2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

A Web-Based Linear-Systems iLab

Gerardo Viedma, Isaac J. Dancy, and Kent H. Lundberg
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology, Cambridge, MA 02139
email: klund@mit.edu

Abstract— **This paper describes a web-based laboratory for students in courses on Feedback Systems. This project uses the iLab architecture, which provides a framework for remote-lab development and deployment, using a three-tiered client/broker/server architecture. This three-tiered approach simplifies the development of remote labs by providing reusable components for laboratory-administration functions.**

In the specific lab described here, students use a Java-based Lab Client to configure system parameters of a state-variable filter and submit jobs to the Lab Server. The Lab Server computer uses a dynamic signal analyzer to take frequency-response measurements of the configured filter.

I. Introduction

Remote laboratories are real equipment laboratories that can be operated and controlled remotely through an experiment interface [1]. A remote laboratory simplifies the logistics and requirements involved in conventional laboratory work, including scheduling of equipment, lab space, staffing, training, and safety. Students can conduct their experiments from any computer on their own schedule, instead of in a specialized laboratory on the staff's schedule. A remote laboratory can also provide for much more efficient sharing of expensive measurement equipment.

There are many approaches to the design of Internet-based remote laboratories for control education. Early systems required specialized platform-dependent software running at the client computer [2], [3], [4], [5]. Later approaches moved towards browser-enabled technologies for the client, including Java applets [6], static and dynamic HTML pages [7], and CGI scripts [8]. HTML-based solutions often result in thin clients with little processing abilities and rely heavily on server-side technologies such as CGI that tightly couple client and server development [9].

Most current designs employ Java-applet technology for the client environment, due to Java's processing abilities and platform independence. Many of these systems rely heavily on TCP/IP sockets for communication [10], [11]. Although an efficient means for client to server communication, sockets require client developers to grapple with a style of programming radically different from the object-oriented paradigms they are accustomed to.

The iLab architecture [12], [13] provides a framework for lab development and deployment. This approach differs from sockets-based solutions by hiding many of the details involved in network communication from the developer. This goal is achieved by using web-service technology, which provides an object-oriented interface to client/server communication based on traditional method calls that take place over HTTP.

In addition, the iLab architecture also alleviates the workload on teaching assistants and professors. Previous remote laboratory designs have wrestled with the provision of administrative services not specific to the laboratory. In doing so, it has been the tendency to include this kind of functionality at the server end along with the laboratory-specific services [6], [14]. In contrast, the iLab architecture decouples laboratory-specific operations related to running experiments from the more generic administrative tasks of user authentication, user authorization, group management, and results-storage functionality. The iLab architecture extends the client/server weblab topology by incorporating an additional third tier: the Service Broker, as shown in Figure 1. The Service Broker handles all administrative tasks, thus freeing the server machine (and its developers) from having to implement custom administrative solutions for each different weblab.

II. Software Architecture Overview

Our Feedback Systems iLab [15] integrates into iLab's Batched Experiment Architecture. All communication takes place via web services using SOAP as the communication protocol. The use of SOAP and web services allows us to make no assumptions regarding the platforms and programming languages used to implement the individual tiers. With this scheme, the first tier can be implemented in a different programming environment and run on a different operating system from either the second or third tiers. The only requirement for intercommunication is for each tier to conform to the published iLab experiment API. Moreover, all communication takes place over the HTTP protocol to which campus networks have access.

The three tiers in the Feedback Systems iLab are shown in Figure 2 [16]. The first tier is the **Lab Client** implemented as a Java applet running on the student's browser. The Lab Client provides a virtual interface to the lab equipment and experiment hardware by means of a GUI through which users can submit experiments and manipulate results. The middle tier is composed of the **Service Broker** providing the shared generic services for all iLabs. Among these common services are user authentication and registration, user authorization and credential management, as well as experiment specification and result storage. Finally the third tier consists of the **Lab Server**, where the user-submitted

