REAL-TIME, SENSOR-BASED COMPUTING IN THE LABORATORY

O.O. BADMUS, D. GRANT FISHER, SIRISH L. SHAH
University of Alberta • Edmonton, Alberta, Canada T6G 2G6

This paper is based on experiences in the Department of Chemical Engineering at the University of Alberta over the past thirty years. The original computer system (1966) was an IBM-1800 system with several hundred process input/output points connected to student and research labs throughout the building. Over the years, this system evolved into the current network of PCs and workstations running a variety of application software (such as Real-Time Matlab/Simulink,[1] Labview,[2] G2,[3]) in addition to user-developed software packages for process identification, multivariable optimal control, etc. This paper will describe:

- The experimental processes used in our undergraduate CPC laboratory
- The process instrumentation and process-computer interface
- The computer and communication systems
- Typical software: operating system, application, and process control

A complete description of all our applications is impossible because of space limitations, but sufficient detail will be given so that the reader can see what type of hardware and software is required for process applications of real-time, sensor-based (RTSB) computers and also to evaluate the importance of RTSB computing in the engineering curriculum and to engineers in industry. After the following brief overview, there will be more detail on the hardware and software, followed by a discussion of some applications.

Lanre Badmus *completed his BSc, MSc at the University of Lagos (Nigeria), and his PhD at Georgia Tech. He is currently a post-doctoral fellow and a sessional instructor at the University of Alberta working in the area of RTSB computer control and system identification.*
Grant Fisher *worked for Union Carbide Canada Ltd. for five years, completed his PhD at the University of Michigan, and has been teaching and doing research in the areas of process control and RTSB applications for the past thirty-two years.*
Sirish Shah *completed his MSc at UMIST (England), his PhD at the University of Alberta, worked at Esso Petroleum, and returned to teach at the University of Alberta in 1979. His principal teaching and research interests are in identification, control, and multivariable statistical analysis.*

PILOT PLANT PROCESSES

The process shown in Figure 1 is one of nine processes in the computer process control (CPC) lab. It was designed and built to provide students with "hands-on" experience with the type of equipment and processes found in introductory control textbooks[e.g.,4-5] and in the process industries. All processes are equipped with industrial instrumentation and PID controllers and are interfaced to PC-based RTSB computer systems for more advanced, user-defined applications. Thus, it is possible to run each process manually, with local industrial controllers and/or via computer software.

The three-tank process in Figure 1 can be configured to provide a wide variety of level, flow, and temperature processes that are familiar to most users and are reasonably easy to model using either first principles (*e.g.,* material and energy balances) or empirical process identification. Possible process configurations include first-, second-, and third-order systems, with self-regulating, integrating, non-minimum phase or non-linear characteristics with or without time delays. Possible control schemes include almost all the conventional (PID) feedback, feedforward, cascade, ratio, interaction compensation, etc., schemes described in textbooks and widely used in industry.[6] More advanced identification and control applications, some of which are described later, include discrete computer control algorithms, transfer function or state space identification, internal model control (IMC), generalized predictive control (GPC), and model based (SISO or MIMO) optimal constrained control algorithms (MOCCA) similar to the dynamic matrix control (DMC) algorithms widely used in industry.

PROCESS COMPUTER INTERFACES

Process computer interfaces are provided so that analog and/or digital signals can be read from, or sent to, each pilot-plant process using an industry-standard personal computer. The interface hardware in our CPC lab includes Opto-22 subsystems, National Instrument, and Computer Boards in-

terface cards (described below) chosen to provide the flexibility required to implement a wide range of undergraduate and graduate applications. They also illustrate important interface concepts such as handling low-level signals, different digital codes, differential analog inputs, anti-aliasing filters, etc.

Computer Software • Once again, the emphasis is on establishing a flexible environment that provides experience with basic concepts as well as with commercial products used in industry and in lab automation. The CPC lab includes a relatively large Bailey Network 90 system to provide experience with industrial distributed control systems (DCS), but most of the student lab and research work is done using the Matlab and Simulink Real-Time Workshop,[1] Labview,[2] G2 expert system development software,[3] or conventional programming languages such as Visual Basic or C++ running under operating systems such as QNX, DOS, Windows, UNIX, and OS2. As our later discussion will show, the application software systems and the user interface are the most important *functional* components for most students.

