit/control)) [18] is an object-based software package. It is developed in a C/C++ interpreter – Ch ([www.softintegration.com](http://www.softintegration.com)) [19–21]. The CSST provides a C/C++ control class with member functions for the modeling, analysis, and design of linear time-invariant (LTI) control systems. The CCST provides commonly used functions in control systems design and analysis, such as time-domain response, frequency-domain response, system analysis, system design, model conversion, and system conversion. Most functions can be applied to both continuous- and discrete-time LTI systems, which are modeled in single-input, single-output (SISO) and multi-input, multi-output (MIMO) state-space equations, transfer functions, or zero-pole-gain representations. The functional modules of the CCST are outlined in Table 1. The CCST has been widely used in the industry for solving practical engineering problems and in the academia for instructional improvement.

Traditionally, the control system design and simulation are often done in a mathematically rich computing environment such as MATLAB and Mathematica. Code written in these environ-

**Table 1** Functional Modules of the Ch Control System Toolkit

Create LTI models	Model data extraction
Model conversion	System conversion (continuous—discrete)
System interconnections	System gain and dynamics
Time domain analysis	Frequency domain analysis
Root locus	Controllability and observability
Pole placement design	Linear-quadratic regulator design
Equation solvers	Plot customization

ments, however, is not directly usable in real-time control systems. Although most of these environments provide translators to convert the control code into a language useful for the real-time control (most likely, the C language), the reproduction of the program code can be error prone. Besides, the computer generated C code may not be well organized and understandable under some circumstances. Therefore, it can be difficult to maintain the generated C code and integrate it into a large-scale software system.

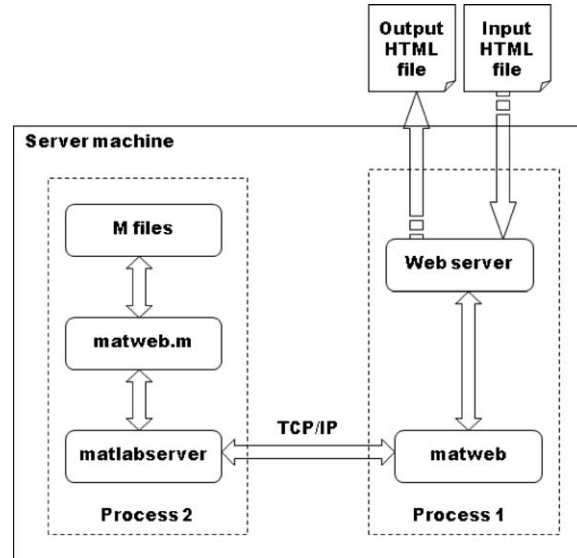
There are some other MATLAB-like interpretive programming languages, such as Sysquake and Scilab, which provide the same group of commands for control tasks as those found in MATLAB. Both Sysquake and Scilab are software packages designed for numerical computation and scientific visualization. The major purpose of these two software packages is to offer facilities for interactive calculations and visualizing the data and computation results. Integrating with Apache, Sysquake Remote enables a server to send the computation results to a client in text or graphics embedded in HTML files.

The CCST is developed in a C/C++ interpreter, with extensions of numerical and matrix computation and 2D/3D plotting. Implemented in C/C++, the CCST can seamlessly interface with other C/C++ programs, such as C code for accessing memory using pointers and input/output control in real-time embedded systems. The characteristic of the CCST enables users to do the modeling, analysis, design, and real-time control all in one language, which eliminates bugs introduced through the code translation. Moreover, a large body of existing C/C++ libraries, such as graphical user interface in X11/Motif, Windows, GTK and 3D graphics in OpenGL, computer vision in OpenCV, data acquisition in NI-DAQ and motion control in NI-Motion, is readily available for application development.

The CCST offers most features found in the MATLAB Control System Toolbox. A detailed syntax comparison between the CCST and the MATLAB Control System Toolbox can be found at [www.softintegration.com/products/toolkit/control/ch\\_matlab.html](http://www.softintegration.com/products/toolkit/control/ch_matlab.html). Multiple sample programs using both approaches for solving same problems are also compared on the Web at [www.softintegration.com/demos/toolkit/control](http://www.softintegration.com/demos/toolkit/control). Since the CCST is developed on top of Ch and the Common Gateway Interface (CGI) is supported in Ch ([www.softintegration.com/products/toolkit/cgi](http://www.softintegration.com/products/toolkit/cgi)), Web-based control systems/tools such as the WCSDAS introduced in the next section can be developed straightforwardly by integrating the CCST, Ch, and Ch CGI.

## WEB-BASED CONTROL SYSTEM DESIGN AND ANALYSIS SYSTEM

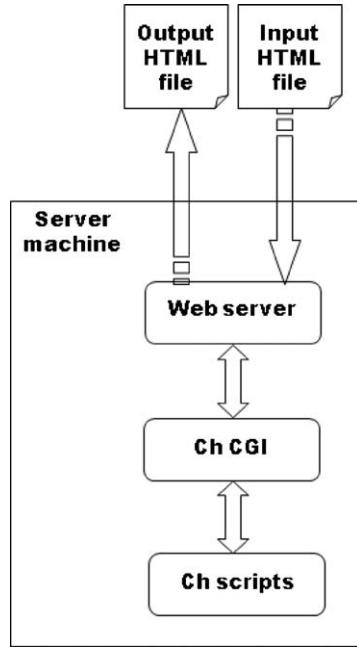
The WWW is a natural candidate to implement software tools that allow students to interact with the subject matter anytime anywhere. The WWW is platform and location independent, and it is

**Figure 1** Web-based systems using MATLAB as the computational engine.

effective for distance learning where an Internet connection is all that is needed. Based on the CCST, we have developed a WCSDAS [22]. The main idea of designing such a Web-based tool is to use the WWW as a communication infrastructure to allow multiple users to access Ch-based computational toolkits such as the CCST. In the WCSDAS, a Web browser provides an environment for accepting users' input and transmitting users' input to a Web server. The Web server supports the execution of CCST functions and interfaces with clients through CGI and HTML files. The computation time is not hindered by the network speed between the server and client machines because the computation is performed purely on the server machine. The WCSDAS provides students with a great opportunity to learn control theories and prototype control systems on the Web at any time and location they desire.

From the development point of view, using the CCST to implement a Web-based control system is simpler than the MATLAB Control Toolbox. For a MATLAB-based Web system, an additional application server program, matlabserver, is needed for the communication between the Web server process and the MATLAB process, as shown in Figure 1. The major components of a MATLAB Web server include: matweb, matlabserver, and matweb.m. The matlabserver manages the communication between the Web application and MATLAB. It invokes matweb.m, which in turn runs the M-files to perform necessary computation and generate an output HTML document. The matweb is a TCP/IP client of the matlabserver. It uses CGI to extract data from an input HTML document and transfer these data to the matlabserver through TCP/IP communication. The matlabserver and matweb can locate within a single computer (but different processes) or separate computers.

In contrast to a MATLAB-based Web system, Ch CGI can directly communicate with Ch scripts as shown in Figure 2. There is only one process. Ch CGI acts as an interface between a Web server and Ch scripts. Ch scripts extract data from an input HTML document, use the extracted data to perform calculations, and create an output HTML document with the computation results.



**Figure 2** Web-based systems such as the WCSDAS using Ch as the computational engine.

#### AN EXTENSION OF WEB-BASED CONTROL SYSTEM DESIGN AND ANALYSIS SYSTEM

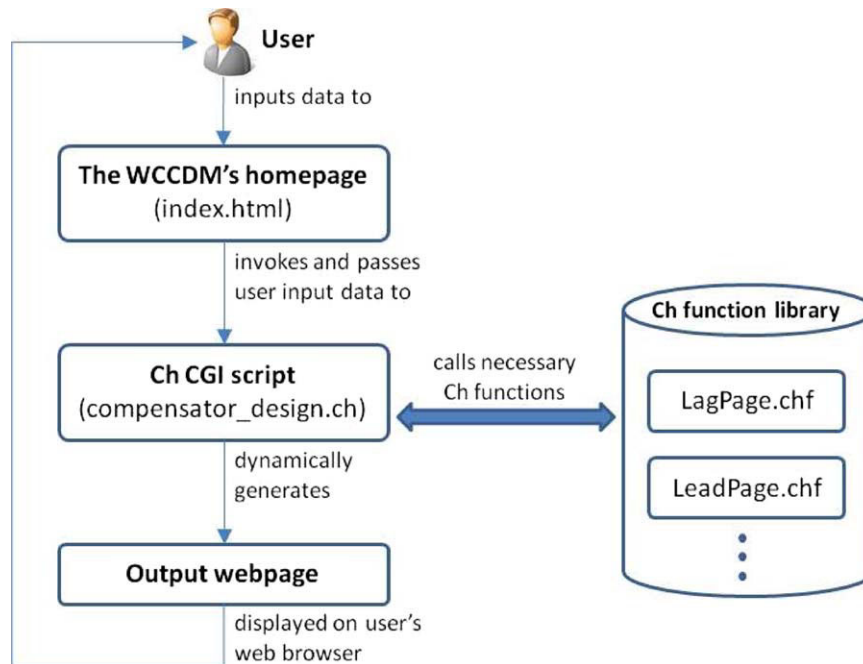
Since the CCST, WCSDAS, and Ch CGI are all C-based, open source, and open architecture software packages, all the member functions of the CCST and all the internal functions and programs of the WCSDAS can be reused or modified to develop new Web-based tools. These augmented Web-based tools play a role

of extending and complementing the functionalities of the WCSDAS. Therefore, they not only can extend the functionalities of the WCSDAS to cover a variety of different aspects in the automatic control field, but also can address the difficulties that students may encounter in learning the design and analysis of control systems.

