FEATURE

Web-Based Control System Design and Analysis

Design, implementation, and salient features

By Qingcang Yu, Bo Chen, and Harry H. Cheng

The World Wide Web has provided an opportunity for design and analysis of control systems through the Internet. An increasing number of Web-based software packages have been developed to enhance the teaching and design of control systems [1]. Today, one of the most popular applications in control systems is Web-based educational environments and laboratories. As an example, the interactive study support environment presented in [2] provides course management, online exercises, and laboratories. Based on VCLab [3], which contains Java applets and MATLAB plug-ins, the system allows students to enter MATLAB commands on the Web page and submit these commands to the MATLAB program for execution. The output generated by these commands can be displayed on the same Web page. To use this Web-based study environment, however, the MATLAB software has to be installed on the client machines. Instead of using MATLAB plug-ins to perform numerical computation on the client side, Java applets are sometimes used for numerical computation on the client side. For example, applications of Java applets can be found in the Web-based two-degrees-of-freedom robot manipulator simulation system [4] and the Virtual Control Lab [5]. Due to the lack of

0272-1708/04/$20.00©2004IEEE

Table 1. WCDAS supports many commonly used functions for control system design and analysis. These functions include both classical and modern control methods.

1) Time Domain Response Analysis
- Step response
- Impulse response
- Initial response
- Simulation response

2) Frequency Domain Analysis
- Bode diagram
- Gain and phase margin
- Nichols chart
- Nyquist diagram
- Frequency response

3) Analysis and Design in State Space
- Controllability analysis
- Controllability staircase
- Gramian
- LQE design
- LQG design
- Lyapunov equation solvers
- Observability analysis
- Observability staircase
- Pole placement

4) Root Locus Design
- Root locus

5) Model Reduction and Dynamics
- Bandwidth
- Pole-zero map
- Damping factors and natural frequencies
- DC gain
- Sort poles
- Minimal realization
- Pzacancel

6) Model Conversion
- State-space model
- Transfer function model
- ZPK model

7) System Conversion
- Coordinate transformation
- Continuous time to discrete time
- Discrete time to continuous time
- Discrete time to discrete time
- Map delays to poles

8) System Interconnection
- Series
- Parallel
- Feedback
- Append
- Connect

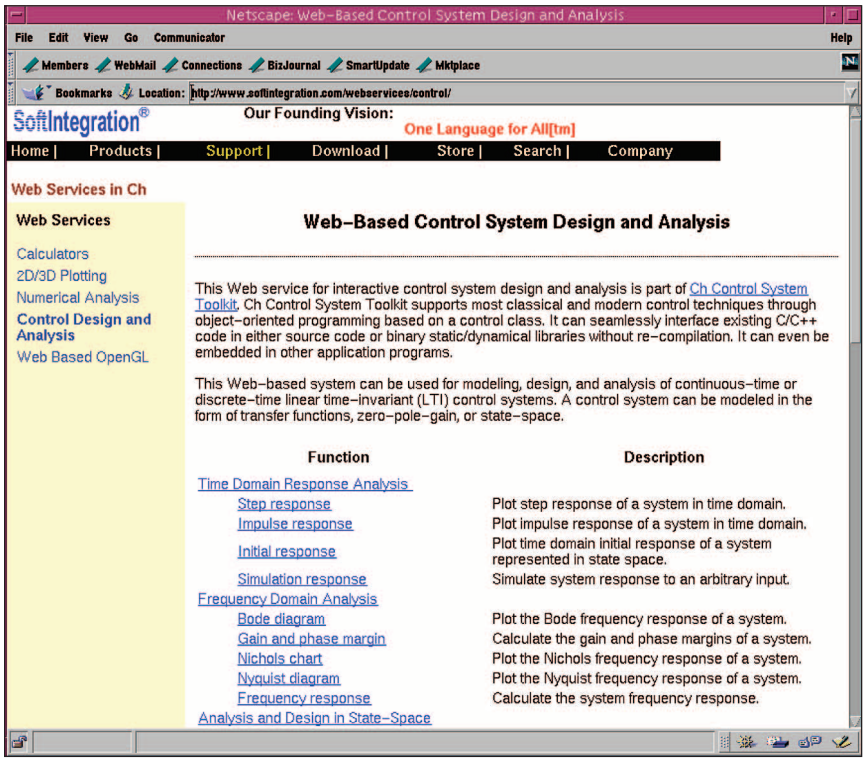

Figure 1. *A partial view of the index page of the Web-based control design and analysis system. The functions provided by the system are classified into the eight categories shown in Table 1.*

powerful numerical computing capabilities in Java and its applets, however, applications using Java applets alone in the simulation of dynamic systems are limited.

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 tool kit. A remote laboratory offers a physical experimental apparatus to remote users through the network.

Most Web-based laboratories use MATLAB as the computational engine. For example, Sanchez et al. [6] proposed a virtual and remote laboratory using Java and MATLAB. In this system, a Web page with Java applets is used as the graphical user interface (GUI) for remote access of the lab. The computations for controller design and analysis are performed in MATLAB in a separate process invoked by an application server called Internet Virtual Lab (IV-Lab). The application server communicates with the Web server using TCP/IP sockets. The implementation of such a virtual control laboratory is complicated because of the inherent deficiency in interfacing MATLAB with external programs. Instead of writing an application server on their own, other remote laboratories [7], [8] use the MATLAB Web server (MWS) to communicate between the HTTP Web server and MATLAB. However, even when using MWS, the communication between the HTTP Web server and MATLAB is inefficient. In our experience, software based on MWS is difficult to develop and maintain.

We have developed a Web-based interactive control design and analysis system (WCDAS) [9] based on Ch, which is a C/C++ interpreter [10], [11], and the Ch control systems toolkit (CCST) [12], [13]. WCDAS covers many classical and modern techniques for control systems design and analysis. Most functions in the system support

both continuous-time and discrete-time linear time-invariant systems modeled in state space, transfer functions, or zero-pole-gain representations. Users can select a design and analysis method and specify system model type, system type, and system parameters in the Web browser. These data are transferred to the server for numerical computation, and the simulation results are sent back to the client through the common gateway interface (CGI) using the Ch interpretive environment. Because both CGI scripts and simulation programs are written in Ch, data exchange between client and server is easily achieved. The system is available for use through the Web without any software installation, system configuration, or programming. This Web-based system is ideal for teaching as well as for solving practical problems in control systems design and analysis. The software packages Ch, CCST, and WCDAS are available for downloading on the Web [12]. The design, implementation, and salient features of WCDAS are described in this article.

From the system developer's point of view, WCDAS is easy to maintain and extend.

Ch and Web-Based Control System

The Ch language [10], [11] is a superset of the C interpreter. Ch supports all features of the C language standard ratified in 1990 [14]. New features such as complex numbers, variable length arrays, IEEE floating-point arithmetic, and type-generic mathematical functions first implemented in Ch were adopted in C99 [15], a new C standard ratified in 1999. In addition, Ch supports classes in C++ for object-based programming. Like other mathematical software packages such as MATLAB, Ch has built-in support of two- and three-dimensional graphical plotting features and computational arrays for matrix computation and linear system analysis with advanced numerical analysis functions based on LAPACK. The application programming interface (API) of the CGI in Ch is similar to those in active server pages (ASP) and Java server pages (JSP). Four classes—CResponse, CRequest, CServer, and CCookie—are provided in the Ch CGI toolkit [16]. CGI programming in Ch is interpretive without compilation and linking, platform independent, and easy to debug and maintain. As a superset of C, Ch can interface with C/C++ programs in both source code format and binary static or dynamical libraries. Ch can also be embedded in other application programs as a scripting engine [17].