Figure 2 shows the signal transformations in a typical data acquisition and control system. It shows nine steps starting with data acquisition from analog sensors and ending with actuation of process equipment (*e.g.,* control valves) by output signals from the computer. In the figure, transmission of field signals is accomplished in the analog mode, while data analysis is performed in the digital mode. In light of the new Fieldbus technology[9] being rapidly adopted by industry, however, this may no longer be the rule-of-thumb. The advantage of analog signal transmission is its inherent infinite spectral content as compared to discrete (digitally) sampled data, which has a finite frequency spectrum. But analog signal transmission often involves lengthy, dedicated cabling, and high maintenance and overhead costs. The newest industrial impetus is toward digital signal transmission via fieldbuses, with smart digital sensors that implicitly (and automatically) perform signal conditioning, quantization, etc., in the field.

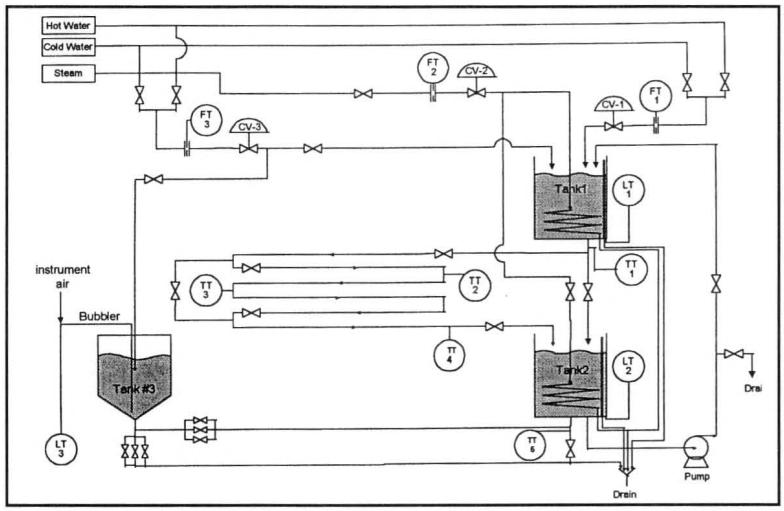

Figure 1. *Three-tank level plus temperature system (Instrument Trainer P-7)*

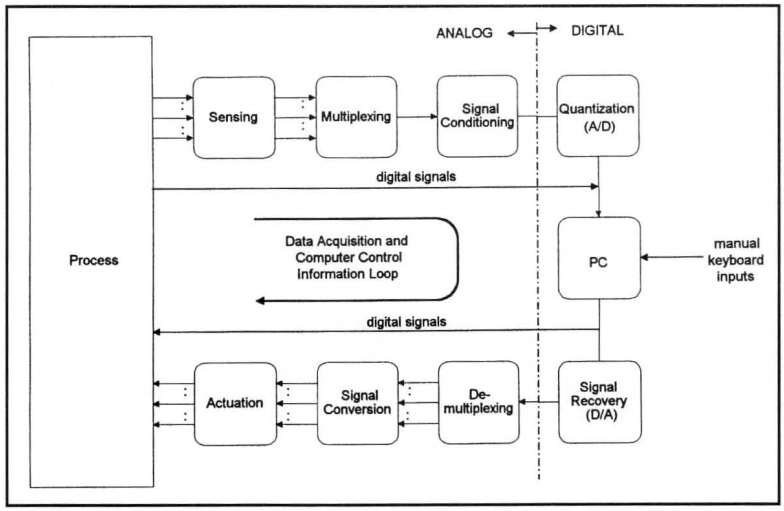

Figure 2. *Signal transformations in typical data acquisition and control system.*

RTSB SYSTEMS

In this section we will provide more detailed descriptions of the pilot plant process, computer interface, and computer hardware and software used in the UA/CPC RTSB systems.