In this paper, a WCCDM is used as an example to demonstrate the extensibility of the WCSDAS. With the WCCDM, the performance change of a control system due to the variation of system parameters is observed immediately on the Web. The WCCDM helps students relate the system performance with the parameters or variables being manipulated. The implementation of the WCCDM is straightforward by following the design pattern of the WCSDAS. In addition, a number of internal functions and some of the source code of the WCSDAS as well as some member functions of the CCST are used in the implementation of the WCCDM. Using the WCCDM to solve a sample compensator design problem will be presented in detail in a later section. The WCCDM is available for downloading at [iel.ucdavis.edu/course/EME172](http://iel.ucdavis.edu/course/EME172).

The file structure of the WCCDM is shown in Figure 3. A user selects option for Lag compensator design or Lead compensator design on the WCCDM's homepage, named index.html, through a Web browser. When the user clicks the "Continue" button on the index page, a Ch CGI script named compensator\_design.ch is automatically invoked, and the user's selection is passed to the script. The Ch CGI script calls corresponding Ch functions based on the user's selection. These Ch functions generate new web pages to request additional information from the user, perform numerical computation, make plots, and dynamically generate a final output webpage to be displayed on the user's Web browser. All the Ch functions for the WCCDM are located in a Ch function library. Each Ch function is implemented in a separate file that has the same name as the function and a filename extension .chf, such as LagPage.chf and LeadPage.chf shown in Figure 3.

A detailed explanation of the first few Web pages of the WCCDM is presented below. The most important lines in the



**Figure 3** The file structure of the WCCDM.

source code of the WCCDM's index page are shown below

```
<form method="post" action="/cgi-bin/projects/course/control/compensate/compensator_design.ch">
<h3><u>Compensation type: </u></h3>
<input type="radio" name="CompType" value="Lag">Lag compensation <br>
<input type="radio" name="CompType" checked value="Lead">Lead compensation <br>
<br>
<input type="submit" value="Continue">
</form>
```

These lines of code handle the selection of the type of compensator to be designed. When this HTML form is submitted, the Ch CGI script `compensator_design.ch` will be called to drive the rest of the interaction between the Web server and the user's Web browser. The Ch CGI script invoked is specified by the action attribute. Two radio buttons are created for selecting the type of compensator to be designed. Depending on which radio button is selected, the browser will pass either the value "Lag" or "Lead" to the Ch CGI script `compensator_design.ch`. Program 1 shows a simplified version of the `compensator_design.ch`.

```
#include <stdio.h>
#include "compensate.h"
int main()
{
    int num, i;
    chstrarray name, value;
    num = Request.getFormNameValue(name, value);
    Response.setContentType(contenttype);
    Response.begin();
    printf("<head>\n");
    printf("<title>%s</title>\n", title);
    printf("<style>#eqn {position:relative;top:8px;}</style></head>\n");
    if(strcmp(value[0], "Lag") == 0)
    {
        LagPage();
    }
    else if(strcmp(value[0], "Lead") == 0)
    {
        LeadPage();
    }
    Response.end();
    return 0;
}
```

**Program 1. A simplified Ch CGI script for the WCCDM.**

The header file `compensate.h` contains different kinds of information, such as declarations of variables, macros, and functions, and inclusions of other header files that are needed to compose `compensator_design.ch`. The variables "name" and "value" are of type `chstrarray`. This data type is defined in the Ch CGI package and denotes an array of strings. These variables will be used to hold the name and value retrieved from the aforementioned HTML form using the following statement:

```
num=Request.getFormNameValue(name, value);
```

where "Request" is an instance of the `CRequest` class that is defined in the Ch CGI package, and "getFormNameValue()" is a member function of the `CRequest` class. The script sets the content type of the Web page to be created and begins the CGI response with the following statements:

```
Response.setContentType(contenttype);
Response.begin();
```

where "contenttype" is a string defined in `compensate.h`, "Response" is an instance of the `CResponse` class that is defined in the Ch CGI package, and "setContentType()" and "begin()" are member functions of the `CResponse` class. The script then prints a rudimentary html header with three "printf()" statements. Afterward, the program flow is diverted based on the selected compensator's type by the following statements:

```

if(strcmp(value[0], "Lag") == 0)
{
    LagPage();
}
else if(strcmp(value[0], "Lead") == 0)
{
    LeadPage();
}

```

where "LagPage()" and "LeadPage()" are Ch functions that create the Web pages for lag compensator design and lead compensator design, respectively. Finally, the script ends the CGI response by the following statement:

```
Response.end();
```

where "end()" is a member function of the CResponse class.

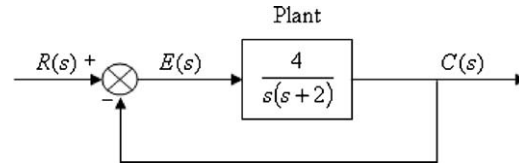
### INTEGRATING CH CONTROL SYSTEM TOOLKIT AND WEB-BASED CONTROL SYSTEM DESIGN AND ANALYSIS SYSTEM INTO TEACHING AND LEARNING FOR AUTOMATIC CONTROL COURSES

The CCST, WCSDAS, and WCCDM were integrated into the teaching and learning for an undergraduate course "Automatic Control of Engineering Systems" at the University of California, Davis. The interactive nature of these software packages allows students to have a better understanding of lecture and discussion examples through visualizing the immediate effects of parameter changes on the system performance. These three software tools were used for lecture and discussion session examples, homework, and compensator design problems. In the beginning of the summer session, basic functionalities of the tools and sample programs were introduced in the discussion sessions. The homework assignments were graded weekly. Commonly made errors related to the programming were discussed during the discussion sessions. In addition, two lab sessions per week were used to help students solve control problems using the CCST, WCSDAS, and WCCDM. The students were also encouraged to ask questions via emails and during office hours. Solutions for the homework and design problems were posted on the course website after the assignments were returned to the students. The close interactions among the instructor, teaching assistant, and students allowed the instructor to experience the students' attitudes toward the software and give feedback to the students promptly.

All the software packages and supplementary documents, lecture and discussion presentation materials, and homework solutions using these tools are available for downloading at [iel.ucdavis.edu/course/EME172](http://iel.ucdavis.edu/course/EME172). Following sections give several examples of using the CCST, WCSDAS, and WCCDM for control systems teaching and learning.

#### Use of Ch Control System Toolkit for Automatic Control Courses

For students with programming experiences, the CCST is an easy-to-use toolkit for writing programs to solve various problems associated with the design and analysis of control systems. To help students without programming skill, we provide sam-



**Figure 4** A unity feedback system with a plant having a transfer function  $= 4/s(s+2)$ .

ple programs to illustrate the usage of each function in the CCST package. These sample programs greatly help students use the CCST package. As mentioned in the second section, the CCST provides high-level control system design and analysis functions, similar to functions offered by the MATLAB Control System Toolbox. For an illustrative purpose, the following example, a simplified homework problem where students are asked to generate the Bode plot of a given plant, is used to compare programs written with the CCST and MATLAB Control System Toolbox.

*Example 1:* Bode plot of the plant in a unity feedback system is shown in Figure 4.

Both of the Ch and MATLAB programs that output the Bode plot of the plant are listed in Program 2 and Program 3, respectively. In Program 2, the header file *control.h* that defines the macros and prototypes of member functions in the *CControl* class needs to be included at the beginning of the program. Objects of the *CControl* and *CPlot* classes need to be instantiated for accessing member functions of the *CControl* and *CPlot* classes, respectively. The function *CControl::model()* is used to create a zero-pole-gain model of the plant based on the plant's poles, zeros, and gain. The function *CControl::grid()* is used to generate grid lines for plots. The function *CControl::bode()* is used to create the Bode plot of the plant. Figure 5 shows the Bode plot generated by Program 2. The Bode plot created using the CCST has a default frequency range from 0.1 to 1,000 rad/s as shown in Figure 5. The MATLAB program for solving Example 1 is shown in Program 3. The function *zpk()* is used to create a zero-pole-gain format of the plant. The function *bode()* is used to create the Bode plot of the plant. The grid lines are added to the Bode plot by the function *grid()*. The Bode plot created by Program 3 is the same as Figure 5.