The object-oriented CCST [12], developed in Ch, provides a C++ control class to support most classical and modern control techniques. Its member functions contain most functions and features found in the MATLAB Control System Toolbox [18]. Feature comparison of CCST and the MATLAB Control System Toolbox, including sample code in both Ch and MATLAB for solving the same problems, is available on the Web [12]. CCST is convenient for modeling, designing, and analyzing continuous- or discrete-time linear time-invariant control systems in

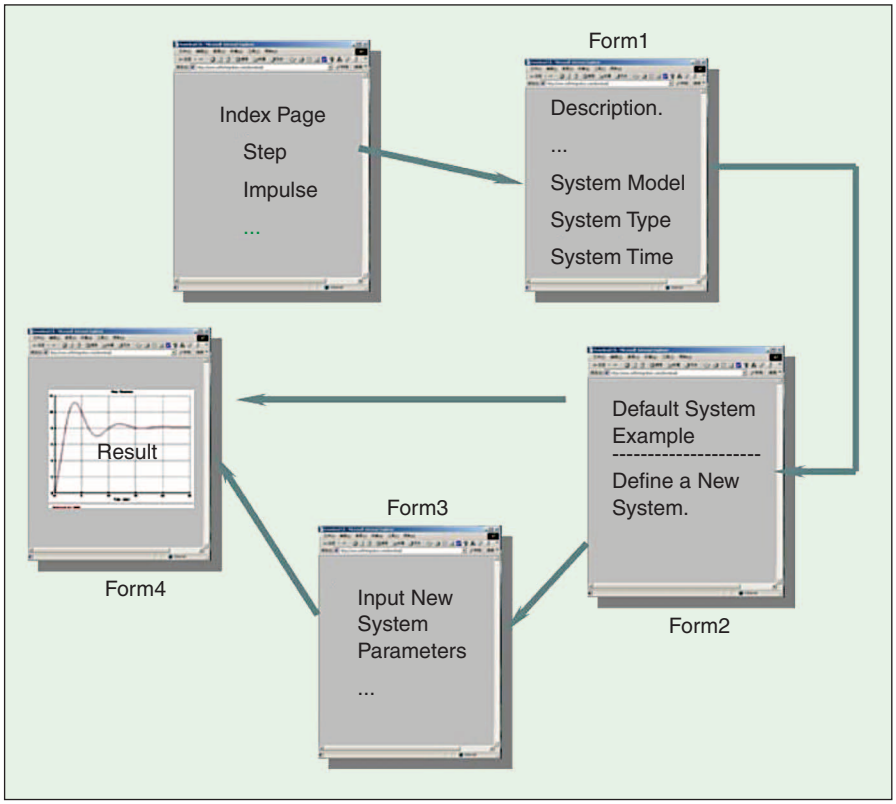

Figure 2. *Overview of the user interface of WCDAS. The user can select a specific function from the index page. The Web page Form 1 describes the selected function and allows the user to select the system type and model type. The upper portion of Form 2 shows a sample system. The user can also define a system in the lower portion of this page. The customized system parameters are submitted in Form 3. Form 4 is the output page.*

both time and frequency domains due to user-friendly graphical representations. Control systems can be modeled in the form of transfer functions, zero-pole-gain representations, or state-space equations.

With WCDAS, control system design and analysis is performed using a Web browser on the client machine without any software installation or tedious programming.

WCDAS is developed using Ch, Ch CGI, and CCST. Unlike Web-based laboratories that use the MATLAB engine, data received by the HTTP server in our system are passed to a computational engine directly through CGI without an extra application server. This approach greatly simplifies the implementation of Web-based control systems design and analysis systems. With our Web-based system, control systems design and analysis is performed using a Web browser on the client machine without any software installation or tedious programming. The user selects a simulation method, system type, system model type, and other system parameters in the Web browser. These selections and data are then transferred to the Web server for computation using Ch and CCST through CGI. The text or graphical results are sent back and displayed in the client Web browser.

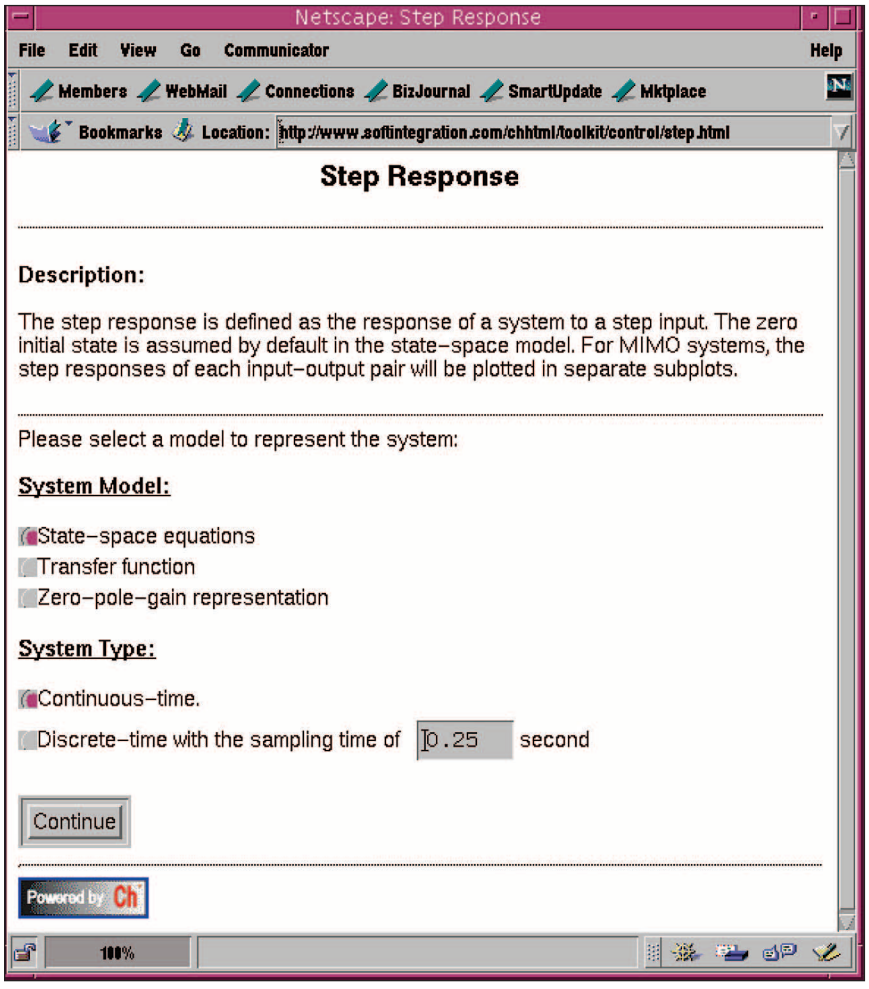

Figure 3. *System model type and system type selection page (Form 1). This page allows the user to choose the desired model and system types.*

Features of the Web-Based Control System

WCDAS makes all features and capabilities of CCST accessible to users through the Web without need for programming. By taking advantage of the Ch language environment and CCST, WCDAS 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 linear time-invariant (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 functionalities of WCDAS are outlined in Table 1. Figure 1 gives a partial view of the index page of the WCDAS.