A Pilot Plant Process for Control Applications • The three-tank level plus temperature system shown in Figure 1 is designed to be versatile and representative of typical chemical process plants. It consists of two cylindrical and one conical glass tanks, each having an internal diameter of 16 cm and a height of 50 cm. The system is equipped with several transducers (sensors): three differential pressure (d/p) cell transducers for measuring level (LT1, LT2, and LT3); three industrial flow transducers (FT1, FT2, and FT3) for measuring the flow rates of steam and water; and five thermocouples (TT1, TT2, TT3, TT4, and TT5) for measuring water temperature at various points in the system. In addition, there are three control valves (CV1, CV2, and CV3) for manipulating the flow rates of water and steam into the tanks. As mentioned in the overview, this process equipment can be readily configured to demonstrate the performance of different process systems and control schemes. All sensors and actuators are connected to the computer via the computer-process interface described below. Current-to-pressure (I/P) converters are used, where necessary, to con-

vert (4-20mA) current signals to (3-15psi) air pressure, *e.g.*, to actuate the control valves.

A large number of different first-, second-, and third-order processes and control configurations can be obtained by careful selection of process inputs and outputs. For example:

- The temperature control in tank 1 is a first-order process with delay (TT2, 3, or 4) or without delay (TT1). The 2 x 2 process (level via cold water flow, CV1, and temperature via steam flow, CV2) has "one-way interaction"[6] while the use of hot water in place of steam gives "two-way interaction" (a full transfer function matrix).
- A number of different first-, second-, or third-order level and temperature processes can be configured. Time-delays can be introduced into the temperature control system as above. Use of a pump rather than gravity for outlet water flow generates an integrating (unstable) process and control of level in the tapered region of tank 3 introduces *strong* nonlinearities.
- The interconnection between the sensors, actuators, computer interfaces, etc., is NOT fixed. The students use electrical cords with banana plugs and/or plastic instrument-air tubing with quick-disconnect fittings to configure each application, *e.g.*, simple data acquisition or feedback, feedforward, cascade, ratio, multiloop, multivariable, etc., control. The large number of alternatives overwhelms some beginning students, but *the fact that the students select from several realistic control strategies, must use different design methods, and actually physically implement (connect) the hardware is an extremely important feature. It significantly improves the educational aspects of the applications and is much more realistic from an industrial perspective than one in which the components are "hardwired" to perform a single function.*

Process Computer Interface • The data acquisition and control system shown functionally in Figure 2 is implemented as a PC-based, portable RTSB system that uses data acquisition boards that plug into the computer's I/O bus. The primary data acquisition board provides sixteen A/D channels and two D/A channels, along with thirty-two DIO (digital I/O) channels. There is also an auxiliary board that provides two additional D/A channels (see Appendix for details). There is also an additional integrated circuit board for thermocouple measurements that provides cold junction compensation (CJC), signal conditioning, open thermocouple detect (OTD), and lowpass filtering. A software driver controls the I/O operations between the interface and the user code (in Matlab/Simulink, Labview, or other conventional high-level programming languages). Our (four) RTSB computer systems are mounted on small carts and are versatile enough to be compatible with a number of laboratory pilot plants.

Four of the processes are equipped with an external (stand alone) data acquisition subsystem (OPTO-22) that can be linked to the PC via a standard RS-232 serial port connection. *This approach makes the interface more visible to the student, illustrates how data can be digitized "in the field," and has the advantage that it can be connected to any PC (including a laptop brought in by the user).*

Hardware Selection • The selection of an appropriate process-computer interface is *application specific,* that is, the appropriateness of a data acquisition board (and its signal processing components) depends not only on the process but also on the requirements of the application. The availability of a variety of computer-process interfaces with varied functionalities (including PC-cards for laptop computers) complicates the selection process, partly due to incompatibility of products from different vendors.