The above comparison demonstrates that the CCST can obtain the same solutions, through similar programs, as the MATLAB Control Systems Toolbox. In addition, since the CCST is a C-based, open source, and open architecture software, the CCST is a proper candidate for developing diverse utility programs that facilitate solving various control problems in real-time control systems.

```

#include <control.h>
int main() {
    CControl plant;
    CPlot plot;
    double k = 4;
    array double complex p[] = {0, -2};

    plant.model("zpk", NULL, p, k);
    plant.grid(1);
    plant.bode(&plot, NULL, NULL, NULL);
    return 0;
}

```

**Program 2.** Ch program that outputs the Bode plot of the plant shown in Figure 4.

```
%This Matlab program will create a Bode plot
% with grid lines and a frequency range from
% 0.1 to 1000 rad/s.
```

```
k = 4;
p = [0, -2];
```

```
plant = zpk([], p, k);
bode(plant, {0.1, 1000});
grid on;
```

**Program 3.** Matlab program that outputs the Bode plot of the plant shown in Figure 4.

### Use of Web-Based Control System Design and Analysis System for Interactive Teaching and Learning

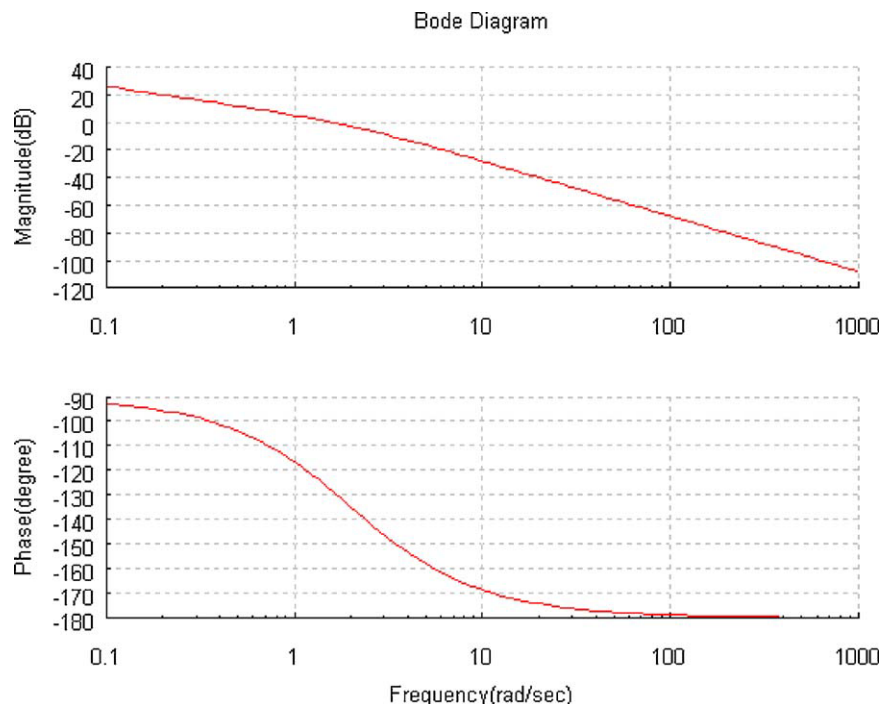
With the WCSDAS, the control systems design and analysis are performed using a Web browser on a client machine. Students can select a design and analysis method and specify system model type, system type, and system parameters in the Web browser. The students' inputs are validated, and informative error messages are displayed when the inputs are not valid. These input data are then transferred to a Web server for numerical computation, and the simulation results are sent back to the client machine and displayed in the client Web browser through the common gateway interface (CGI) using the Ch interpretive environment. The capability of immediate response to the system's input makes the WCSDAS a suitable interactive teaching and learning tool.

The design of the WCSDAS does not have a limit on the number of people who can simultaneously access and use it. However, a Web server may set the maximum number of simultaneous

access. In addition, the WCSDAS uses the most primitive features for the Web. Therefore, the system can be accessed through almost all commonly used Web browsers in various different platforms. It has been successfully tested using the Internet Explorer, Opera, and Google Chrome Web browsers on Windows machines and the Mozilla Firefox Web browser on Windows, Linux, and Mac OS machines.

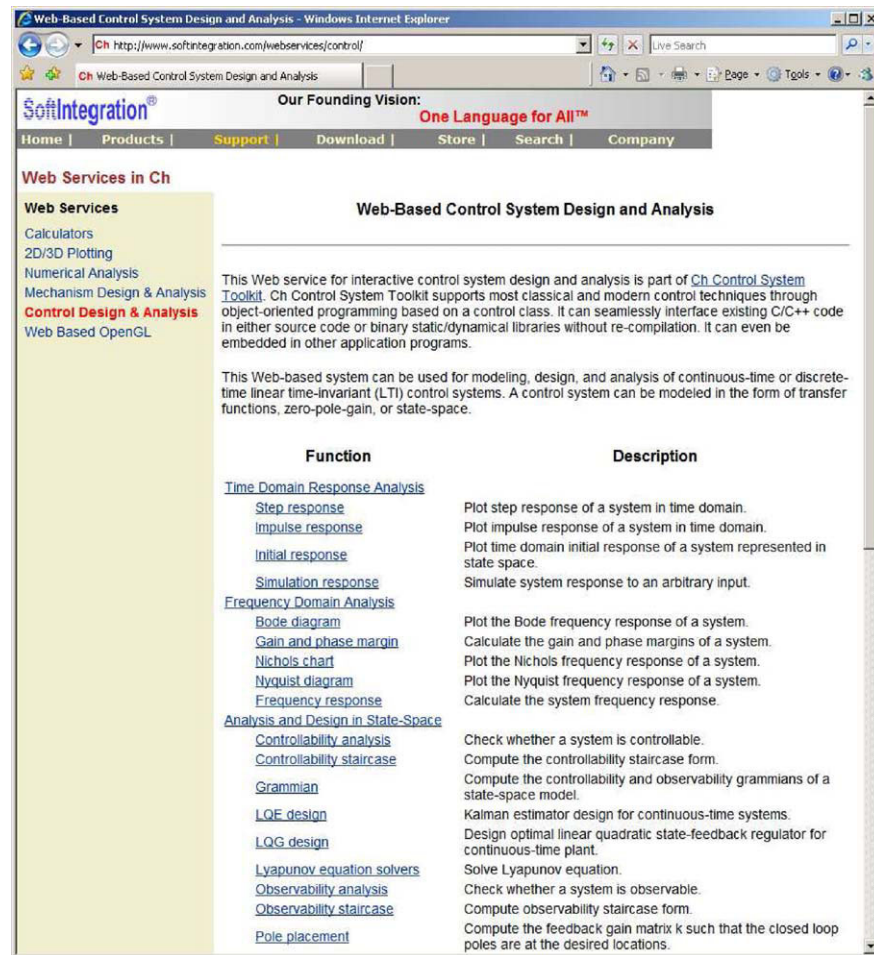
A unique feature of the WCSDAS is its ability to design, analyze, and verify control strategies over the Internet without the need of software installation, system configuration, or programming. Students can focus on the control systems problems and obtain the results immediately and interactively. The WCSDAS provides one example for each function in the CCST to illustrate its usage. By following these examples, entering system parameters in Web forms, and clicking buttons to select different choices, students can gain the design and analysis experience of control systems. Figure 6 shows a partial view of the index page of the WCSDAS.

In this section, using the WCSDAS to solve Example 1 is demonstrated as follows. By clicking the "Bode diagram" hyperlink on the index page shown in Figure 6, the front page of the Bode response is popped up as shown in Figure 7. Selecting the "Zero-pole-gain representation" and clicking the "Continue" button, a plant definition page is generated as shown in Figure 8. Specifying no zeros and two poles and clicking the "Submit" button, a new page is brought up as shown in Figure 9, which allows students to enter poles and the gain of the defining plant. Once finishing the input, students click the "Run" button to display the Bode plot of the plant as shown in Figure 10. From the previously described procedure, students do not need to write any programs to solve control problems by using the WCSDAS.