0-7803-9098-9/05/$25.00 ©2005 AACC

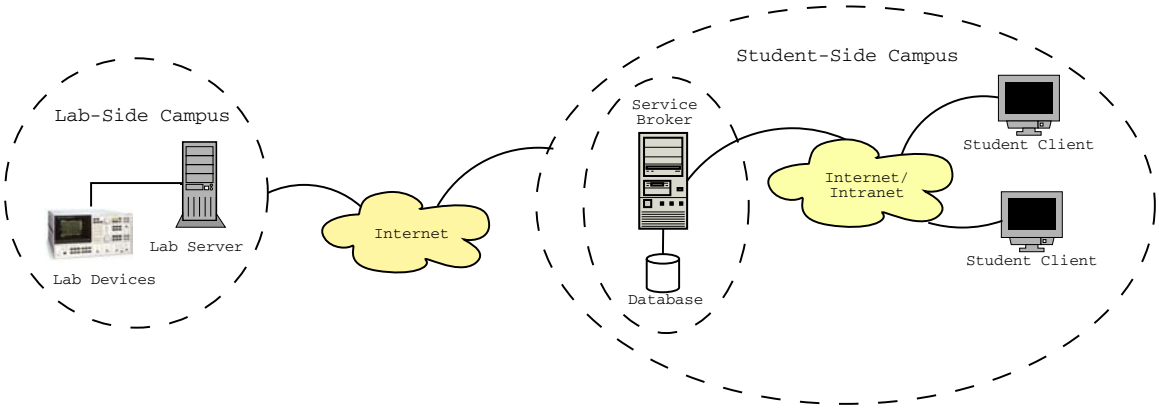

Fig. 1. Architectural overview of the three-tiered iLab system. The Service Broker handles all administrative tasks, thus freeing the server machine (and its developers) from having to implement custom administrative solutions for each different weblab. The Service Broker architecture also simplifies iLab sharing between universities by alleviating the lab-side (host) university from administering guest users. The host university can grant access to the student-side (guest) university's Service Broker, and the guest university can then administer its own users.

experiments are executed on the actual laboratory hardware including the system under test. Once an experiment has been successfully completed, the Lab Server notifies the Service Broker that results are available to be retrieved.

The three-tiered architecture has the additional advantage that all communication to the Lab Server must go through the Service Broker, thus making it the single point-of-contact. This scheme enables us to hide the hostname of the Lab Server, and lets us place the Lab Server behind a strict firewall, accepting only those connections originating from the trusted Service Broker host. As a result, we can have a world-accessible weblab without needing to expose our Lab Server to the world.

III. Describing Experimental Specifications

The iLab framework stipulates three different specifications for describing the experiment universe. The content of these specifications is unique to the Feedback Systems iLab, and provides a common understanding of the experiment world between Lab Client and Lab Server. In addition, the Feedback Systems iLab was designed so that the specifications describing experiments reside on the World Wide Web. Experiments can thus be modified remotely by the staff.

The three specifications defined in iLab are the *Lab Configuration*, the *Experiment Specification* and the *Experiment Result*. Our implementation introduces one additional specification: the *Experiment Routine*. In order to facilitate interoperability and the transfer of information across the Web, these specifications are encoded using the syntax of the Extensible Markup Language (XML) [17].

In a nutshell, each experiment is completely specified by an Experiment Routine maintained by the course staff and weblab administrators. This specification provides the set of experimental routines to run at the lab hardware connected to the Lab Server. It is also used to specify the inputs that make up the Lab Configuration, and that students provide when running their experiments from the Lab Client. The Experiment Specification consists of the experimental parameters specified by the user through the input fields at the Lab Client. The Lab Server then uses these experimental parameters to set up the experiment hardware appropriately when running the user's experiment request. Finally, upon successful completion of an experiment, the data vectors of the measured frequency, magnitude, and phase data are packaged into the Experiment Result at the Lab Server, and sent back to the Lab Client via the Service Broker.