In general, the considerations for the selection of a data acquisition board for an application depend on three specific requirements—the process, the computer, and the data acquisition requirements:

- *Process Requirements* When a process has a single mode of operation with well-defined measurements (process outputs) and manipulated (process inputs) variables, it is straightforward to determine the number of analog input and output points required on the interface. In university applications, however, flexibility is important and a single RTSB system may be used for multiple applications. Therefore, our interfaces are larger than required for most student labs, have provision for a full range of analog and digital I/O connections, and include provision for low-level (*e.g.*, thermocouple) and/or differential analog inputs. (Differential inputs are more expensive, but are much better for handling common mode voltages, noisy signals, multiple grounds, etc.) Separate converters are used to handle the milliamp, voltage, pneumatic, pulse, etc., signal conversions associated with different sensors/actuators. *It is extremely important to provide electrical "isolation" (e.g., optical) between the process and the computer, plus overload/electrical-shorting, etc., protection, especially for applications where students configure their own applications.*
- *Computer Requirements* A desktop PC is commonly used because it is cost effective, versatile, and reliable enough for most university applications. The recent offering of PCMCIA cards that can handle process input/output signals has further reinforced the concept of *portable data acquisition platforms.* A laptop computer has the portability edge and may be preferred for field studies, classroom demonstrations, or in situations where students have their own laptop. It should be noted that many popular operating systems (*e.g.*, Windows) and some application software packages are not "real-time" in the strict definition of the term (*e.g.*, cannot handle multiple, high-speed, pre-emptive interrupts), but are adequate for most single-user, student lab applications.
- *Data Acquisition Board Requirements* The primary considerations when purchasing a data acquisition board are (a) user preference, (b) accuracy, and (c) sampling rate. The **user preference** is exercised in making the choice between a plug-in board and a standalone subsystem. Plug-in cards are less expensive, have a high data throughput, and use up less space (since they are resident in the PC). The standalone subsystems, on the other hand, are easily expandable (since they can be

daisy-chained to one another), do not require expansion slots, and are built to be rugged anough to withstand industrial environments. High data throughput can also be achieved in standalone data acquisition boards that use parallel or proprietary interfaces (as opposed to serial interfaces).

Considerations regarding the accuracy of a data acquisition board come about when specifying the acceptable resolution or number of bits used to represent a process signal, *e.g.,* a 12-bit analog-to-digital converter (ADC) has 4096 (or 2^{12} quantization levels). A 12-bit ADC applied with 0-5Vdc range signal and a gain of two has a code width of 0.61mV/bit. The value 0.61mV is the signal value represented by 1 bit, the *Least Significant Bit,* and can be interpreted as the sensitivity of the system. For more critical applications it is also important to consider the settling time, drift, temperature sensitivity, etc., of the converters, amplifiers, multiplexers, etc.

The sampling rate and anti-aliasing filters for the analog signals are also important specifications. The Nyquist theorem stipulates that sampling should be done at a rate that is more than twice the process bandwidth, *i.e.,* >2 samples per period of the highest frequency of interest. This is not a limitation in most chemical process applications because of their relative slow response rates. Analog anti-aliasing filters are required to prevent high frequencies (*e.g,* noise) from "folding over" and distorting the lower frequency process measurements.

Computer Software • Many software packages currently exist that facilitate the implementation of data acquisition and control algorithms in real-time applications.[8] Note the difference between packages that perform a specific function with a minimum of user-effort (*e.g.,* simple specification of parameters or options with no user programming) versus those that facilitate the development of a user-designed and implemented application. Student-computer interfaces and data processing algorithms are usually application and hardware specific and hence are best developed by the students using a *program development environment* provided by a commercial package (*e.g.,* Matlab, Labview, G2, C-compilers, etc.) plus libraries of numerical and signal processing algorithms. "Low-level software" such as drivers for the process I/O interface are much more difficult to write because they require detailed knowledge of the operating system plus the computing and interface hardware. *Therefore, it is strongly recommended that hardware (e.g., computer interface boards) be selected that has flexible, reliable drivers supplied by the hardware supplier or third-party software developers. Note that the drivers do not always support all the features contained in the hardware or the functional compatibilities desired by the user.*