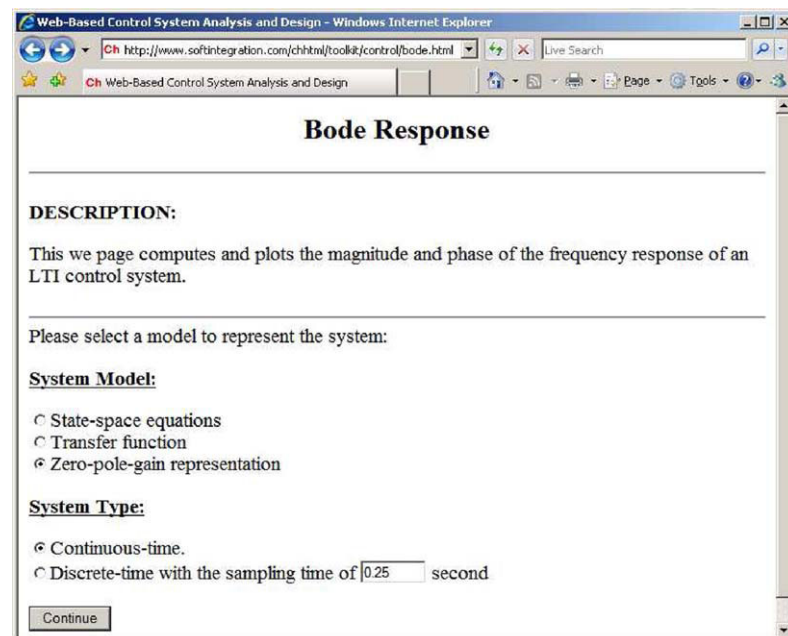


**Figure 5** Bode plot generated by Program 2. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]





**Figure 6** A partial view of the index page of a Web-based control system design and analysis system. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]



**Figure 7** The front page of the Bode response function. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]



**Bode Response**

**Example:**

To build a system with dimensions different from the system below, please click [here](#).

Zeros:

Poles:

Gain K:

☐ Plotting grid

---

**Define Dimensions of the System:**

The number of zeros   
 The number of poles

**Figure 8** The default plant definition page of the Bode response example. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

**Bode Response**

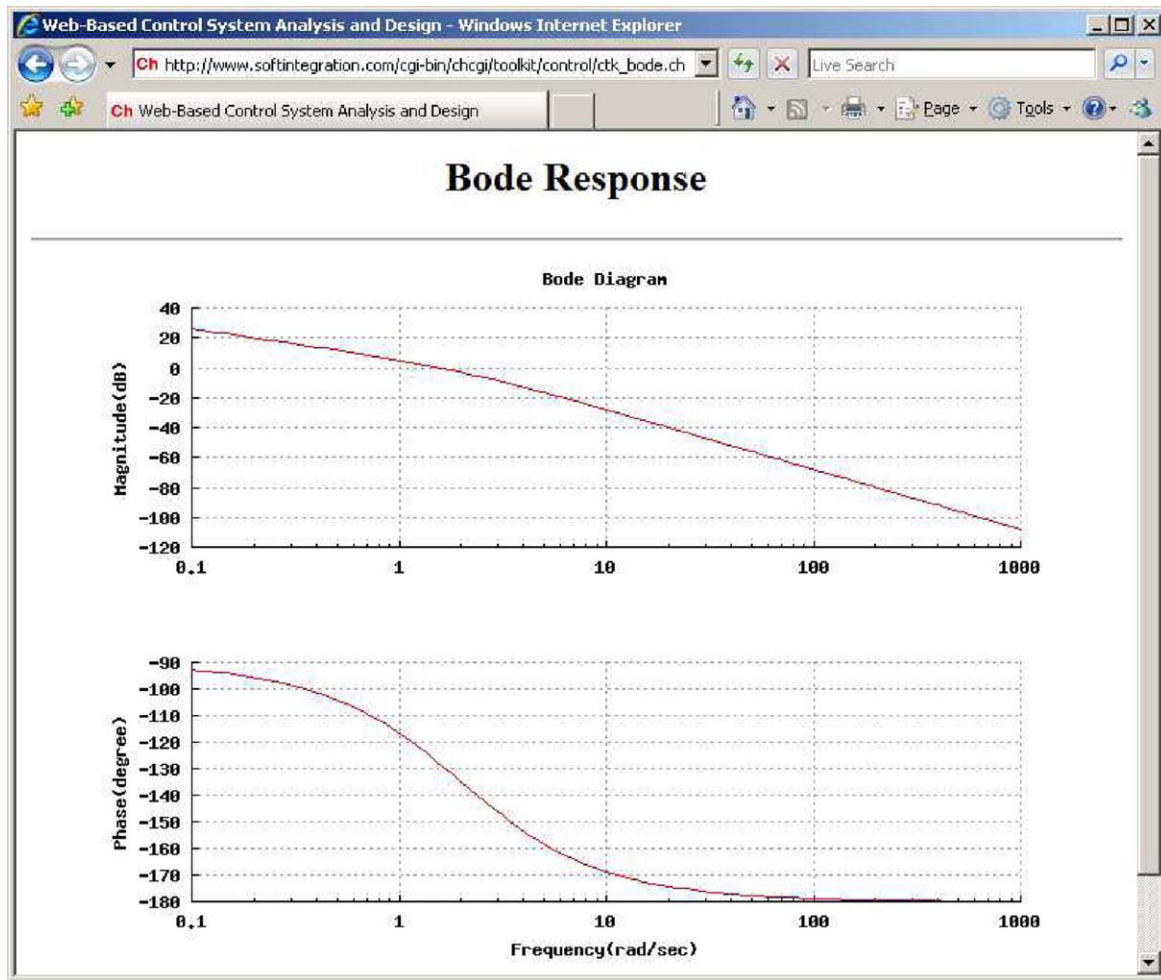
**Please fill out the entries below**

Pole =

Gain K:

☒ Plotting grid

**Figure 9** A Web page allowing users to define a plant. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]



**Figure 10** The Bode plots created by the WCSDAS for the system shown in Figure 4. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

### Ch Control System Toolkit-Based Interactive Utility Programs

The CCST is an appropriate toolkit for students with certain programming experience to solve various control problems. For complicated control system design and analysis, interactive utility programs may be useful to assist students without any programming experience. Since the CCST is open source and open architecture, instructors are able to develop interactive programs that allow students to input control system parameters and receive system output from computer terminals. With interactive utility programs, students can focus on comprehending the design and analysis theories/principles without the need of worrying about the programming issues. A lead compensator design problem given below illustrates how to utilize CCST-based utility programs to assist students in compensator design.

**Example 2:** For the system shown in Figure 4, design a lead compensator so that the characteristic equation of the system has a pair of dominant poles at  $-2 \pm i2\sqrt{3}$ .

First of all, for students with programming experiences, a concise Ch program, such as Program 4 in Appendix, can

be written with the CCST to solve this problem. Students can assign different values to the compensator's pole or zero to obtain the corresponding system performance. On the other hand, for students without programming experiences, a CCST based interactive program can be provided to assist them in solving this problem. As an example, a simple interactive utility program, *interact\_leaddesign.ch*, was developed based on Program 4 in Appendix to provide a rudimentary interface to allow for sequential user inputs as shown in Figure 11. The program *interact\_leaddesign.ch* is available for downloading at [iel.ucdavis.edu/course/EME172](http://iel.ucdavis.edu/course/EME172). With this program, a student needs to define the plant, specify a dominant pole, and specify either the pole or zero of the lead compensator. As shown in Figure 11, a student can accept a default value by pressing the "Enter" key, or change a default value by typing in a different number. Once a student completes the input sequence, the compensator's information, the corresponding step response characteristics, and plots for the root locus and step response of the compensated system will be displayed as shown in Figures 11–13, respectively. Repeating the above process with new input values, students will have different outputs.

```

Ch Professional
C:/users/ycchou/PhD/paper/tdscc/program/ch> ./interact_leaddesign.ch

Values inside brackets are default values.
Press Enter to adopt default values or type in different values.

Define the plant:
  number of zeros [0]:
  number of poles [2]:
    #1 pole:
      real part [0]:
      imaginary part [0]:
    #2 pole:
      real part [-2]:
      imaginary part [0]:
  gain [4]:

Specify a pole for the dominant pole pair:
  real part [-2]:
  imaginary part [3.4641]:

Specify the compensator (0: pole 1: zero) [0]: 1
  zero [-4]: -3

Compensator:
  Pole: -5.600001
  Zero: -3.000000
  Gain: 4.799998

Rise time: 0.369171
Settling time: 2.079070
Percent overshoot: 20.855323

```

Figure 11 A basic interface, provided by `interact_leaddesign.ch`, allowing for sequential user inputs. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