All functions in WCDAS are interactive, and all parameters such as system types, system model types, and system model data are selected or entered online to solve control design and analysis problems. The user inputs

are validated, and an informative error message is displayed when the inputs are not valid.

A unique feature of WCDAS is its ability to design, analyze, and verify control strategies over the Internet without software installation, system configuration, or programming. The user can focus on the control systems problems and obtain the results interactively. WCDAS provides an example for each function to illustrate its usage. By following the example, entering the system parameters in the form, and clicking buttons to select different choices, the user can gain experience in control systems design and analysis.

From the system developer's point of view, WCDAS is easy to maintain and extend. Each function is implemented by several independent files. The details of the file system of WCDAS are discussed later.

User Interface for the Web-Based Control System

The step-response function is selected as an example to illustrate the user interface (UI) of WCDAS. Figure 2 shows all Web pages related to the step-response function.

The process of using a function in WCDAS starts from the index page. Clicking the "Step response" hyperlink on the index page shown in Figure 1 brings up the page shown in Figure 3 labeled "Form 1." As the first page of the step-response function, the function description is presented at the top of the page. The user can select system model type from state space, transfer function, and zero-pole-gain representation and select system type as continuous or discrete time. For a discrete-time system, the user can specify a sampling time for the system on this page.

By clicking the "Continue" button in Form 1, we reach the page shown in Figure 4, labeled "Form 2." A default sample system whose system type and model type are chosen in Form 1 is given in the upper portion of the page. For the step-response function, the default system in the state-space representation in Figure 4 is

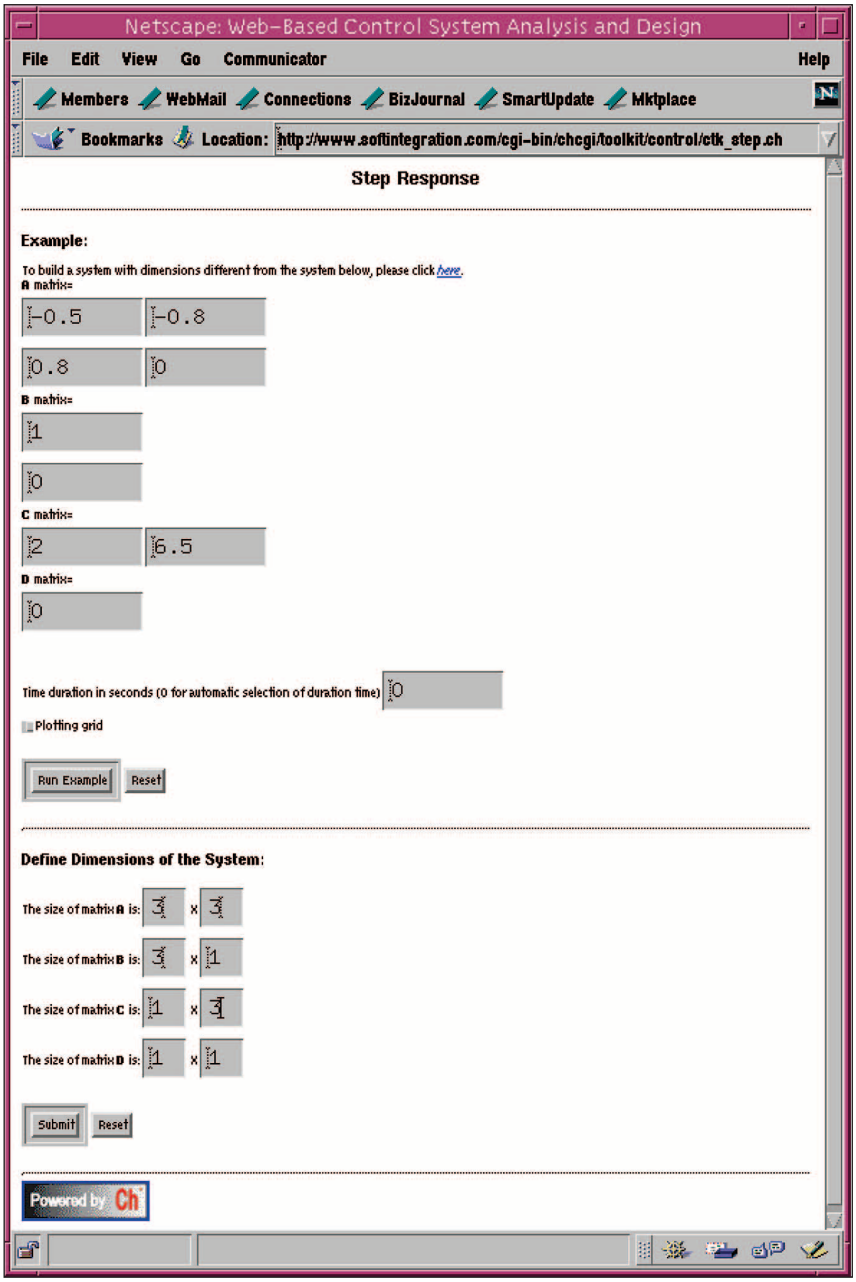

Figure 4. *Run a sample problem or customize a new system (Form 2). The upper portion of this Web page gives a sample system. The user can obtain the step response of the sample system by clicking the* Run Example *button. The lower portion of the page allows the user to define a new system.*

$$\dot{x}_1 = -0.5x_1 - 0.8x_2 + u$$
$$\dot{x}_2 = 0.8x_1$$
$$y = 2x_1 + 6.5x_2.$$

According to the state-space equations,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{1}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{2}$$

