USING MATLAB/SIMULINK
For Data Acquisition and Control

N.L. RICKER
University of Washington • Seattle, WA 98195-1750

Much has been published on process control education for chemical engineers. For example, an ongoing controversy regards the extent to which classical linear analysis, such as frequency response, should be included.[1] All would agree, however, that our undergraduates should have experience in applying control concepts to representative problems.

One possibility is to use computer simulations, which are increasingly powerful, affordable, and user-friendly.[1-4] An instructor can tailor simulations to illustrate key concepts of varying complexity. Experimental systems are relatively inflexible, and have safety, cost, and space constraints.

Simulations, however, leave some students cold[5]—an ideal curriculum would supplement them with lab experiments. One approach is to provide small-scale but realistic processes. Excellent examples include those described by Luyben,[6] and Lennox and Brisk,[7] which are relatively complex. We favor such experiments for our unit operations lab and in our elective advanced process control course where students have two to three weeks to complete a project.

We take a different approach in our required class on process dynamics and control (ChemE 480) which encompasses ten weeks—three hours of lecture and three hours of lab per week. The typical enrollment is fifty students. There are four lab sections, each one limited to sixteen students who work in teams. The lab portion employs eight identical experimental units, each of which interfaces to a PC (Pentium III, Windows NT). Professor Brad Holt designed the units to be inherently safe, self-contained (they require electrical power only), and sufficiently flexible to support a variety of simple experiments.[8] They are inexpensive and have operated for more than a decade with little maintenance.

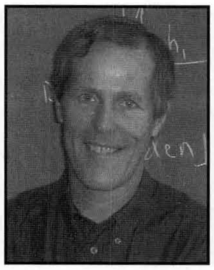
A key advantage is the ability to coordinate lab and lecture, even when enrollment is large. In a given week, all students perform the same experiment, which draws upon the most recent lectures. Table 1 shows the topics covered in 2000-2001. Weeks one, eight, and nine are simulations (using MATLAB and Simulink as described by Bequette, *et al.*[3]), but the others are experiments.

Our original lab computers were Macintosh systems, and we used Mac WorkBench[9] for data acquisition and control. WorkBench provided a student-friendly graphical interface for configuring control strategies. In 1998 we switched to Windows-based machines, however, and decided to standardize on LabVIEW,[10] which was being used in most of our research labs.

This worked well in the unit operations lab, where we were able to preprogram LabVIEW VIs, but it was a disaster in the control lab. Undergraduates found LabVIEW programming to involve a steep learning curve. Moreover, the LabVIEW graphical representation had little in common with the block diagrams found in process control texts—a pedagogical disadvantage.

Meanwhile, MATLAB and Simulink[11] were being used routinely for ChemE 480 and our reactor design course. Simulink was especially popular because of its intuitive graphical interface. Thus, when The Mathworks released their Data Acquisition (DAQ) Toolbox in 1999, we decided to test it as a LabVIEW replacement. We already had an educational site license for MATLAB and Simulink, and the incremental cost of the DAQ Toolbox was negligible.

Direct DAQ Toolbox use requires high-level MATLAB and object-oriented programming skills, however. We therefore packaged DAQ commands as Simulink objects, which could be configured graphically, as in a simulation. The standard Simulink Library blocks provided signal generation, signal processing, and display capabilities. We tested this approach for the first time in the fall of 2000. It quickly



Larry Ricker *received his BS in chemical engineering from Michigan in 1970 and an MS from Berkeley in 1972. He worked as a systems analyst for Air Products and Chemicals until 1975, when he returned to Berkeley and was awarded the PhD in 1978. He then joined the University of Washington faculty of chemical engineering, where he specializes in process monitoring, control, and optimization— with an emphasis on Model Predictive Control.*

became a mainstay of both the control and unit operations laboratories. In the control lab, for example, we found it easy to implement both standard feedback and advanced techniques, such as Model Predictive Control. Student reaction has been very favorable. The remaining sections describe this software, and illustrate some of the ways it can be used.