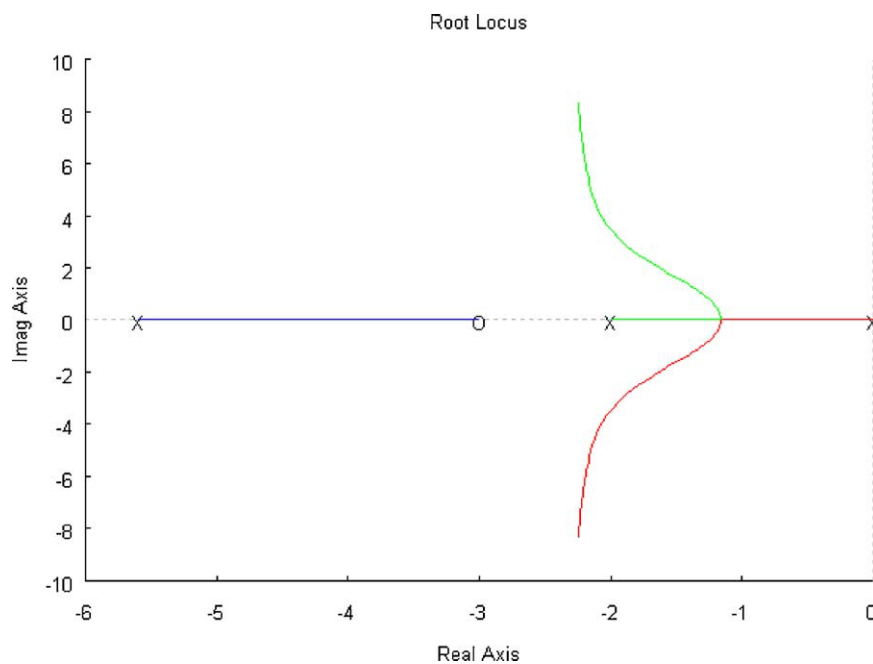
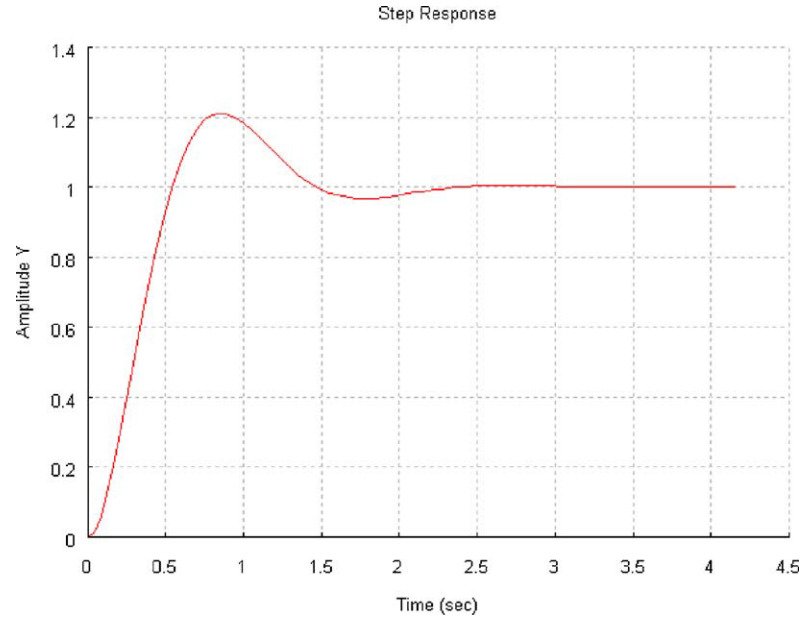


Figure 12 Root locus of the system, created by `interact_leaddesign.ch`, for the compensator's zero at  $-3$ . [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]



**Figure 13** Step response of the system, created by interact.leaddesign.ch, for the compensator's zero at  $-3$ . [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

### Web-Based Controller/Compensator Design Module for Lead/Lag Compensator Design

Traditionally, the design of a control system is an iterative process. Each iteration consists of two steps. In the first step, the unknown parameters of the system regarding the controller's pole, zero, and gain are calculated based on the design specifications. In the second step, the performance of the system is evaluated through simulation and compared to the given specifications. If the system's performance does not meet the design specifications, the design variables are modified and a new iteration is carried on. These repeated calculations and simulations impede students in the process of comprehending the relationship between the system performance and maneuvered design parameters.

A WCCDM can avoid previously mentioned repeated manual calculations and simulations. With the WCCDM, the modification of the system parameters produces an immediate effect on the system's response, which allows students to perceive the gradient of change in the system performance due to the parameters they modify. Currently, the WCCDM provides Lead/Lag compensator design via the root locus. The frequency response design approach with the Bode plot can be easily implemented and added to the WCCDM using the CCST. Similar to the WCSDAS, the maximum number of students who can access the WCCDM is determined by a Web server. Like, WCSDAS, the WCCDM can be used through various commonly used Web browsers in different platforms.

In this section, Example 3 is used to illustrate how to utilize the WCCDM for the lead compensator design.

**Example 3:** For the system shown in Figure 4, design a lead compensator so that the characteristic equation of the system has a pair of dominant poles at  $-2 \pm i2\sqrt{3}$ . The first design is to choose the compensator's pole at  $-20$ . The second design is to choose the compensator's zero at  $-4$ . Compare these two designs and select the one that yields better system performances.

Figure 14 shows the front page of the WCCDM for the Lead/Lag compensator design. Students can choose the type of

a compensator at the bottom of the front page. For solving Example 3, students should select the "Lead compensation" option and click the "Continue" button. Since the plant is given in zero-pole-gain model, users can choose "Zero-pole-gain representation" on **The lead compensator design page** shown in Figure 15. By clicking the "Continue" button, **The plant definition page** is brought up as shown in Figure 16. This page gives a sample plant. Students can also define a new plant in the lower portion of the page. Here the sample plant, which matches the plant in Example 3, should be used. The root locus plot of the uncompensated system is displayed on the next page, **The lead compensation design specification page** by clicking the "Root locus" button, as shown in Figure 17. In this page, students can then specify the lead compensation design specifications associated with the system performance and the compensator parameters. There are two ways to describe the desired dynamic response performance of the compensated systems. The "Method I" specifies the percent overshoot and the settling time/rise time. The "Method II" chooses the dominant pole's locations. Here, "Method II" is selected to solve the Example 3. Since the default values in the WCCDM are set for the specifications used in the Example 3, by simply clicking the "Submit" button, the uncompensated system's root locus and the step response as shown in Figure 18 are displayed at the upper portion of the next page; and the compensated system's characteristics, root locus, and the step response as shown in Figure 19 are outputted at the lower portion of the page. The lead compensator is redesigned by choosing the compensator's zero at  $-4$ , and the corresponding output for the compensated system is shown in Figure 20. Comparing the outputs shown in Figures 19 and 20, the first design has better system performances, that is, shorter rise time and settling time, and a smaller percent overshoot.

The WCCDM introduced in this section shows the potential to extend the WCSDAS by developing complementary Web-based tools. These augmented Web-based tools not only can extend the functionalities of the WCSDAS, but also can help students avoid tedious calculations associated with the compensator design.

Web-Based Compensator Design via Root Locus - Windows Internet Explorer

http://ieel.ucdavis.edu/course/control/compensate/index.html

## Web-Based Compensator Design via Root Locus

**Description:**

Given a unit feedback system shown in Figure 1, design a compensator that is in cascade with the plant to improve the steady-state error or transient response of the system. The passive compensator used to improve a system's steady-state error is the lag compensator. The passive compensator used to improve a system's transient response is the lead compensator. The compensated system is shown in Figure 2.

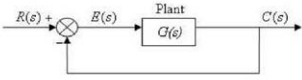


Fig 1 Uncompensated system

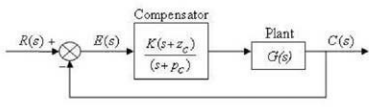


Fig 2 Compensated system

**Compensation type:**

☐ Lag compensation

☒ Lead compensation

**Source code and contact:**

Click [here](#) to download the HTML and Ch CGI source code for this Web page and the subsequent Web pages.

For any issues related to this Web site, please contact [control@ieel.ucdavis.edu](mailto:control@ieel.ucdavis.edu)

**Figure 14** The front page of the Web-based controller/compensator design module. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

Web-Based Compensator Design via Root Locus - Windows Internet Explorer

http://ieel.ucdavis.edu/cgi-bin/projects/course/control/compensate/compensator\_design.ch

## Lead Compensator Design

**Design procedures:**

1. Evaluate the performance of the uncompensated system to determine how much improvement in transient response is required.  
(Note that when formulas for a second order system are used to calculate transient response properties of a higher order system, the second order approximation should be validated.)
2. Find the compensated system's dominant poles.
3. Select lead zero  $z_c$  or lead pole  $p_c$ .
4. Calculate  $p_c$  or  $z_c$  using  $\angle L(s) = (2k+1)180^\circ$  ( $k = 0, \pm 1, \pm 2, \dots$ )
5. Find gain  $K$  using  $|KL(s)| = 1$
6. Validate the second order approximation of the compensated system.
7. Compare the transient response of both uncompensated and compensated systems to see if the compensated system meets the design specifications.