- *Graphics and Presentation (GUI) Capabilities* The visual display and presentation of variables are a critical part of RTSB applications. There are often a large number of process variables, *e.g.,* temperature values, pressure values, valve positions, etc., that the operator must constantly monitor and compare with their limiting or constraint values. Hence, data acquisition and control software must provide an effective computer/user interface. (Note the important differences between software that can display/plot data in real-time as it is received rather than handling file data after a run is completed.) The Labview software, for example, offers extensive libraries of "virtual instruments" for operator inputs, displays, signal processing, etc., that can be selected from function libraries and arranged and interconnected graphically to construct a RTSB computer-user interface. The authors' experience with the current versions of the software is that real-time Matlab/Simulink is ideal for applications where the emphasis is on individual user implementations of data processing or control applications and is the most popular choice for theses and course projects. Labview, however, provides more insight and options for users developing industrial-type interfaces (especially for use by others) and is therefore used in our RTSB applications courses. The most important factors in selecting the application software are normally the *user requirements* (*e.g.,* a quick, convenient way to do a single thesis or lab implementation versus a vehicle to develop a more industrial-type system for use by others) and *user experience* (*e.g.,* are they already familiar with some software from other courses). As shown by the later examples, if the hardware and software drivers have already been developed and installed, then all a user has to do to change a simulation into an experimental application is to substitute the appropriate data I/O blocks into the Matlab/Simulink diagram in place of the process simulation blocks.
- *Analysis and Design Capabilities* The application development software should provide an efficient computing environment along with a suite of common process control functions and utilities. The availability of function libraries for signal processing, statistical evaluations, frequency analysis, array data manipulations, etc., greatly enhances the utility of the software for real-time control applications. Due to the specific requirements of most RTSB applications and/or the desirability of students using software developed in other courses, it is desirable to be able to include user-written functions developed in conventional high-level programming, *e.g.,* Basic, C/C++, FORTRAN, etc. This can be more difficult than one would expect because it requires acquiring an understanding of the application software structure, interfacing (data passing) conventions, timing considerations, etc., in order to successfully integrate the user's code with the application software. In our view, the capability of easily integrating user-written functions and subprograms into the application software is essential for student labs.

APPLICATIONS

The specific applications discussed below assume that a process similar to that in Figure 1 is used along with a process interface (Figure 2 and Appendix) and real-time Matlab/Simulink software. Similar applications, however, could also be done with other hardware/software. Details will depend on the actual requirements, but most applications will include the following steps:

- Build a Simulink diagram that will do the necessary data I/O, calculations, and display.
- Code (in C) the sections of the algorithm that are not readily generated from Simulink blocks, *e.g.,* For-loops. Then compile the C-code as MEX files and integrate them into

Simulink as S-function blocks.

- Set the real-time options and build the real-time code (these are selections on the pull-down menu 'Code' in the Simulink window).
- Verify that all I/O connections and hardware options are correct.
- Test the generated EXE file and then implement it on the experimental process.
- Perform the necessary on-line (*e.g.,* control) and off-line (*e.g.,* file processing) operations.

The following five application examples demonstrate specific features and advantages of RTSB computer applications. They range from simple data logging to multivariable predictive control. In addition, each example is illustrated with a different subsystem of the process equipment (in Figure 1). Because the Matlab/Simulink Real-Time Workshop[1] lacks a convenient user-accessible timing routine, applications were run in a quasi-real-time mode in which the timing operations were controlled from a user 'timer.m' file. The simplest implementation in this mode is to halt the entire system for the duration of the "wait" between control calculations rather than using multitasking or interrupts. This is obviously realistic only when using a dedicated PC and when the sampling period is significantly larger than the computation time.

Open Loop Data Acquisition • Figure 3 shows the Simulink block diagrams developed for a simple data logging and display application. The process response (level measurement, LT-1) to a step input change in the inlet water flow rate (FT-1) is recorded and plotted as a function of time. The data can be used for process identification, comparison of data processing options (*e.g.,* filtering, spectral analysis), comparison with simulations, fault detection, etc. For example, this step response data can be used to compute a first-order continuous transfer function model for the process

$$G(s)=\frac{ke^{-\theta s}}{\tau s+1} \qquad \text{where } k=6.67,\ \tau=240s,\ \theta=100$$

Students in courses such as Statistics and ChE Unit Operations were more highly motivated and learned better when required to work interactively with RTSB data than when hypothetical data were used as part of a homework problem.