DATA ACQUISITION IN SIMULINK

Simulink is a simulation platform, not a real-time environment. To appreciate the advantages and limitations of our approach, one must first understand Simulink's simulation methodology. A Simulink diagram consists of interconnected *blocks* (see Figure 1). Each block can model a continuous system (such as CSTR), a discrete operation, or a hybrid of the two. *Signals* connect the blocks, and represent variables and parameters that change as a function of time. During a simulation, Simulink calls upon each block repeatedly (in a certain order) for the information needed to calculate the signals.

TABLE 1
ChemE 480 Lab Topics

Week	Topic
1	MATLAB tools for analysis and dynamic simulation. Modeling in Simulink.
2	Introduction to experimental system, data acquisition, sensor calibration.
3	First-order systems. Dynamics of liquid level in tanks with controlled inflow and gravity-driven outflow. Effect of tank geometry, outflow restriction.
4	First-order systems in series. Series connection of the two tanks studied in Week 3. Noninteracting and interacting configurations.
5	Proportional control of liquid level (series connection as in Week 4). Effect of controller gain on second-order response (damping, gain, offset, etc.).
6	PID control. Build a controller using Simulink blocks. Level control experiments. Controller tuning.
7	Frequency response. Direct sine-wave forcing of tank levels (first-order and second-order configurations). Pulse testing.
8	PID control of simulated "mystery" process (chosen from among several possibilities). Measure essential process response characteristics. Tune controller accordingly.
9	Cascade control of a simulated process. Process description given, but transfer functions unknown.
10	Feedforward-feedback control of level in second tank (series connection) with feed rate to first tank as measured disturbance.

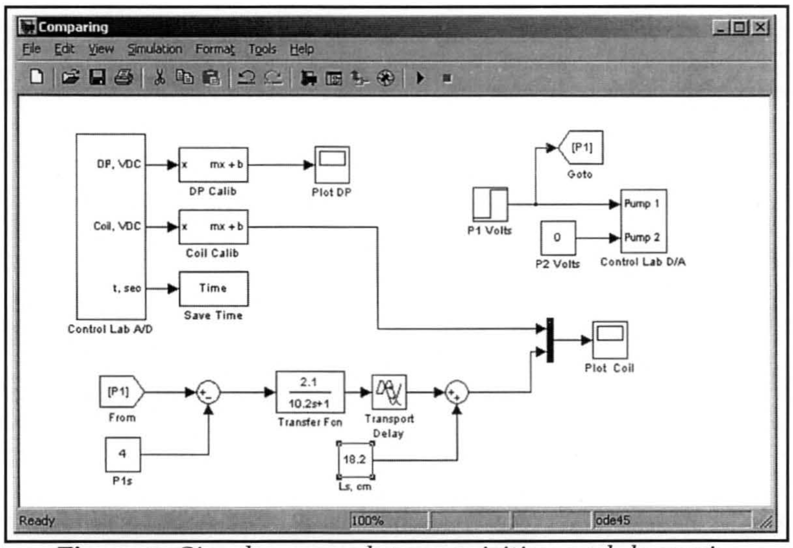


Figure 1. Simultaneous data acquisition and dynamic modeling using Simulink.

A Simulink block can include MATLAB code, so it is possible to use the MATLAB DAQ functions within Simulink. The main problem we faced is that the *simulated* time between successive block call varies (Simulink uses variable-time-step integration by default), and the elapsed *real* time between successive calls varies even more, depending, *e.g.*, on each block's complexity, the computing power available, and resource competition from other programs running simultaneously. Thus, the real time required to complete a sequence of block operations is unpredictable, which is incompatible with DAQ needs. For example, one usually wishes data to be acquired and control actions to be taken at a specified frequency.

On the other hand, modern computers are very powerful and can often simulate a complex system's response orders of magnitude faster than real time. Therefore, our approach was to slow Simulink down until it was closely aligned with real time. To do so, we developed two DAQ block types

- ► *An A/D block to sample an analog signal*
- ► *A D/A block to send an analog signal to the equipment. (We also developed a block to support data acquisition via a serial communication link.)*

These are discrete-operation blocks, which Simulink calls at specified time instants, t_k, where $t_k = k\Delta t$, k is an integer, and Δt is a specified sampling period. Note that t_k is measured in terms of the *simulated* time, *i.e.*, Simulink's independent variable. Each time a DAQ block executes, it checks the computer's real-time clock. If necessary, the block pauses the simulation until the real time catches up with the simulated time. It then executes its data acquisition and allows the simulation to continue. This cycle repeats until a specified run time elapses or until the user stops the experiment.