IV. Lab Server Software

The Lab Server communicates with the Service Broker via an ASP .NET web-service interface running over Internet Information Services (IIS). The Lab Server also runs an SQL server accessed from the web services module for authentication and authorization, logging, etc. In addition, the database is used to enqueue experiment requests and to save any experimental results that were processed at the Lab Server.

The Lab Server also executes an experiment engine on its own thread, separate from the web services module. The experiment engine periodically checks the queue of submitted experiments, and retrieves the job with highest priority or the first one in the queue. It then processes the experiment specification provided by the user from the Lab Client, and configures the hardware appropriately. This step is accomplished by communicating with an HP 3562A digital signal analyzer over the GPIB bus, and whose probes are attached to the system under test.

V. Lab Client Software

The Lab Client GUI consists of three main components, as shown in Figure 3 [18]. First, the upper left side of the screen contains a number of text fields with their corresponding labels. Through these editable UI components, the user is able to configure the experiment by varying the value

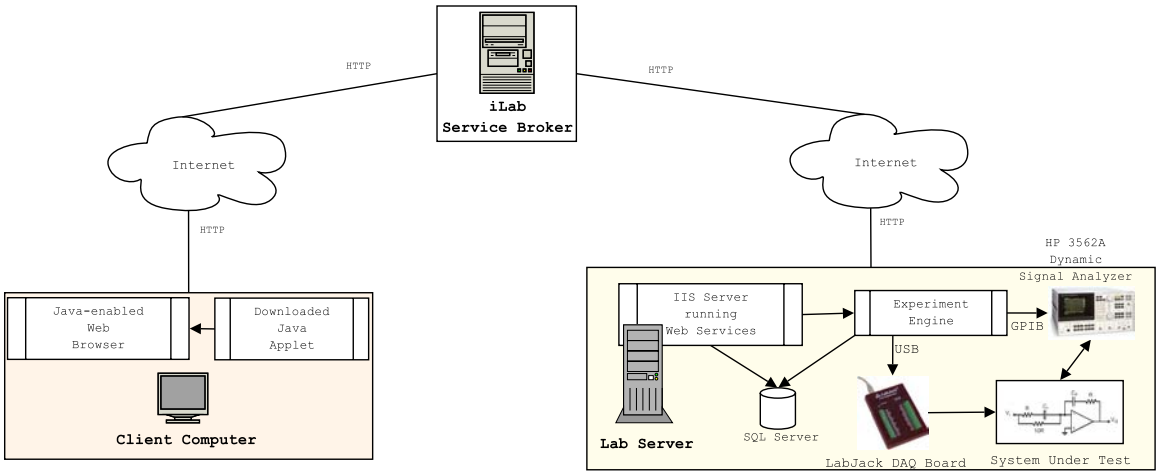

Fig. 2. Architectural overview of the Feedback Systems iLab. The Lab Client provides a virtual interface to the lab equipment and experiment hardware. The Service Broker provides the shared generic services for all iLabs, including user authentication and registration, user authorization, and credential management, as well as experiment specification and result storage. The Lab Server executes the specified experiments on the actual laboratory hardware including the system under test.

of the parameters that will be sent to the Lab Server in the Experiment Specification. The Lab Client applet parses the Lab Configuration provided to it by the Lab Server to dynamically construct these editable components.

Second, on the upper right side of the screen, an image representing the block diagram of the experiment is displayed. The URL of this schematic is also specified in the Lab Configuration and therefore can be easily updated.

Finally, the lower side of the applet contains a graph panel that displays Bode, Nichols, and Nyquist plots for the collected data. The graph panel enables users to interact directly with the displayed results, by allowing them to select and click on particular results to perform a number of tasks. The operations that may be performed on the data include exporting experiment results to a number of different formats, loading data saved from previous experiments, and deleting particular sets of results. In addition, the Feedback Systems iLab provides a simple mechanism for course administrators to publish theoretical or canned data for public consumption; students can then load these data directly on their Lab Clients for further analysis and comparison with measured results.

VI. Motivating the Laboratory Assignment

The first order of business in many control courses is to re-awaken students' familiarities with basic transfer functions and their behavior. This review is usually achieved with a barrage of pole-zero, step response, or Bode plot associations in homework and many examples in lecture.