Multiloop PID Control • Figure 4 illustrates the implementation of multiloop PID control of the process level and temperature using the manipulated variables—inlet water flow rate (FT-1) and steam (FT-2) respectively. This process exhibits a triangular (or one-way) interaction, in the sense that a change in the inlet water flow affects both level and temperature, while a change in the steam flowrate influences only the temperature and not the level. *The main advantage of an RTSB application like this is not in obtaining a single set of data as shown in Figure 4, but the ease of implementing and evaluating different control strategies (e.g., the advantages of feedforward); different control algorithms or PID-tuning methodologies; data processing options (e.g., filtering); or different model-based control strategies.*

System Identification • Figure 5 compares the measured process variables (temperature and level) with the corresponding simulated process variables. The simulated process outputs are from a

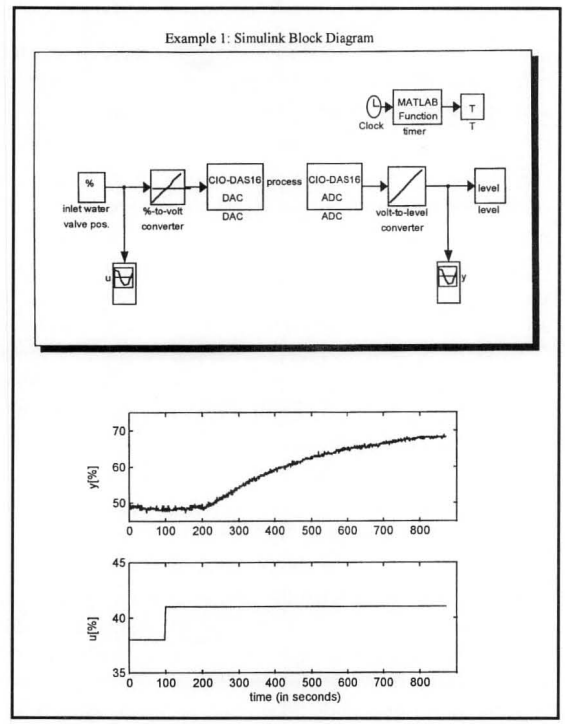

Figure 3. *Open-loop data acquisition (data logging): process response (level measurements) to an input step change in inlet water flowrate.*

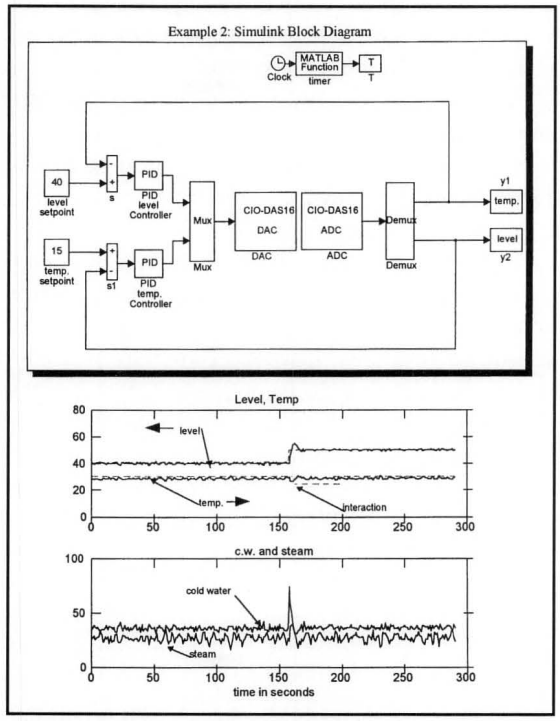

Figure 4. *Multiloop control of process level and temperature using inlet water flowrate and steam flowrate, respectively. A step change in level setpoint causes a disturbance in temperature that is quickly rejected by the temperature loop controller.*