EXAMPLE: SIMULTANEOUS DAQ AND MODELING

Figure 1 shows an example from Week 3 of the ChemE 480 lab. The objective is to perform step-tests of a first-order system, comparing its response to a model. The physical system is the "short tank" in Figure 2 (next page), which is cylindrical, about 25 cm tall, and 18 cm in diameter. The student varies the flow rate via a 0-10 VDC signal that regulates a miniature variable-speed gear pump (labeled P1 in Figure 2). A sensor (not shown in Figure 2) returns a 0-10 VDC signal in proportion to the tank's liquid level. For this experiment, all valves except V2 are closed, and liquid leaves the tank by gravity-driven flow (through a central drain pipe, which has a series of small orifices drilled along its length).

Returning to Figure 1, the block labeled "Control Lab D/A" controls the pumps. Here, the student is sending a step function to pump P1, and a constant zero voltage to pump P2, which is to remain turned off. The D/A Block is a customized Simulink *mask*. When the student opens it, a dialog asks for the sampling period (typically one second). Hidden underneath the mask is our general-purpose D/A block, which we configured by specifying the type of hardware being used, the D/A channel numbers, etc. We have chosen to spare the students these details.

Similarly, the block labeled "Control Lab A/D" periodically samples the voltages coming from the unit's two level sensors. Again, the students need only specify the sampling period. The output from each sensor is going to a linear calibration block, which converts the voltage to a liquid level (centimeters). Note that in the prior week, the students configured the calibration block. The signal from the differential pressure sensor (DP) is irrelevant here. The "Coil" signal is measuring the step response, and the student is plotting it on a real-time display.

The student is also plotting the output of a first-order-plus-delay model, which is running in parallel with the experimental unit. Its input signal is the voltage being sent to the pump, from which the student is subtracting the constant labeled P1s (the steady-state pump voltage = 4.0 VDC). The resulting "deviation variable" feeds a standard Simulink *Transfer Fcn* block (in this case, a continuous-time, first-order system with a gain of 2.1 and a time constant of 10.2 seconds). The transfer function output goes to a standard *Transport Delay* block. Its output is a deviation variable, so the student adds the steady-state level (18.2 cm) to allow a direct comparison with the plotted experimental value.

Each run requires about two minutes, in order to allow the experimental system to reach the new steady-state. The real-time display focuses the student's attention on the speed and magnitude of the two responses. The student can change the model and rerun the experiment in order to observe the result. The data can also be saved to MATLAB (via Simulink's *To Workspace* block), for use with off-line data analysis tools. For example, one could use the System Identification Toolbox to estimate model parameters. Figure 3 shows the fit of a first-order-plus-delay model (solid line) to the response of an over-damped, second-order system (two tanks in series). Here, the sampling period is two seconds and for clarity we are showing every fourth data point only.

In subsequent weeks, the teams use standard Simulink blocks to configure and test feedback controllers, ranging from proportional to PID with anti-windup and derivative-on-measurement features, as described by Chung and Braatz.[12] The students apply these in feedforward/feedback and cascade combinations. Since they already know how to configure Simulink simulations, the transition to real-time control is easy and they can implement strategies of surprising sophistication.

For example, in Week 10 we run a contest to see which team can design a feedback/feedforward system providing the minimum integral absolute error (IAE) for a given disturbance. We make the problem more challenging by adding transport delay to the feedback loop. In each run, the students calculate the IAE and display it in real time using Simulink's absolute value, integration, and digital display blocks. This motivates them to investigate the reasons for large IAE values and to modify their strategy and tuning constants accordingly.