The motivation behind this web-based laboratory on second-order-system responses is to bypass these expository details and give students an interactive and engaging way to review this important material.

VII. System-Under-Test Analysis

This weblab requires hardware that implements a variety of second-order systems, from lightly damped conjugate-pole pairs to over-damped negative-real-axis poles. The parameters of these systems is user-programmable via the Lab Server. To achieve a wide range of systems and results, two canonical second-order systems are cascaded, giving the user four degrees of freedom.

The experiment hardware is voltage-controlled in order to translate lab server commands into second-order system parameters. The state-variable-filter topology (block diagram shown in Figure 4) provides this functionality. An attractive feature of the state-variable filter is its realization from simple building blocks such as integrators, summers, and gain elements.

The closed-form transfer function of the state-variable filter in Figure 4 is

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{1}$$

which is exactly the canonical second-order transfer function. If the parameters ω_n and ζ are voltage-controlled, then this topology can implement any second-order frequency response. In fact, this topology is widely used in music-synthesis applications [19], [20] due to its broad capabilities.

The state-variable filter uses simple integrators, gain elements, and feedback to implement a variety of second-order responses. If the integrator and feedback gains are carefully controlled, then a wide range of responses can be realized. Figure 5 represents the overall topology of the cascaded state-variable-filter design. The AO_n signals are voltages provided by the Lab Server through the hardware interface. A frequency analyzer drives the input `IN` and measures the output response at the output `OUT`.

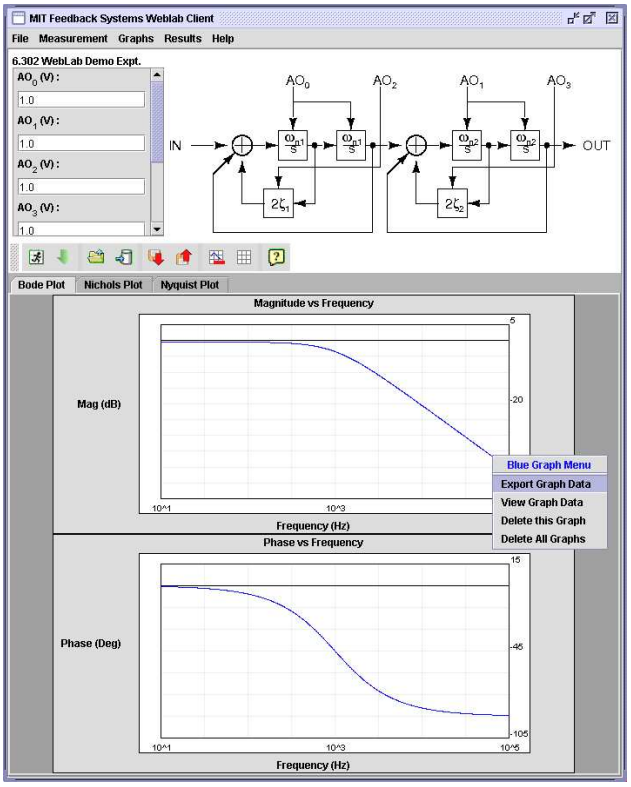

Fig. 3. Lab Client applet for the Feedback Systems iLab. The student configures the experiment by entering values in the upper-left text fields. The upper-right side of the screen displays a block diagram of the experiment. The lower frame of the applet contains a graph panel that displays Bode, Nichols, and Nyquist plots of the collected data.

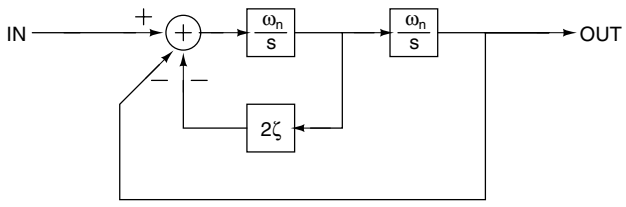

Fig. 4. State-variable filter topology. The closed-loop transfer function implements the canonical second-order system (1) using only integrators, a gain element, and a summer.