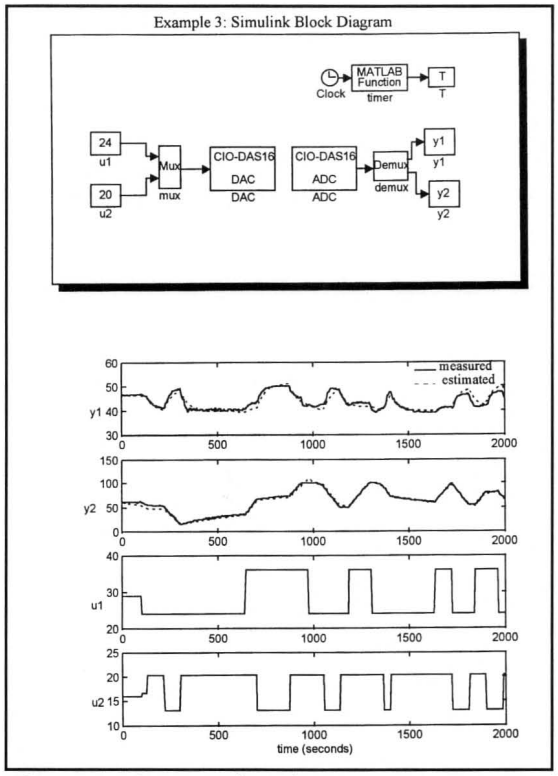

Figure 5. *System identification; comparison of the process level and temperature measurements with corresponding simulated process variables using an MVSSID[7] identified state-space model.*

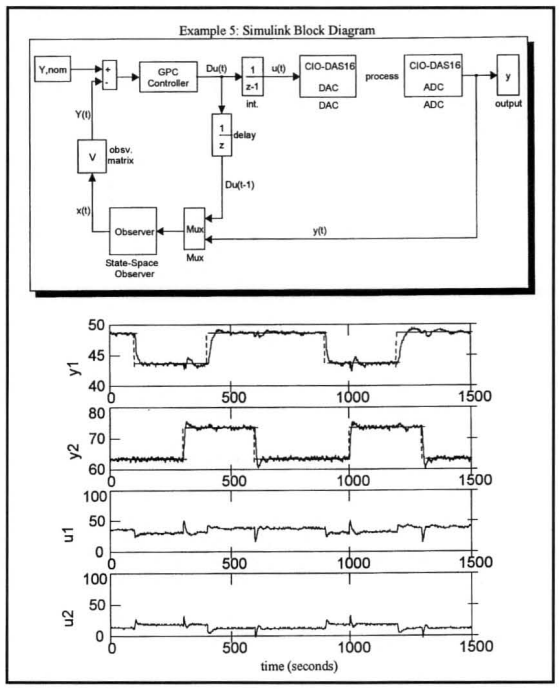

Figure 6. *Multivariable state space generalized predictive controller (tuning parameters: prediction horizon=10; control horizon=2; input suppression parameters=[0.05 0.05].*

state-space model generated with the identification package MVSSID.[7] The package is an identification software program that has been developed to generate multivariable state-space models from discrete input-output process data. The generated state-space model for the I/O data in Figure 5 is

$$\begin{bmatrix} x_{t+1}^1 \\ x_{t+1}^2 \end{bmatrix} = \begin{bmatrix} 0.98357 & -0.02175 \\ 0.00086 & 0.87232 \end{bmatrix} \begin{bmatrix} x_t^1 \\ x_t^2 \end{bmatrix} + \begin{bmatrix} -0.55116 & -0.70521 \\ -0.03065 & 0.32403 \end{bmatrix} \begin{bmatrix} u_t^1 \\ u_t^2 \end{bmatrix}$$

$$\begin{bmatrix} y_t^1 \\ y_t^2 \end{bmatrix} = \begin{bmatrix} 0.00313 & -0.55015 \\ -0.46075 & -0.03016 \end{bmatrix} \begin{bmatrix} x_t^1 \\ x_t^2 \end{bmatrix}$$

Once again, it is easy to compare different identification algorithms, different model structures, the effect of input excitation, and/or data processing options. It is particularly instructive for students to plot the simulated and experimental data in the same figure and to explore the significance of assumptions made during the derivation of the model.