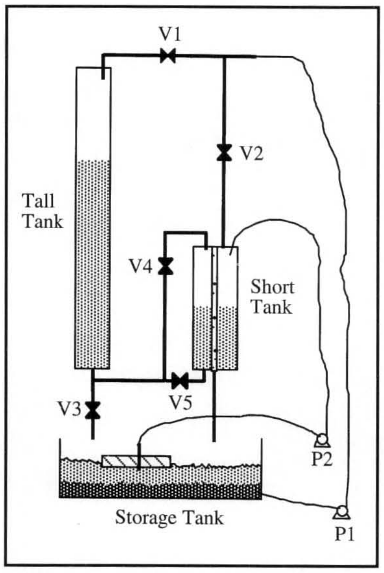


***Figure 2.** Schematic of the experimental unit.*

DISCUSSION

The Simulink platform also allows plug-and-play testing for modern control techniques. For example, Bemporad, *et al.*,[13] have developed an MPC block for Simulink. It was intended for simulations, but it provided excellent control of our lab unit. The software solved the MPC quadratic program in real time, consuming only a small fraction of the specified two-second sampling period. Installation was no more difficult than for any other Simulink block.

There are some potential disadvantages, however, including

▲ There is no guarantee that

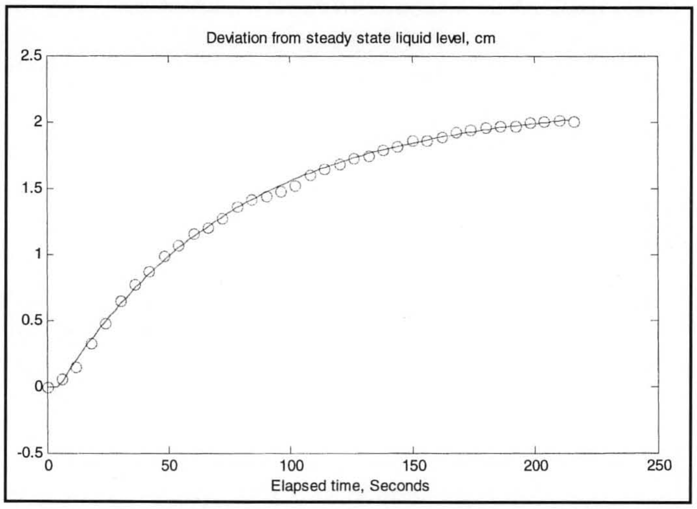


***Figure 3.** Off-line analysis of step-response data to identify a first-order-plus-delay model.*

data acquisition will occur at precisely spaced time intervals. In particular, Simulink initialization overhead causes the first sample to be delayed by as much as a second or two. A heavy computational burden—*e.g.*, another program running temporarily in the background—could also delay execution. Otherwise, however, DAQ actions typically occur within 0.05 seconds of the intended real time. The A/D block's time output (see Figure 1) allows the user to monitor the correspondence between real and simulated time.

▲ Output signals from the A/D block are piecewise constant. If the sampling period is small, relative to the dominant system time constants, the impact will be insignificant, but it could be an issue in certain applications. For example, a typical rule of thumb for sampled-data implementation of PID feedback control is that the sampling period should be less than five percent of the combined delay and dominant time constant.[14]

▲ Similarly, the D/A block updates its analog output at the sampling instants only.

We have used sampling periods as small as 0.5 seconds, but 1.0 second and greater is more realistic. Thus, our approach would be a poor choice for applications demanding >1 Hz (or those having safety issues!). Fortunately, most process control systems operate on a compatible time scale, and this has not been a problem in the ChemE 480 lab—even though we cover continuous systems only. In fact, students rarely notice that DAQ is discontinuous unless we point it out.

We also use the software in our follow-on (elective) control course, which covers sampled data systems. The students are then in a position to understand the impact of reduced sampling frequency and to experiment with this additional design parameter.

Our approach can also be used to provide real-time simulation. For example, one could include a "dummy" DAQ block in a simulation to synchronize it with real time, allowing a student to interact with the simulated process through Simulink input and display blocks. Such a real-time simulation could also be controlled by software residing on another computer, either via the DAQ signals, a serial link, or other means.