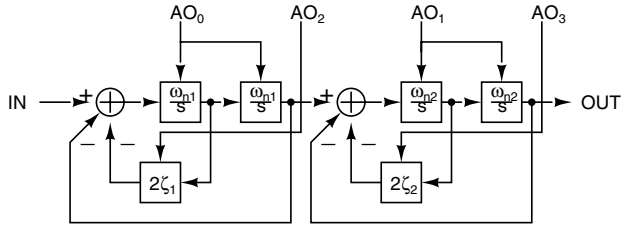

Fig. 5. The state-variable filter topology used in this lab. The analog outputs AO_n from the Lab Server control the gain of connected system blocks. This cascade provides four student-settable system poles.

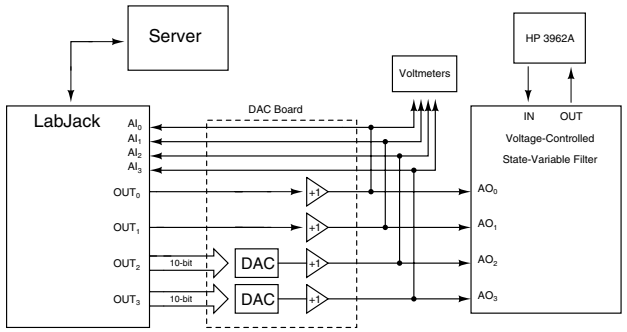

Fig. 6. Server-side hardware configuration. The Lab Server controls the LabJack via USB and the HP 3562A via the GPIB interface. The LabJack drives two 5-volt analog voltages and 20 lines of 5V TTL-compatible digital logic. These 20 digital lines drive two 10-bit DACs, yielding a total of four analog voltages with 10-bit resolution. The voltmeters provide administrators with command-signal diagnostics during testing.

VIII. Server/Hardware Interface

The Lab Server exists in a secured lab with the experiment hardware and communicates with the Service Broker to receive all client experimental parameters. The LabJack™ [21] connects to the Lab Server through the universal serial bus (USB) and applies the command signals to the system under test, as specified by the student.

The LabJack drives two 5-volt analog voltages and 20 lines of 5V TTL-compatible digital logic. These 20 digital lines drive two 10-bit DACs, yielding a total of four analog voltages with 10-bit resolution. An overall diagram of the experiment hardware is shown in Figure 6.

IX. System Circuit Design

The circuit design requires a voltage-controlled integrator and a voltage-controlled gain element.

A. Voltage-Controlled Integrator

Operational transconductance amplifiers (OTAs) are often used to implement variable-gain integrators [19], [20]. Unfortunately, the accuracy of this approach is limited by the linearity of the OTAs, which is insufficient for this application [22].

Alternatively, voltage multipliers can be used to implement the variable gain. Multipliers with good linearity can be found, though at a cost exceeding $29 per chip [23]. Fortunately, Analog Devices generously donated the AD532 voltage multipliers used in our hardware. The circuit diagram for an inverting integrator is shown in Figure 7. The input-output relation for this circuit is

$$\frac{v_O}{v_I} = \left(\frac{v_C}{10\ \mathrm{V}}\right)\frac{1}{RCs}, \tag{2}$$

where the voltage v_C is tuned to control the gain of the integrator.

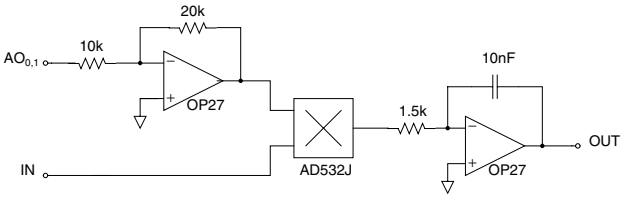

Fig. 7. Voltage-controlled inverting integrator. The amplifier with inverting gain of two scales $AO_{0,1}$ up to the full input range of the multiplier and results in a total noninverting relation from IN to OUT.