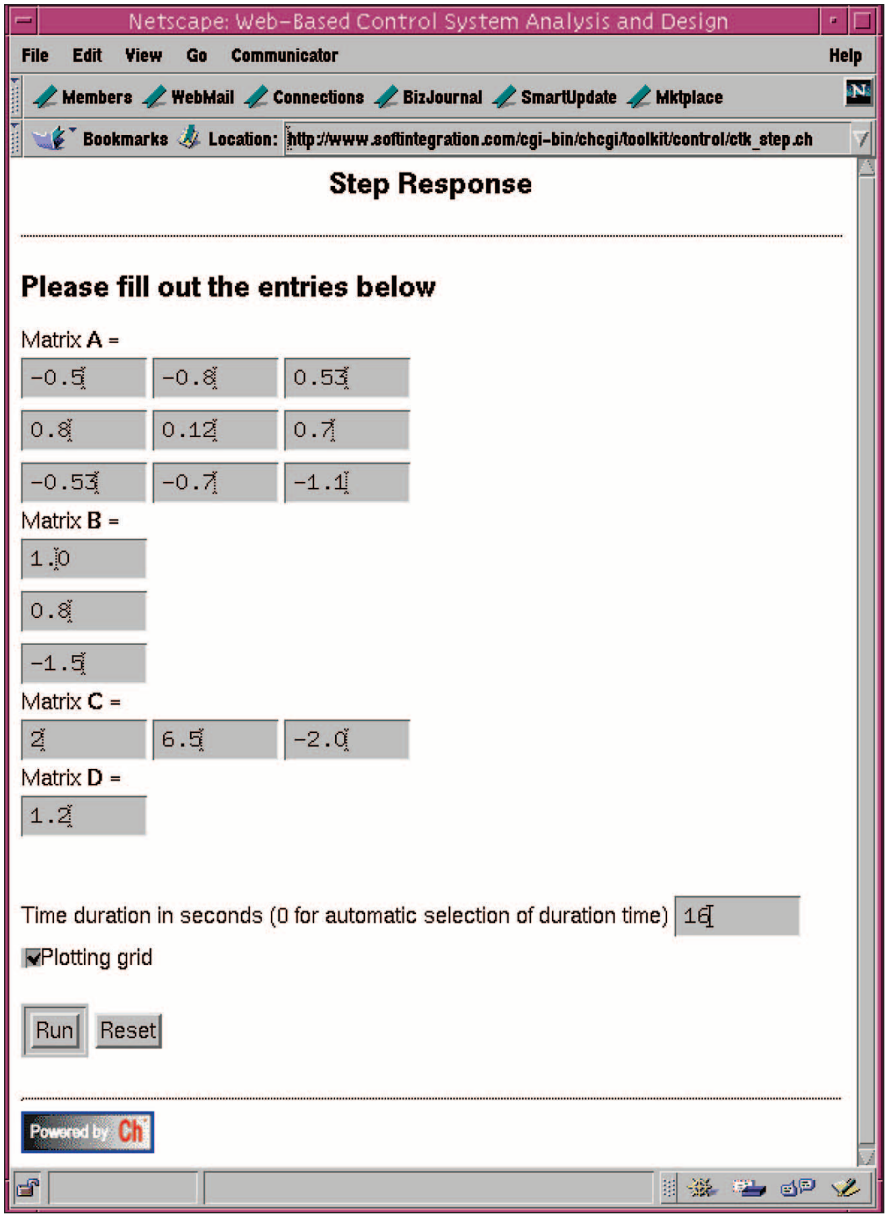

Figure 5. *The parameter input page (Form 3). This page allows the user to enter system parameters.*

values for system matrices **A**, **B**, **C**, and **D** are

$$\mathbf{A} = \begin{bmatrix} -0.5 & -0.8 \\ 0.8 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1.0 \\ 0 \end{bmatrix}$$

$$\mathbf{C} = [2 \;\; 6.5]$$

$$\mathbf{D} = [0].$$

The step response of this sample system is obtained by clicking the "Run Example" button.

The user can also define a system in the lower portion of Form 2. The dimensions of the system matrices, the orders of the numerator and denominator polynomials, or the number of zeros and poles can be specified by the user. For example, the lower portion in Figure 4 asks the user to specify dimensions of the system matrices for the state-space model chosen on Form 1 shown in Figure 3.

By clicking the "Submit" button in the lower portion of Form 2, Form 3 is generated as shown in Figure 5. This page asks the user to enter parameters of the customized system. The user can enter each entry of the system matrices for a state-space model; the coefficients of the numerator and denominator polynomials for a transfer function model; or the zeros, poles, and gain for a zero-pole-gain representation.

The step response of the default system or user-defined system is shown in Form 4 in Figure 6. This Web page is reached by clicking the "Run Example" button in Form 2 or the "Run" button in Form 3. The user can specify the final time of the response and select grid on or off for the step response in Form 3.

Implementation of Web-Based Control System

System Architecture

The system architecture of WCDAS is shown in Figure 7. When a user (client) sends the HTTP server a request for a hypertext markup language (HTML) page, the server can respond to this request directly. If the user's request is for executing a Ch script program in the server, the Ch script program is invoked by Ch CGI [16]. The Ch program retrieves user's data from an HTML document through CGI by the member function "GetForms" of the CRequest class and returns the resulting data to CGI by class CResponse. When a plot output is requested, a Ch plot program named *plot.ch* is called by a Ch script to generate a plot in the portable network graphics (PNG) image file format. The output plot, together with the text results generated by a Ch script, are assembled by member functions of class CResponse and sent to the HTTP server via CGI and then to the user's Web browser.

File System

Figure 8 shows the file system of WCDAS. Each function is implemented by both HTML files and Ch scripts. A Web browser can access HTML files directly. Ch scripts are interpreted by the Ch language environment through CGI. For instance, the Step response on the index page in Figure 1 is linked to an HTML file *step.html*, which creates Form 1 in Figure 3. Forms 2 and 3 and the resulting output page Form 4 are created by a Ch script called *ctk_step.ch*. The handling of multiple pages in one Ch file is discussed later.

As previously described, if the output of a function, such as a step response is a plot, then the Ch script *plot.ch* is invoked to generate the corresponding plot. The program *plot.ch* retrieves the required data through CGI and sends a plot stream back to CGI, as shown in Figure 7.

Data Structure

Most functions in WCDAS support system models in state space, transfer functions, or zero-pole-gain representations for both continuous- and discrete-time systems. To customize a system, the user has to make a suitable choice and enter all model parameters. To avoid the need for entering all of the inputs into a single Web page, multiple Web pages are designed for users to select choices and enter data. The user's selections and system parameters are saved and passed from one Web page to another. To decrease the complexity of handling and transferring data, a clear and well-organized data structure is required.

Figure 9 shows the three-part data structure used for state-space models. The first part is the header, which saves the page ID, system model ID, system type ID, and sampling time for discrete-time systems. The page ID is used to identify different forms described previously. The value of *ss* for the system model *sysModel* indicates state space equations. The value for system type *sysType* can be either continuous or discrete time. The value of *TsValue* specifies the sampling

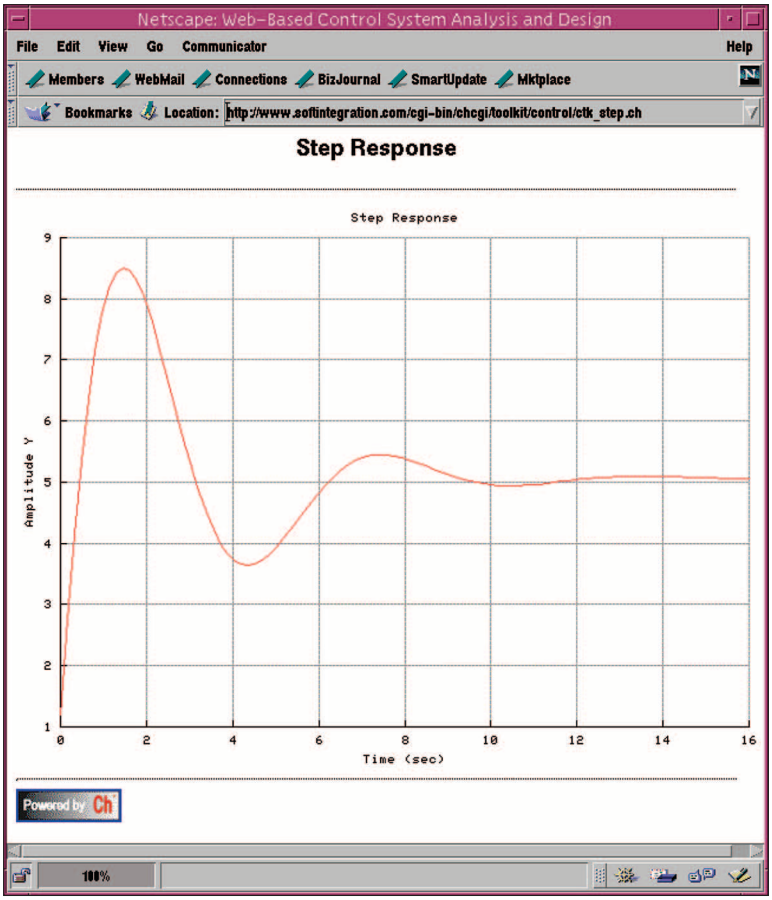

Figure 6. *The resulting output page (Form 4). This page shows the step response of the default or user-defined system.*