**Plant model:**

☐ State-space equations

☐ Transfer function

☒ Zero-pole-gain representation

**Figure 15** The lead compensator design page. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]



**Plant Characteristics**

**Plant:**

**Poles:**

0    -2

**Gain:**

4

Root locus    Reset

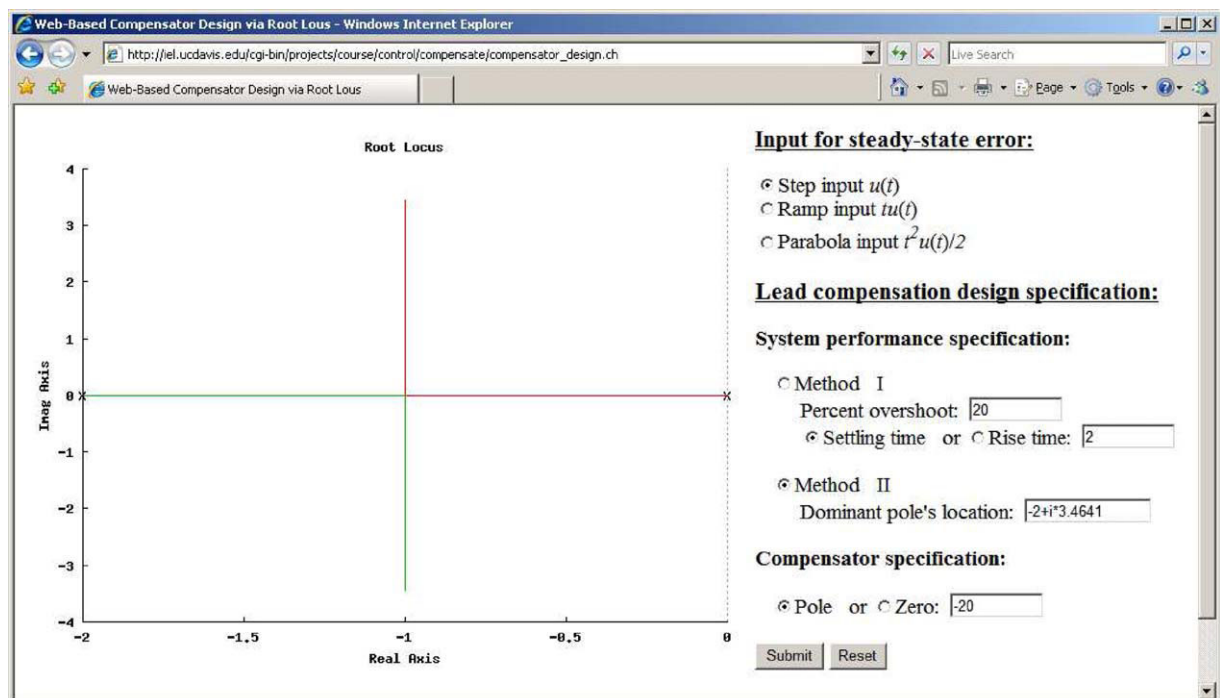
**Define dimensions of the plant:**

Number of zeros: 2

Number of poles: 4

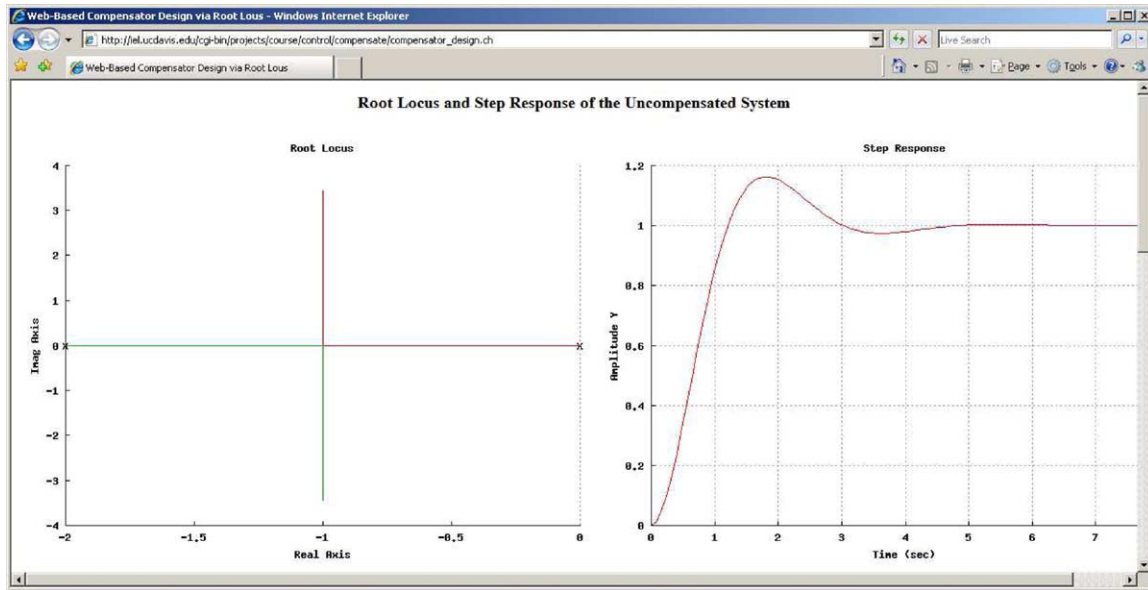
Submit    Reset

**Figure 16** The plant definition page. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

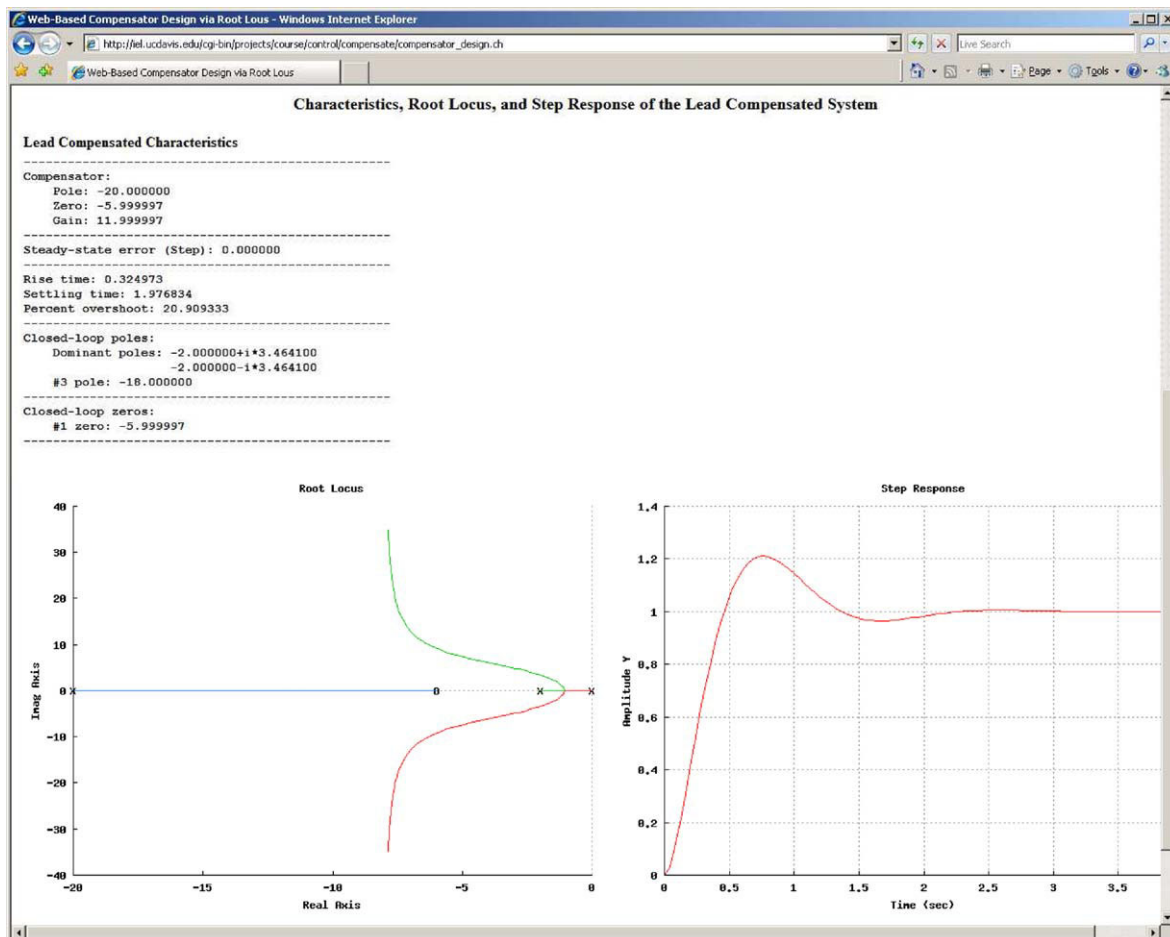


**Figure 17** The lead compensation design specification page. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

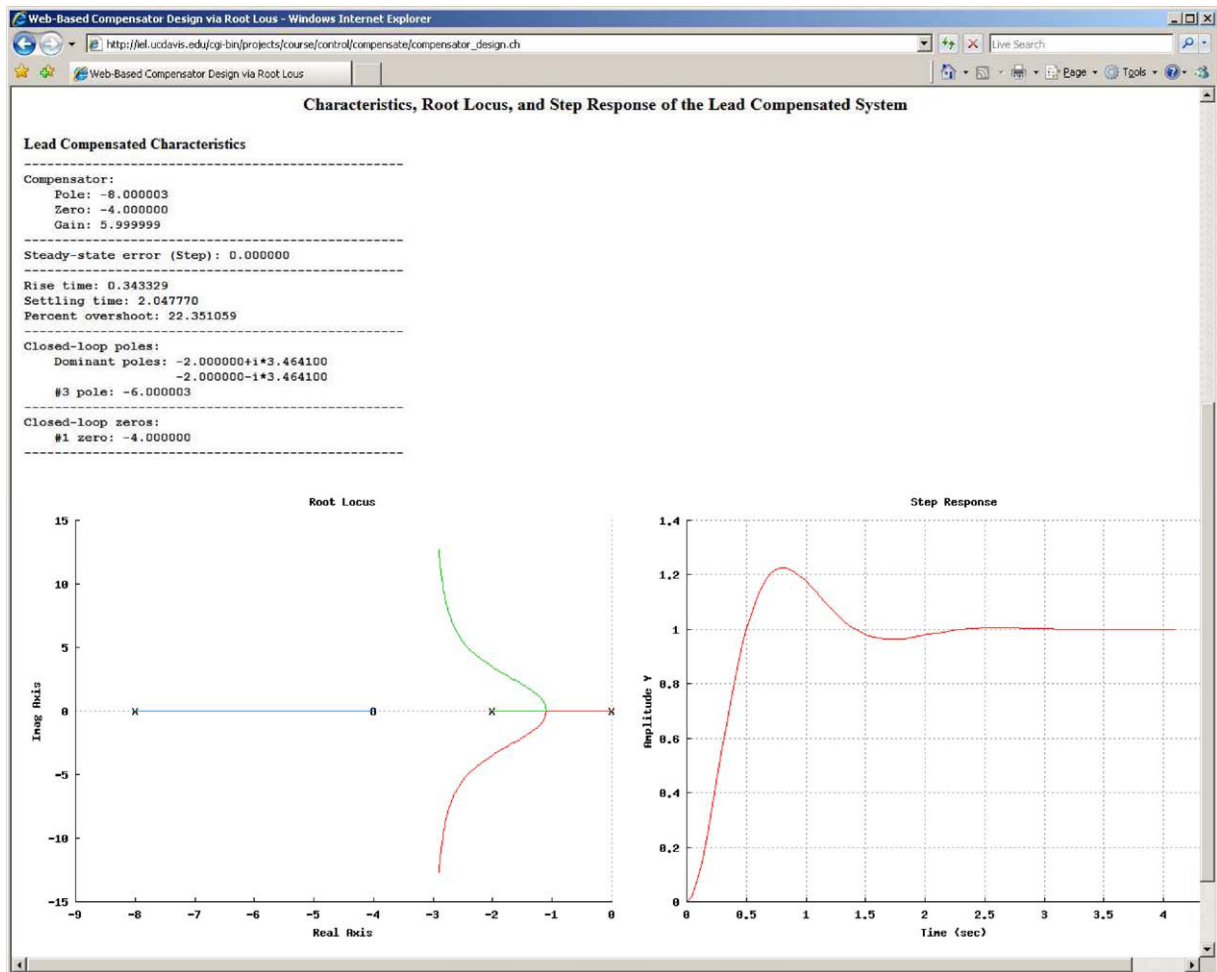




**Figure 18** The root locus and the step response of the uncompensated system. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]



**Figure 19** The system's characteristics, root locus, and the step response of a compensated system with a lead pole at  $-20$ . [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]



**Figure 20** The system's characteristics, root locus, and the step response of a compensated system with a lead zero at  $-4$ . [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

## STUDENT EVALUATION

The Ch Control Systems Toolkit, WCSDAS, and WCCDM have been used for teaching an undergraduate control course "Automatic Control of Engineering Systems" at the University of California, Davis. The Web-based tools, WCSDAS and WCCDM, were also used as teaching tools for the course "Dynamic Systems and Controls" at the Michigan Technological University. The effectiveness of the software tools was evaluated through a questionnaire survey. The classified questions in the survey are listed in Table 2. The survey contains six types of questions with respect to students' experience and proficiency of using different operating systems, helpfulness of the software tools for the design and analysis of control systems, errors encountered when using these tools, sufficiency of the error messages provided by the tools for error correction, features of the tools that are preferred or could be added or improved, and suggestions or comments for the course.

The survey was conducted both at UC Davis and Michigan Tech. Overall, the students' evaluations were quite positive. For the survey conducted at UC Davis, most students found that the CCST, WCSDAS, and WCCDM were very helpful for control system design and analysis. Students liked the simplicity and ease

**Table 2** Classified Questions in the Student Survey

1	Experience and proficiency of using different operating systems
2	Helpfulness of the CCST, WCSDAS, and WCCDM for the design and analysis of control systems
3	Errors encountered when using the CCST, WCSDAS, and WCCDM
4	Sufficiency of the error messages provided by the CCST, WCSDAS, and WCCDM for error correction
5	Favorite features of the CCST, WCSDAS, and WCCDM, and features that could be added or improved
6	Suggestions or comments for the course

of using these tools. They especially liked the Web-based tools. Based on the collected data of the survey, although 73% of the students in the class did not have C/C++ programming experience, most of them were able to use the CCST by following the provided example programs. Some students, however, recommended cover more C/C++ topics either in discussion sessions or during lectures. Student feedback at Michigan Tech was also very positive. Some exemplary students' comments are as follows: "Very user friendly! Very cool (helpful) tool!" "The warning signs were con-

**Table 3** Quantitative Survey Results for the Second to Fourth Types of Questions in Table 2

Question	Yes	No	Not sure	Total