B. Voltage-Controlled Gain Element

A voltage-controlled gain element in the feedback paths of Figure 5 is implemented with another multiplier device, where $v_C/10$ V is the variable-gain parameter.

C. Final Circuit

These simple building blocks are used to implement the state-variable filter. The 5-volt analog signals are multiplied by two to exploit full dynamic range of the multiplier chips. The two voltages controlling the ω_n parameters are inverted to make the integrators noninverting. The final circuit schematic of the state-variable filter is illustrated in Figure 8. The complete implementation consists of two of these circuits cascaded in series to realize a greater variety of systems and assignment possibilities.

X. Results

Students are expected to evaluate the state-variable-filter topology in block-diagram and circuit form, obtaining relations between the voltage command signals and the corresponding block-diagram parameters. Students can then relate several second-order systems (as shown in Figure 9), express these systems in terms of their second-order parameters, and finally calculate the voltages required to make the experiment implement such systems.

While we expect that students will fine-tune their understanding of second-order systems, we also hope that they will make a few observations relating to the limitations of this specific design and implementation — problems inherent in any real hardware. For example, students should observe the difficulty in simulating the sharp and large-valued gains evident in lightly damped systems. Students should also observe the difficulty in discerning between similar systems — that is, four distinct, negative poles closely clustered in contrast to four poles at one, single location.

The system succeeds in producing smooth, accurate frequency responses and compares favorably to theoretical results. Figure 10 compares an experimental result to a theoretical result, and confirms this system's functionality.

For the purposes of this experiment, the state-variable filter proves to be a simple and robust solution. This weblab allows students to skip the tedious circuit-construction tasks involved with building a specific system, while still learning from the actual, measured results from a real-world system.

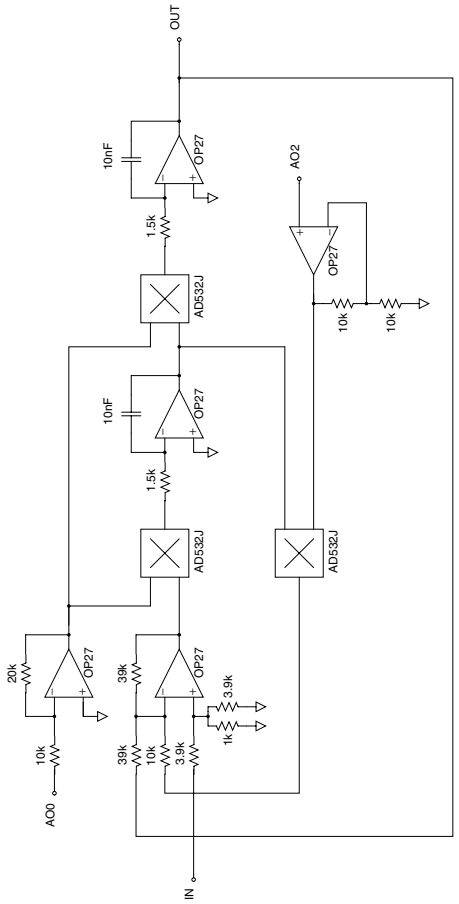

Fig. 8. Complete state-variable-filter circuit implementation. The laboratory system uses two of these circuits in series.

XI. Student Reactions

After completion of the WebLab assignments, students were surveyed regarding their experiences with the system. Students indicated a very positive experience with the ease of use and responsiveness of the WebLab. However, the results also show the students' discontent with the pedagogical effectiveness of the WebLab. For example, only 60 percent of those surveyed regarded the WebLab as an effective tool for experimenting with a real system.

Among the written responses, the student consensus was that the system was very easy to use, convenient, provided an intuitive interface to a real system, and responded quickly to user interaction. In students' words, these attributes made it easy to "quickly try out different things and see the result without the headache of setting up and troubleshooting lab equipment". A few students mentioned that the WebLab felt somewhat artificial, with "an interface basically indistinguishable from a good Matlab script". It is interesting to note that, in general, students who realized they were experimenting on a real system tended to rate the WebLab assignment high on pedagogical effectiveness.