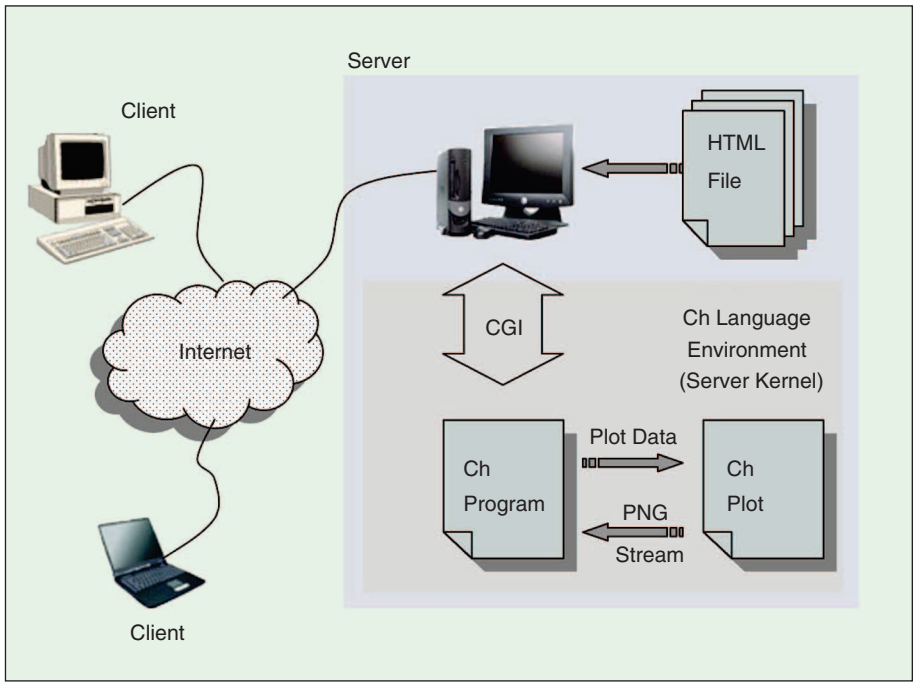

Figure 7. *The system architecture of WCDAS. The Ch CGI manages the communication between the HTTP server and Ch programs.*

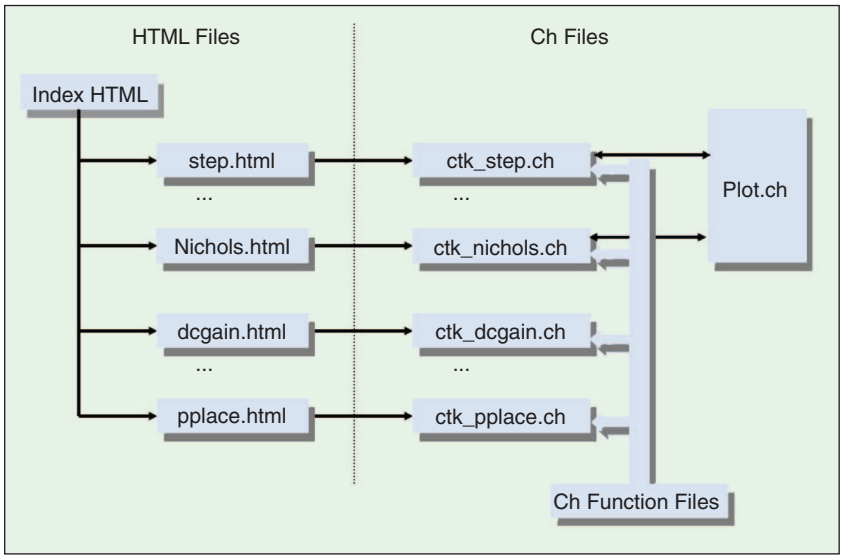

Figure 8. *The file system of WCDAS. Each function has its own HTML file and Ch CGI script for easy development and maintenance. Functions shared by multiple Ch scripts are treated as function files.*

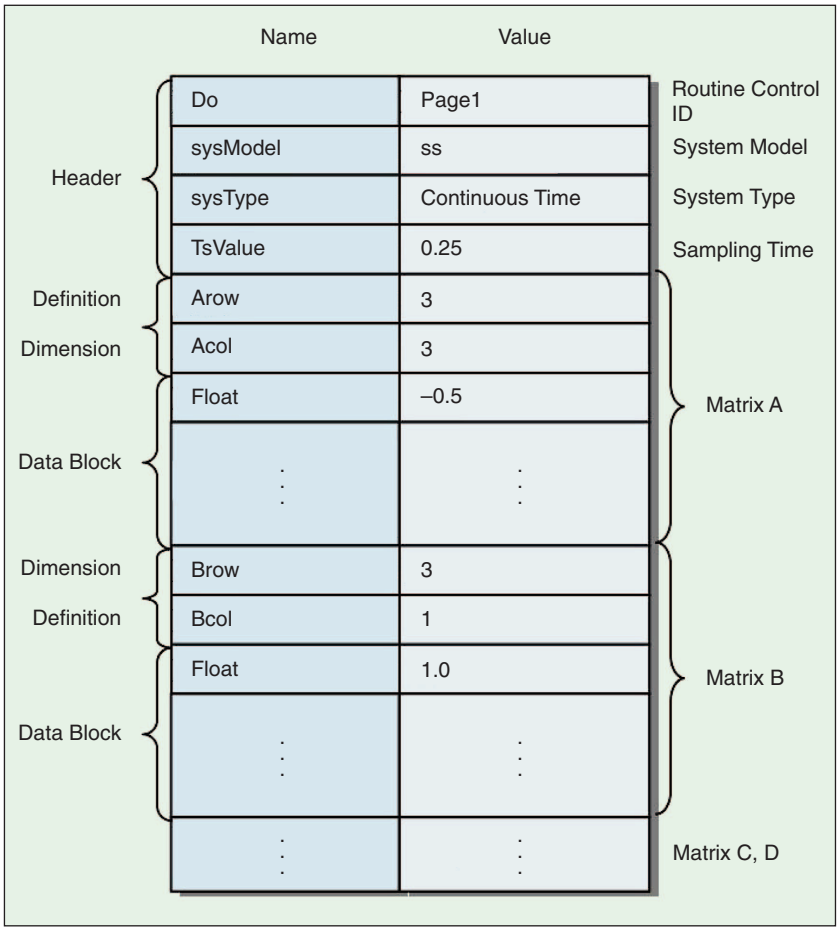

Figure 9. *The data structure for a state-space model. The data from HTML files are passed to Ch scripts based on a predefined data structure for different system models. The data structure used to represent a system in WCDAS is composed of data header, dimension definitions, and data blocks.*

time. The second part of the data structure defines the dimension of the matrices **A**, **B**, **C**, and **D** in (1) and (2). The third part of the data structure consists of the values of system matrices. For a transfer function or a zero-pole-gain model, the second part defines the order of the numerator and denominator polynomials of a transfer function or the number of zeros and poles. The third part defines the coefficients of the numerator and denominator polynomials or the values of zeros and poles.