It's worth considering the more powerful commercially-available alternatives. For example, The MathWorks offers the "Real Time Workshop" and "xPC Target" packages that provide additional functionality and can operate at much higher sampling frequencies. A classroom license for these two packages costs about $88 per seat (compared to $18 for the DAQ Toolbox) and require a C++ compiler (about $50 per seat).

HARDWARE REQUIREMENTS

The DAQ Toolbox supports certain National Instruments, Agilent, and ComputerBoards hardware. See the MathWorks web site for up-to-date hardware compatibility information. We use the National Instruments PCI 6024E, which has 12-bit resolution, handles 16 (single-ended) analog inputs and two analog outputs, supports digital I/O, and listed for $595 in 2000. Data acquisition via serial communication requires only that the computer have one or more serial ports—the DAQ Toolbox is not needed.

SOFTWARE REQUIREMENTS

The Simulink DAQ blocks described here require MATLAB Version 6 (Release 12), Simulink Version 4 (Release 12), and DAQ Toolbox Version 2 (Release 12). The author, Professor Ricker (<ricker@u.washington.edu>), will provide the DAQ blocks and documentation for classroom use at no cost.

CONCLUSIONS

The Simulink-based data acquisition approach is a useful tool for undergraduate process control education. Students can perform simulations and work with physical systems from within a single software package, which is conceptually simpler, more flexible, and less expensive than a typical industrial control system. Sampling frequencies are limited to 1 Hz or lower, however. Commercially available alternatives provide faster sampling and other enhancements.

ACKNOWLEDGMENTS

ChemE 480 TA Mike Johnson helped with the initial software testing and provided much useful feedback. Professor Brad Holt was instrumental in setting up our control lab and promoting the use of a student-friendly graphical interface for data acquisition and control. Professor Frank Doyle, III, and the anonymous reviewers provided constructive comments on the first draft of this paper.

REFERENCES

1. Young, B.R., D.P. Mahoney, W.Y. Svrcek, "A Real-Time Approach to Process Control Education," *Chem. Eng. Ed.*, **34**(3), 278 (2000)
2. Cooper, D., and D. Dougherty, "A Training Simulator for Computer-Aided Process Control Education," *Chem. Eng. Ed.*, **34**(3), 252 (2000)
3. Bequette, B.W., K.D. Schott, V. Prasad, V. Natarajan, and R.R. Rao, "Case Study Projects in an Undergraduate Process Control Course," *Chem. Eng. Ed.*, **32**(3), 214 (1998)
4. Doyle, F.J. III, E.P. Gatzke, and R.S. Parker, *Process Control Modules*, Prentice-Hall PTR, New Jersey (2000)
5. White, S.R., and G.M. Bodner, "Evaluation of Computer-Simulation Experiments in a Senior-Level Capstone ChE Course," *Chem. Eng. Ed.*, **33**(1), 34 (1999)
6. Luyben, W.L., "A Feed-Effluent Heat Exchanger/Reactor Dynamic Control Laboratory Experiment," *Chem. Eng. Ed.*, **34**(1), 56 (2000)
7. Lennox, B., and M. Brisk, "Network Process Control Laboratory," *Chem. Eng. Ed.*, **32**(4), 314 (1998)
8. Holt, B.R., R. Pick, and T. Leach, "An Undergraduate Process Control Laboratory," *Proceedings of the IFAC Meeting on Advances in Control Education*, 197, Boston, MA (1991)
9. Strawberry Tree, a subsidiary of IOTech, Inc., <http://www.strawberrytree.com>
10. National Instruments, <http://www.ni.com>
11. The MathWorks <http://www.mathworks.com>
12. Chung, S.H., and R.D. Braatz, "Teaching Anti-Windup, Bumpless Transfer, and Split-Range Control," *Chem. Eng. Ed.*, **32**(3), 220 (1998)
13. Bemporad, A., M. Morari, and N.L. Ricker, "The MPC Simulink Library, Automatic Control Laboratory," *ETH, Report AUTO1-08* (2000)
14. Marlin, T.E., *Process Control*, 2nd ed., McGraw-Hill, 369 (2000) ❐