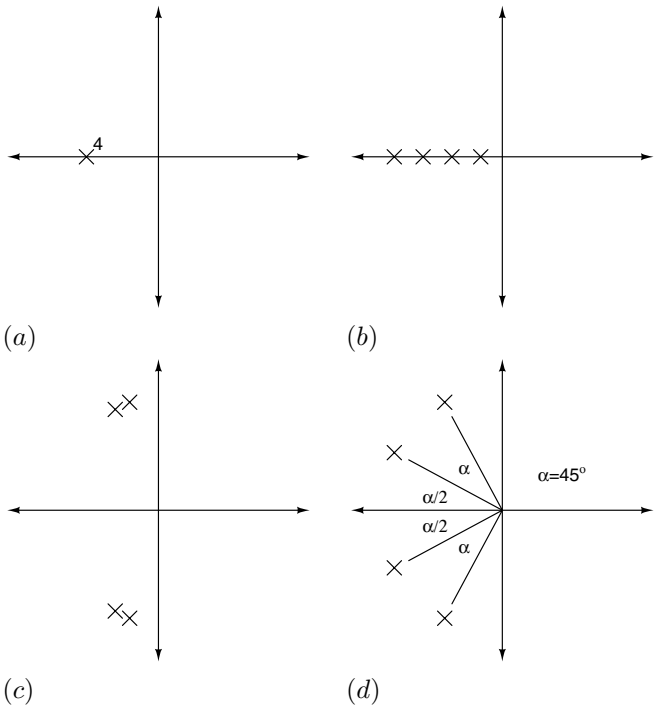

Fig. 9. Example assignment systems. The students are required to express these pole-zero plots in terms of their second-order parameters, and then calculate the voltages required to make the experiment implement the systems.

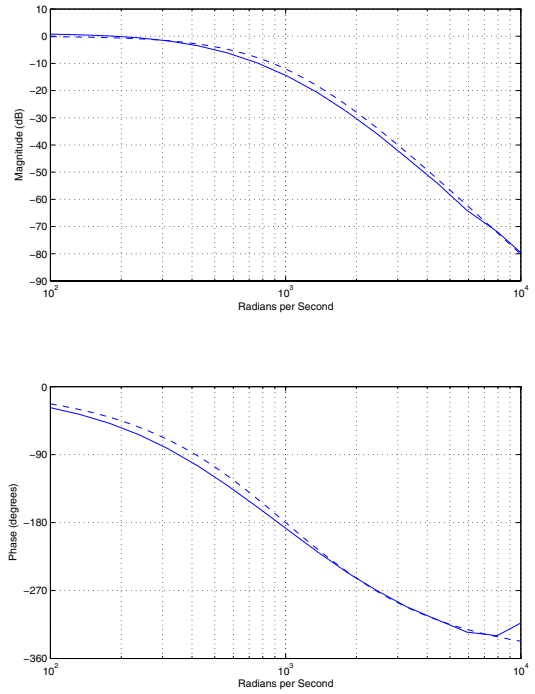

Fig. 10. Measured versus expected results for a typical four-pole system.

REFERENCES

[1] S. Dormido, "Control learning: Present and future," in *Proceedings of the IFAC 15th Triennial World Congress*, Barcelona, Spain, July 2002, pp. 81–103.

[2] M. Exel, S. Gentil, and D. Rey, "Simulation workshop and remote laboratory: Two web-based training approaches for control," in *Proceedings of the American Control Conference*, vol. 5, Chicago, IL, June 2000, pp. 3468–3472.

[3] D. A. Miele, B. Potsaid, and J. T. Wen, "An Internet-based remote laboratory for control education," in *Proceedings of the American Control Conference*, vol. 2, Arlington, VA, June 2001, pp. 1151–1152.

[4] M. Casini, D. Prattichizzo, and A. Vicino, "The automatic control telelab," *IEEE Control Systems Magazine*, vol. 24, no. 3, pp. 36–44, June 2004.