To simplify the software implementation and to reduce code redundancy, functions that are used to access the structured data are implemented separately and shared by all CGI scripts. As an example, the function *read MatrixDim()* of Table 2 reads matrix dimensions from the structured data. In this program, the variables *name* and *value* are the arrays of strings with names and values from a fill-out form. The integer variable *num* contains the lengths of these two arrays. The return value is an array of eight integers with an index starting from one. The eight elements in the returned array stand for the extent of the rows and columns of matrices **A**, **B**, **C**, and **D** in sequence. The function *atoiCheck()* in Table 2 checks and converts the string of *value[i]* to an integer. If there is any error, such as an invalid input for an integer number, a Web page with an error message is generated.

Data Passing Between Pages

As discussed before, the user's model definition data are entered in multiple pages. Data passing between different pages is shown in Figure 10, where Forms 1–4 are defined in the previous section of the UI. The header of the data structure of the system, including system model type and system type, are specified in Form 1. This information is passed to Form 2 when the user clicks the "Continue" button in Form 1. If the user selects to run the default example in Form 2, WCDAS has all of the data and can generate the resulting output page directly. If the user selects to

define a new system in Form 2, the header information and the dimensions of the system matrices are passed to Form 3. Combining this information with the values of the system matrices entered in Form 3 generates a complete set of the data to represent a new system. All of the data are passed to Form 4 to generate the output response of the new system.

The Ch script *plot.ch* is invoked to generate a PNG image stream when a plot output is requested. The method for passing data to *plot.ch* is critical for Web-based plotting. If the data are passed improperly, the output might remain the same even if the user had submitted new system parameters. In our implementation, we pass the parameters along with the name of the CGI script. A unique name is created each time when the script *plot.ch* is invoked. Therefore, different results are generated for different submissions. The output plot is displayed on the Web page by the following code segment, where the member function of class Cserver *CServer::URLEncode()* is used to encode the data passed to the program:

```
printf ("<center><img src=\"/cgi-bin/chcgi/
toolkit/control/plot.ch");
for (i=0;i<num;i++){  /* pass data to plot.ch */
    putc (i==0 ? '?' : '&', stdout);
    fputs (Server.URLEncode(name[i]) ,stdout);
    putc ('=',stdout);
    fputs (Server.URLEncode(value[i]),stdout);
}
```

Handling Multiple Pages in One Ch Script

Forms 2–4 shown in Figure 2 are created by the same Ch script file *ctk_step.ch*. This Ch script program can handle three different pages. The value of the field named *Do* in the header of the data structure shown in Figure 9 informs this Ch script as to which page it is handling. The value of the variable *Do* should be one of the three values *Page1*, *Form 2*, or *Result*. The critical part of program *ctk_step.ch* is shown in Table 3.

The function *printFormHead* inside Table 3 displays the page ID, system model type, system type, and sampling time at the top of the form. The source code of this function is shown in Table 4. To simplify CGI programming, the Ch code between statements `fprintf stdout ≪ ENDPRINT` and `ENDPRINT` is transferred verbatim to an HTML file. The variable inside the parentheses following the symbol "$" is replaced by the actual value at runtime. For system interconnections, such as series interconnection of two subsystems, the two subsystems need to be identified with different model IDs. This identification is handled by two global variables *IDidx* and *modelID*. Variable *IDidx* has the value 0 or 1. Variable *modelID* is an array with two elements *sysModel* and *sys1Model* indicating systems 0 and 1, respectively.

Input Data Validation

For security and robustness reasons, WCDAS can handle unintended inputs and mistakes. Although the fill-out forms on the Web pages have greatly reduced the possibility of a user mistake compared to programming, we are vigilant about all inputs by checking all data passed by means of CGI. We try to be defensive in CGI programming.

Table 2. Read matrix dimensions from the structured data. This function reads the dimensions of the system matrices. The function is shared by all CGI scripts that need the dimension information of system matrices.

```
array int readMatrixDim(int num, chstrarray
        name, chstrarray value) [1:8]   {
    array int nvar[1:8];
    int i;
    for (i=0; i<num; i++)  {
    if(strcmp(name[i],"Arow")==0)
       nvar[1]= atoiCheck(value[i]);
    if(strcmp(name[i],"Acol")==0)
       nvar[2]= atoiCheck(value[i]);
    if(strcmp(name[i],"Brow")==0)
       nvar[3]= atoiCheck(value[i]);
    ...
    }
    return nvar;
}
```

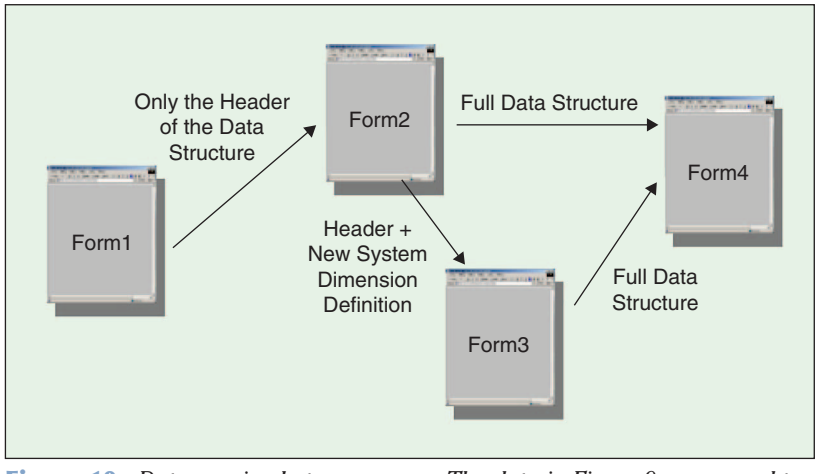

Figure 10. *Data passing between pages. The data in Figure 9 are passed to the next page.*

Table 3. The code segments of program ctk_step.ch. This program handles Web pages Form 2 to Form 4. Each form is processed by the corresponding function as shown below.

```
/* generate Form1 */
int processFirstPage(chstrarray name, chstrarray value) {
    /* part I: show default example */
    printFormHead("ctk\_step", "Result", value);  /* setup <Form> information */
    ...
    printRunButton("Run Example");

    /* part II: show user defined system input box   */
    printFormHead("ctk\_step", "Form2", value);  /* setup <Form> information */
    ...
}

/* generate a fill-out form ---Form3, for user to input each element for new system */
int processForm(int num, chstrarray name, chstrarray value) {
    printFormHead("ctk\_step", "Result", value);  /* setup <Form> information */
    ...
}

/* calculate and show result ---- Form4, plot if necessary. */
int processResult(int num, chstrarray name, chstrarray value) {
    ...
}

int main() {
    num =Request.getFormNameValue(name, value); // get CGI value
    /* value[0] contains variable Do value */
    ...
    /* "Page1" was passed from Form1 -- step.html */
    if(strcmp(value[0],"Page1")==0) {
        processFirstPage(name, value);      // show Form2
    }
    /* "Form2" was passed from function processFirstPage() . */
    else if(strcmp(value[0],"Form2")==0) {
        processForm(num, name, value);    // show Form3
    }
    /* "Result" was passed from both processFirstPage() and processForm() . */
    else if(strcmp(value[0],"Result")==0) {
        processResult(num, name, value);   // get result, show Form4
    }
    ...
}
```

Table 4. Print header of forms. In Ch CGI, the Ch code between statements `fprintf stdout << ENDPRINT` and `ENDPRINT` is printed out verbatim through the standard output stream.

```
int printFormHead(char *chFileName, char *doWhat, chstrarray value)  {
    fprintf stdout << ENDPRINT
        <FORM method="POST" action="/cgi-bin/chcgi/toolkit/control/$(chFileName).ch">
        <INPUT type="hidden" name="Do" value="$(doWhat)">
        <INPUT type="hidden" name="$(modelID[IDidx])" value="$(value[1])">
        <INPUT type="hidden" name="sysTypeStr" value="$(value[2])">
        <INPUT type="hidden" name="TsValue" value="$(value[3])">
    ENDPRINT
    return 0;
}
```

If an input value is not valid, an informative error message is displayed. As discussed in the section on data structures, the values of name fields for each element in the data structure indicate which type of data this element should be. The values of *Int*, *Float*, and *Complex* correspond to the int, double, and double complex data types in C. Passed data that cannot be converted to these data types are considered not to be valid.