Did you find the WCSDAS helpful for control systems design and analysis?	42	4	13	59
Did you encounter any errors while using the WCSDAS? Were the error messages provided by the WCSDAS adequate enough for error correction?	5	52	2	59
Did you find the WCCDM helpful for compensator design?	42	3	14	59
Did you encounter any errors while using the WCCDM? Were the error messages provided by the WCSDAS adequate enough for error correction?	6	47	6	59

cise, and made aware of the exact problem.” “I like the Root locus sketching tool. I think it is great and needs no improvement. They were very helpful and descriptive enough, very clear.” “I think this is great tool, very useful and saves a lot of time when you need results. Also, it is great that you don’t need to have the MATLAB installed in your computer.” “I liked the user interface so that I didn’t have to write Matlab code.” “Overall it is a very helpful system that is easy to access from any computer.”

The quantitative survey results at Michigan Tech regarding the WCSDAS and WCCDM are listed in Table 3. Table 3 shows the survey results for the second to fourth types of questions in Table 2. A total of 59 students took this survey. As for the WCSDAS, there are 42 students who found the WCSDAS helpful for control systems design and analysis. There are only five students who encountered errors such as incorrect format for input a formula when using the WCSDAS. Those students also found the error messages adequate enough to correct their errors. As for the WCCDM, there are 42 students who found the WCCDM helpful for compensator design. Similarly, there are only six students who encountered errors when using the WCCDM. Those students also found the error messages adequate enough to correct their errors.

## CONCLUSIONS

Based on the CCST and the WCSDAS, the WCCDM has been developed. The CCST, WCSDAS, and WCCDM have been used for teaching the course of automatic control of engineering systems. With the CCST, students only need to write a few lines of C/C++ code for solving complicated engineering problems in the design and analysis of control systems. In addition, the CCST functions can be easily integrated with C/C++ programs to develop a diverse set of interactive utility programs that can help students learn the design and analysis of control systems without any programming requirements. The WCSDAS and the WCCDM are platform and location independent. Complicated control problems can be solved using these Web-based tools *without even writing a single line of code anytime anywhere* through a Web browser. The CCST, WCSDAS, and WCCDM are open source and freely available for downloading from the Internet for teaching and student learning. From a pedagogical point of view, students can examine the source code to under-

stand theories and algorithms as well as their implementation in software. They can modify the existing code to solve other problems.

The CCST, WCSDAS, and WCCDM have been used to teach undergraduate control courses at the University of California, Davis and Michigan Technological University. Students’ feedback of using these tools for learning control system design and analysis is quite positive. Students especially like the simplicity of the Web-based learning tools. All these software packages, utility programs, supplementary documents, lecture and discussion presentation materials, and homework solutions using the CCST, WCSDAS, and WCCDM are available for downloading on the Internet ([www.softintegration.com/products/toolkit/control](http://www.softintegration.com/products/toolkit/control)). They can be modified to adapt to similar courses in other institutions. Because of the close tie to real-time hardware, our C/C++ based courseware and teaching strategy can also be used for teaching other courses such as mechatronic systems.