[5] D. Z. Deniz, A. Bulancak, and G. Özcan, "A novel approach to remote laboratories," in *ASEE/IEEE Frontiers in Education Conference*, vol. 1, Boulder, CO, Nov. 2003, pp. T3E–8–T3E–12.

[6] J. Sánchez, S. Dormido, R. Pastor, and F. Morilla, "A Java/Matlab-based environment for remote control system laboratories: Illustrated with an inverted pendulum," *IEEE Transactions on Education*, vol. 47, no. 3, pp. 321–329, Aug. 2004.

[7] A. Ferrero, S. Salicone, C. Bonora, and M. Parmigiani, "ReMLab: A Java-based remote, didactic measurement laboratory," *IEEE Transactions on Instrumentation and Measurement*, vol. 52, no. 3, pp. 710–715, June 2003.

[8] M. L. Corradini, G. Ippoliti, T. Leo, and S. Longhi, "An Internet based laboratory for control education," in *Proceedings of the 40th IEEE Conference on Decision and Control*, vol. 3, Orlando, FL, Dec. 2001, pp. 2833–2838.

[9] Q. Yu, B. Cheng, and H. H. Cheng, "Web-based control system design and analysis," *IEEE Control Systems Magazine*, vol. 24, no. 3, pp. 45–57, June 2004.

[10] C. C. Ko, B. M. Chen, J. Chen, Y. Zhuang, and K. C. Tan, "Development of a web-based laboratory for control experiments on a coupled tank apparatus," *IEEE Transactions on Education*, vol. 44, no. 1, pp. 76–86, Feb. 2001.

[11] H. H. Hahn and M. W. Spong, "Remote laboratories for control education," in *Proceedings of the 39th IEEE Conference on Decision and Control*, vol. 1, Sydney, Australia, Dec. 2000, pp. 895–900.

[12] S. Lerman and J. del Alamo, "iLab: Remote online laboratories." [Online]. Available: http://icampus.mit.edu/projects/iLab.shtml

[13] J. Harward *et al.*, "iLab: A scalable architecture for sharing online experiments," in *International Conference on Engineering Education*, Gainesville, FL, Oct. 2004.

[14] B. Aktan, C. A. Bohus, L. A. Crowl, and M. H. Shor, "Distance learning applied to control engineering laboratories," *IEEE Transactions on Education*, vol. 39, no. 3, pp. 320–326, Aug. 1996.

[15] K. H. Lundberg *et al.*, "6.302 iLab homepage: A WebLab for signals, systems, circuits, and control," Massachusetts Institute of Technology. [Online]. Available: http://web.mit.edu/6.302/www/weblab/

[16] G. Viedma, "Design and implementation of a feedback systems web laboratory prototype," Advanced Undergraduate Project, Massachusetts Institute of Technology, May 2004. [Online]. Available: http://web.mit.edu/6.302/www/weblab/

[17] "Extensible Markup Language (XML)," The World Wide Web Consortium (W3C). [Online]. Available: http://www.w3.org/XML/

[18] S. Lokanathan, "Extension to the feedback systems web laboratory client prototype," Advanced Undergraduate Project, Massachusetts Institute of Technology, July 2004. [Online]. Available: http://web.mit.edu/6.302/www/weblab/

[19] B. A. Hutchins, *Builder's Guide and Preferred Circuits Collection*. Ithaca, NY: Electronotes, 1986.

[20] H. Chamberlin, *Musical Applications of Microprocessors*, 2nd ed. Indianapolis: Hayden Books, 1985.

[21] "LabJack — USB-based data acquisition and control," LabJack Corporation. [Online]. Available: http://www.labjack.com

[22] R. Johnson, "Programmable state-variable filter design for a feedback systems web-based laboratory," Advanced Undergraduate Project, Massachusetts Institute of Technology, Feb. 2004. [Online]. Available: http://web.mit.edu/6.302/www/weblab/

[23] Digi-Key Corporation. [Online]. Available: http://www.digikey.com