Besides data type checking, additional validation checks are implemented. For example, if a user submits a model such that **A** is 3×3 and **B** is 2×1, the system gives an error message indicating that the dimensions of **A** and **B** do not match.

Application Examples

In this section, two sample applications are used to illustrate how control problems are solved using WCDAS.

Example 1

Consider the linear time-invariant, state-space control systems (1) and (2) with matrices

$$\mathbf{A} = \begin{bmatrix} -0.5 & -0.8 & 0.53 \\ 0.8 & 0.12 & 0.7 \\ -0.53 & -0.7 & -1.1 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1.0 \\ 0.8 \\ -1.5 \end{bmatrix}$$

$$\mathbf{C} = [2 \;\; 6.5 \;\; -2]$$

$$\mathbf{D} = [1.2].$$

To plot the step response of the system, we first click the "Step response" hyperlink in the index page shown in Figure 1. Then we select the system model for the state space equation and system type of continuous time in Figure 3. By clicking the "Continue" button, Form 2 is generated as shown in Figure 4. Next, we define a new system with matrix dimensions of 3×3 for **A**, 3×1 for **B**, 1×3 for **C**, and 1×1 for **D**. Next, by clicking the "Submit" button in Figure 4, Form 3 is brought up as shown in Figure 5, and we fill out each element with the given data. Finally, click the "Run" button to display the result as shown in Figure 6.

Example 2

A feedback system is shown in Figure 11, where the plant system 1 has two inputs and two outputs. The state-space matrices of systems 1 and 2 are given by

$$\mathbf{A1} = \begin{bmatrix} -1.2 & -2 \\ 2 & 1 \end{bmatrix}$$

$$\mathbf{B1} = \begin{bmatrix} -1 & -1 \\ 0 & 2 \end{bmatrix}$$

$$\mathbf{C1} = \begin{bmatrix} -1.7 & 4 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{D1} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$\mathbf{A2} = \begin{bmatrix} 2 & 2 & -0.5 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{B2} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{C2} = [0 \;\; 0.5 \;\; -0.5]$$

$$\mathbf{D2} = [0].$$

To find the state-space matrices of the feedback system we go to the index page shown in Figure 1 and click the

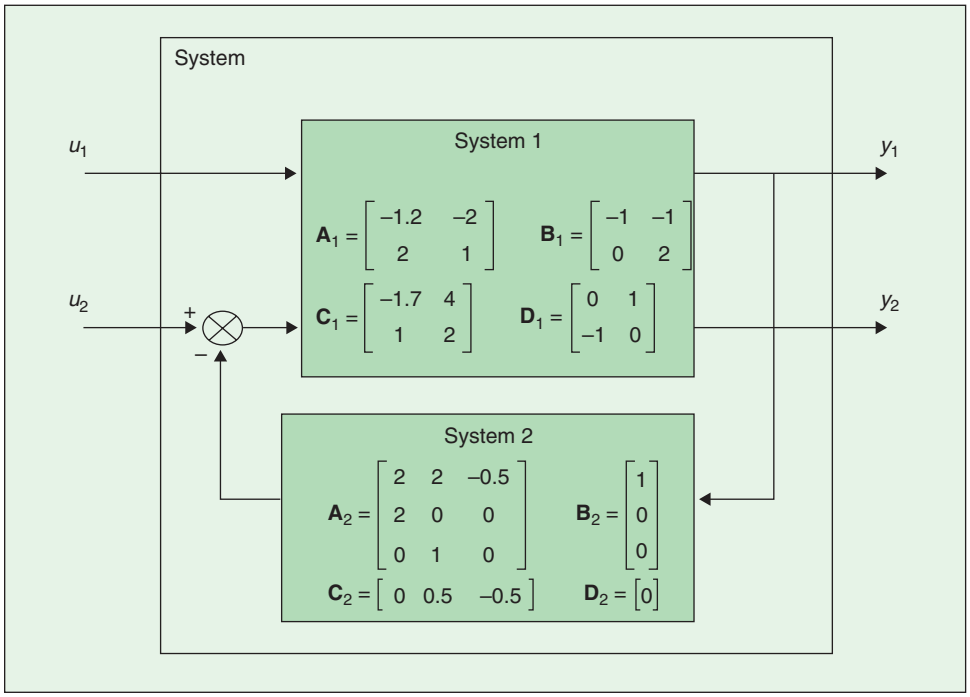

Figure 11. *A feedback system involving two subsystems. The plant subsystem is a MIMO system. The feedback controller is a SISO system.*

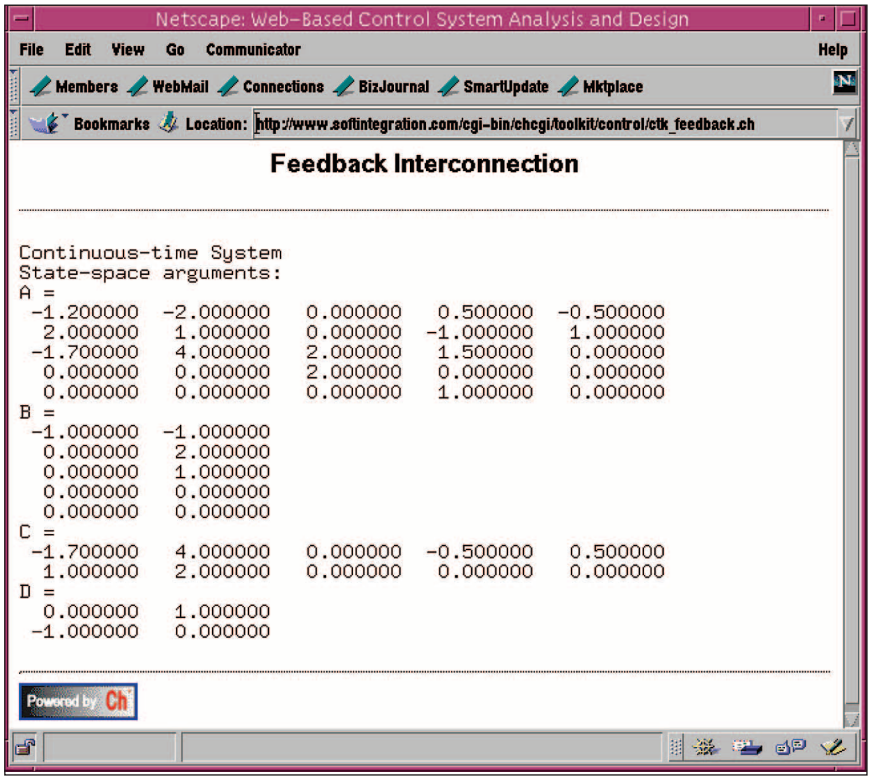

Figure 12. *The state space matrices of the feedback system shown in Figure 11. The feedback system is a MIMO system with two inputs and two outputs. The system has five state variables.*

"Feedback" hyperlink in the "System Interconnection" section. On the "Feedback Interconnection" Web page, the user selects system 1 and system 2 to be state-space models and specifies the dimensions of the state-space matrices. After entering the inputs and outputs of system 1, which are involved in the feedback loop, click the "Continue" button. On the next page, the user enters all values that are required for computation and clicks the "Run" button at the bottom of the page. The state space matrices of the feedback system are displayed as shown in Figure 12.