## ABBREVIATIONS

CCST	Ch Control System Toolkit
WCSDAS	Web-based Control System Design and Analysis System
WCCDM	Web-based Controller/Compensator Design Module

## APPENDIX

```
#include <control.h>

int main() {
    // Define the plant
    double plant_k = 4;
    array double complex plant_p[] = {0, -2};

    // Specify a pole for the dominant pole pair
    array double complex dominant_p[] = {complex(-2, 2*sqrt(3))};

    // Specify the compensator's pole/zero
    array double complex comp_p[1], comp_z[1] = {-3};

    // Necessary variables
    CControl plant, comp, fb_sys, *ol_sys, *cl_sys;
    CPlot rlocus_plot, step_plot;
    double comp_k, tr, ts, os;
    array double fb_num[] = {1}, fb_den[] = {1};

    // Calculate compensator's pole/zero and gain as well as
    // corresponding step response characteristics of the compensated system
    plant.model("zpk", NULL, plant_p, plant_k);
    plant.compensatorZeroPoleGain(dominant_p, comp_z, comp_p, &comp_k);
    comp.model("zpk", comp_z, comp_p, comp_k);
    fb_sys.model("tf", fb_num, fb_den);
    ol_sys = comp.series(&plant);
    cl_sys = ol_sys->feedback(&fb_sys);
    cl_sys->stepinfo(&tr, &ts, &os, NULL, NULL, NULL, NULL);

    // Display compensator's information
    printf("\nCompensator:\n");
    " Pole: %f"
    " Zero: %f"
    " Gain: %f\n", real(comp_p), real(comp_z), comp_k);

    // Display corresponding step response characteristics
    printf("Rise time: %f\n", tr);
    printf("Settling time: %f\n", ts);
    printf("Percent overshoot: %f\n", os);

    // Display plots of root locus and step response for the compensated system
    ol_sys->rlocus(&rlocus_plot, NULL, NULL);
    cl_sys->grid(1);
    cl_sys->step(&step_plot, NULL, NULL, NULL, ts*2);

    return 0;
}
```

## REFERENCES

- [1] S. D. Bencomo, Control learning: Present and future, *Annu Rev Control* 28 (2004), 115–136.
- [2] J. Sanchez, S. Dormido, and F. Esquembre, The learning of control concepts using interactive tools, *Comput Appl Eng Educ* 13 (2005), 84–98.
- [3] M. Johansson, M. Gafvaert, and K. J. Astrom, Interactive tools for education in automatic control, *IEEE Control Syst Mag* 18 (1998), 33–40.
- [4] B. Wittenmark, H. Hagglund, and M. Johansson, Dynamic pictures and interactive learning, *IEEE Control Syst Mag* 18 (1998), 26–32.
- [5] M. Johansson and K. J. Astrom, Virtual interactive systems for control education, in: *Proceedings of the 35th IEEE Decision and Control*, Kobe, Japan, 1996, pp 3888–38889.
- [6] J. L. Guzman, K. J. Astrom, S. Dormido, T. Hagglund, and Y. Piguet, Interactive learning modules for PID control, *IEEE Control Syst Mag* 28 (2008), 118–134.
- [7] J. L. Guzman, Interactive control system design, PhD dissertation, University of Almeria, 2006.
- [8] S. Dormido, F. Gordillo, S. Dormido-Canto, and J. Aracil, An interactive tool for introductory nonlinear control systems education, in: *Proceedings of the 17th IFAC World Congress*, Barcelona, Spain, 2002.
- [9] W. J. R. Hoefer and P. P. M. So, A time-domain virtual electromagnetics laboratory for microwave engineering education, *IEEE Trans Microw Theory Tech* 51 (2003), 1318–1325.
- [10] J. P. Keller, Interactive control system design, *Control Eng Pract* 14 (2006), 177–183.
- [11] M. Casini, D. Prattichizzo, and A. Vicino, The automatic control Telelab: A user-friendly interface for distance learning, *IEEE Trans Educ* 46 (2003), 252–257.
- [12] C. C. Chan, R. Kwan, and S. F. Chan, Learning Control Systems on the Web, In: *Proceedings of the International Conference on Computers in Education*, 2002, Vol. 2|2, Vol. xliii+1580, pp 894–895.
- [13] J. A. Kypuros and T. J. Connolly, Student-configurable, web-accessible virtual systems for system dynamics and controls courses, *Comput Appl Eng Educ* 16 (2008), 92–104.
- [14] S. Uran and K. Jezemik, Virtual laboratory for creative control design experiments, *IEEE Trans Educ* 51 (2008), 69–75.
- [15] J. A. Mendez, C. Lorenzo, L. Acosta, S. Torres, and E. Gonzalez, A Web-based tool for control engineering teaching, *Comput Appl Eng Educ* 14 (2006), 178–187.
- [16] J. Sanchez, S. Dormido, R. Pastor, and F. Morilla, A Java/Matlab-based environment for remote control system laboratories: Illustrated with an inverted pendulum, *IEEE Trans Educ* 47 (2004), 321–329.
- [17] D. Buhler, W. Kuchlin, G. Gruler, and G. Nusser, The virtual automation lab—Web based teaching of automation engineering concepts, In *Proceeding from the 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, Edinburgh, Scotland, 2000.
- [18] Y. Zhu, B. Chen, and H. H. Cheng, An object-based software package for interactive control system design and analysis, *ASME J Comput Inform Sci Eng* 3 (2003), 366–371.
- [19] H. H. Cheng, *C for engineers and scientists: An interpretive approach*, McGraw-Hill, Inc., New York, 2009.
- [20] H. H. Cheng, *Ch: A C/C++ interpreter for script computing*, *C/C++ User's J* 24 (2006), 6–12.
- [21] H. H. Cheng, Scientific computing in the Ch programming language, *Sci Program* 2 (1993), 49–75.
- [22] Q. Yu, B. Chen, and H. H. Cheng, Web-based control system design and analysis, *IEEE Control Syst Mag* 24 (2004), 45–57.

## BIOGRAPHIES



**Bo Chen** is an Assistant Professor in the Department of Mechanical Engineering – Engineering Mechanics and the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton. Her research interests include artificial immune systems and pattern recognition, mobile agent and multi-agent systems, sensor networks and networked embedded systems, structural health monitoring, and vehicle electronics and control networks. She has published over 50 refereed journal and conference papers. Dr. Chen received the Best Paper Award at the 2008 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications. She has been an active member of the ASME, and was Symposium Chairs and Session Chairs for several international conferences. She is a member of the Executive Committee of the Technical Committee on Mechatronic and Embedded Systems and Applications (MESA), ASME Design Engineering Division.



**Harry H. Cheng** is a professor in the Department of Mechanical and Aerospace Engineering, and Graduate Group in Computer Science at the University of California, Davis. He is also the Director of the Integration Engineering Laboratory at UC Davis. Before joining the faculty at UC Davis, he worked as a Senior Engineer on robotic automation systems in the Research and Development Division at United Parcel Service from 1989 to 1992. He is the founder of SoftIntegration, Inc. He received the M.S. degree in mathematics in 1986 and the Ph.D. degree in mechanical engineering in 1989 from the University of Illinois at Chicago. His research is focused on computer-aided engineering, mobile agent-based computing, intelligent mechatronic and embedded systems, robotics, and innovative teaching. He has published over 160 papers in refereed journals and conference proceedings and holds one U.S. patent. He is the

author of the book entitled “*C for Engineers and Scientists: An Interpretive Approach*” published by McGraw-Hill in 2009. He is the original designer and implementer of an embeddable C/C++ interpreter Ch for cross-platform scripting. He participated in revision of the latest C standard called C99 through ANSI X3J11 and ISO S22/WG14 C Standard Committees and made contributions to new C99 numerical features of complex numbers, variable length arrays, and IEEE floating-point arithmetic, which had been implemented in his C/C++ interpreter. He is a Fellow of ASME and a Senior Member of IEEE. He served as the Conference Chair and Program Chair of the IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications.



**Yu-Cheng Chou** is an assistant professor in the Department of Mechanical Engineering at Chung Yuan Christian University, Taiwan. His research interests include mobile agent-based computing, autonomic computing, parallel computing, intelligent mechatronics and embedded systems, and software and system integration. Chou received his master's and PhD degrees in mechanical and aeronautical engineering from the University of California at Davis in 2007 and 2009, respectively. He is a member of the IEEE and the ASME. Contact him at [ycchou@cycu.edu.tw](mailto:ycchou@cycu.edu.tw).