Conclusions

A Web-based control design and analysis system called WCDAS has been designed and implemented using the Ch language environment (a C/C++ interpreter), CCST, and CGI toolkit. Many commonly used functions of control systems design and analysis have been implemented in WCDAS. These functions can be applied to both continuous- and discrete-time linear time-invariant control systems modeled by state-space equations, transfer functions, or zero-pole-gain representations. WCDAS is easy to use and maintain. The design, implementation, and operation of the system were discussed. The user can access WCDAS [9] without any software installation, system configuration, or programming. The user can submit system parameters and receive the text or graphical results through a Web browser. This Web-based control design and analysis system is ideal for rapid prototyping, instructional use, and student learning, as well as for practical engineering applications. The feedback from users has been quite positive, and the convenience and ease of use of the system have

been universally appreciated. Ch, Ch Control System Toolkit, Ch CGI Toolkit, and WCDAS described in this article are freely available for academic use and can be downloaded from the Web [12]. Users can set up WCDAS on their own Web servers to avoid network traffic. The WCDAS and Ch Control System Toolkit, which has similar capabilities as the MATLAB Control System Toolbox, are open source. The ideas and principles presented in this article are applicable to many other areas of control systems. Users can examine the source code to extend the system with new features such as nonlinear control, random white noise analysis, and system identification.

References

[1] S.E. Poindexter and B.S. Heck, "Using the Web in your courses: What can you do? What should you do?" *IEEE Contr. Syst. Mag.*, vol. 1, pp. 83–92, Feb. 1999.

[2] G.J.C. Copinga, M.H.G. Verhaegen, and M.J.J.M. van de Ven, "Toward a web-based study support environment for teaching automatic control," *IEEE Contr. Syst. Mag.*, vol. 20, pp. 8–19, Aug. 2000.

[3] Virtual Control Lab, 2002. Available: http://www.esr.ruhr-uni-bochum.de/VCLab/

[4] J.C. Martinez-Garcia, G.H. Salazar-Silva, and R. Garrido, "Web-based object-oriented control system design," in *Proc. IEEE Int. Conf. Control Applicat.*, Mexico City, Mexico, Sept. 2001, pp. 111–116.

[5] The Virtual Control Lab for the ECOSSE Control HyperCourse. Available: http://www.chemeng.ed.ac.uk/ecosse/control/course/map/index.html

[6] J. Sanchez, F. Morilla, S. Dormido, J. Aranda, and P. Ruiperez, "Virtual and remote control labs using Java: A qualitative approach," *IEEE Contr. Syst. Mag.*, vol. 22, no. 2, pp. 8–10, Apr. 2002.

[7] R.P. Manchon, O.R. Garcia, R.P.N. Garcia, N.G. Aracil, and L.M.J. Garcia, "Remote lab for control applications using MATLAB," in *Proc. IFAC Workshop Internet-Based Control Educ.*, Madrid, Spain, Dec. 2001, pp. 121–126.

[8] J.L. Diez, M. Valles, A. Valera, and J.L. Navarro, "Remote industrial process control with MATLAB web server," in *Proc. IFAC Workshop Internet-Based Control Educ.*, Madrid, Spain, Dec. 2001, pp. 139–143.

[9] *Web-Based Control Design and Analysis System*, Softintegration, Inc., 2003 [Online]. Available: http://www.softintegration.com/webservices/control

[10] H.H. Cheng, "Scientific computing in the Ch programming language," *Sci. Prog.*, vol. 2, no. 3, pp. 49–75, 1993.

[11] *Ch Language Environment User's Guide*, Softintegration, Inc., 2003 [Online]. Available: http://www.softintegration.com

[12] *Ch Control System Toolkit User's Guide*, Softintegration, Inc., 2003 [Online]. Available: http://www.softintegration.com/products/toolkit/control

[13] Y. Zhu, B. Chen, and H.H. Cheng, "An object-based software package for interactive control system design and analysis," *ASME Trans. J. Computing Inform. Sci. Eng.*, vol. 3, no. 4, pp. 366–371, Dec. 2003.

[14] *International Standard: Programming Languages—C*, ISO/IEC, 1990.

[15] *International Standard: Programming Languages—C*, ISO/IEC, 1999.

[16] *The Ch Language Environment CGI ToolKit User's Guide*, Softintegration, Inc., 2003 [Online]. Available: http://www.softintegration.com/products/toolkit/cgi/

[17] *Embedded Ch*, Softintegration, Inc., 2003 [Online]. Available: http://www.softintegration.com/products/sdk/embedch/

[18] *Control System Toolbox User's Guide*. Natick, MA: MathWorks, 1998.

Qingcang Yu received his M.S. degree in electrical engineering from Zhejiang University, China, in 1990. From 2001 to 2003, he was a visiting scholar in the Department of Mechanical and Aeronautical Engineering, University of California, Davis. He is currently an associate professor in the Information and Electronics Department at the Zhejiang Institute of Science and Technology, China. He is also a Ph.D. candidate in computer analysis and comprehension at Zhejiang University. His research interests include Ch, computer-aided design and analysis, image processing and analysis, stereo vision, and mobile robot navigation.

Bo Chen received her M.S. degree in electrical engineering from the Zhejiang Institute of Science and Technology (ZIST) in China and joined the Department of Electrical Engineering at ZIST. She was an associate professor and the vice chair of the Department of Electrical Engineering when she left ZIST in 1999 and began working as a visiting scholar in the Integration Engineering Laboratory at the University of California, Davis. Currently, she is a Ph.D. candidate in the department of Mechanical and Aeronautical Engineering at the University of California, Davis. Her research focuses on computer-aided design and analysis, real-time and embedded control, multiagent systems, and Web technologies.

Harry H. Cheng (hhcheng@ucdavis.edu) is a professor and director of the Integration Engineering Laboratory in the department of Mechanical and Aeronautical Engineering at the University of California, Davis. His current research interests include engineering software design, Web technology and its applications in design and manufacturing, open-architecture mechatronic system integration, and intelligent transportation systems. He is the chief architect of Ch, a C/C++ interpreter for script computing. He has published over 90 technical papers and has one U.S. patent. He received his M.S. degree in mathematics in 1986 and his Ph.D. degree in mechanical engineering in 1989 from the University of Illinois at Chicago. He can be contacted at the Integration Engineering Laboratory, Department of Mechanical and Aeronautical Engineering, University of California. Davis, CA 95616 USA.