Multivariable GPC Implementation • Figure 6 illustrates the real-time implementation of a multivariable generalized predictive controller. The process variables are the temperature and level of water in tank 1 (Figure 1) by manipulating the inlet hot water and inlet cold water flowrates. This subsystem forms a full (and sometimes highly) interacting model since a change in either of the manipulated variables influences both of the process variables. Note that the Simulink diagram necessary to implement this experiment is easily interpreted (or even developed) by users who are not familiar with the details or theory "hidden" inside the functional blocks, *e.g.,* the GPC controller.

The control algorithm is just an example of how RTSB computer systems can be used to demonstrate and evaluate course material (in this case, a first-level graduate control course). Obviously, a similar approach could be used in conjunction with courses focusing on optimization, expert systems, numerical analysis, statistics, unit operations, etc.

CONCLUSIONS

Specific control applications using personal computers, commercial process-computer interfaces, simple processes, and Matlab/Simulink software are described in enough detail that they can be replicated by others. The real purpose of this paper, however, is to demonstrate the importance of RTSB computing and how it can be easily and effectively integrated into university student laboratories. The details of any application should vary with each combination of course, instructor, student, and hardware/software system. Therefore, the authors strongly recommend

- ► The development of flexible, mobile RTSB systems that can be easily interfaced to any experiment the students are required, or can be motivated, to do.
- ► A focus on fundamentals, theory, and concepts rather than on the operation of a specific piece of hardware or on completing a series

Continued on page 289.

exclusion chromatograms of two or three proteins in a mixture at pilot (1-gram) scale using three combinations of column dimensions, elution gradient, and elution flow rate. Scale this process to a specified level of production and resolution using Yamamoto's principles.[8]

5. **Electrophoresis** Purify milligram quantities of an oligonucleotide from a mixture of in vitro transcripts and evaluate purity and yield using preparative polyacrylamide gel electrophoresis. Scale this process to the kilogram level and calculate the cost of step.

The above scaling exercises are, of course, calculations only. The students are expected in each case to write a formal report as if it were intended for a client.

ADDITIONAL ASSIGNMENTS

Homework is assigned just as it is in any engineering course. A few open-ended problems are assigned in which students must find or estimate extensive and/or intensive properties; spreadsheet calculations and some derivations are included. In some cases a term paper must be written in the form of a critique of a single published paper or as an in-depth summary of a single subject (chosen by the student) based on the reading of recent literature. Both descriptive and analytical questions are included on examinations.

ACKNOWLEDGMENTS

Drs. Geoffrey Slaff, Dale Gyure, Robert J. Todd, Brian Batt, and Mr. Michael Sportiello in Colorado contributed to the development of practicums. Participating companies included Amgen, Synergen, and Zeagen. Drs. John J. Pellegrino of the National Institute of Standards and Technology, Robert H. Davis of the University of Colorado, and Dr. Charles Glatz of the Iowa State University participated in the development of the survey, lecture materials and homework problems.

REFERENCES

1. Asenjo, J.A., and J. Hong, "Separation, Recovery, and Purification in Biotechnology," *ACS Symp. Series 314,* American Chemical Society, Washington, DC (1986)
2. Scopes, R.K., *Protein Purification: Principles and Practice,* Springer Verlag, Berline (1982)
3. Giddings, J.C., *Unified Separation Science,* John Wiley & Sons, New York, NY (1991)
4. Bruno, T.J., *Chromatographic and Electrophoretic Methods,* Prentice-Hall, Englewood Cliffs, NJ (1991)
5. Belter, P.A., E.L. Cussler, and W.-S. Hu, *Bioseparations: Downstream Processing for Biotechnology,* Wiley Interscience, New York, NY (1988)
6. Bailey, J.E., and D.F. Ollis, *Biochemical Engineering Fundamentals,* 2nd ed., McGraw-Hill, New York, NY (1986)
7. Schuler, M.L., and F. Kargi, *Bioprocess Engineering,* Prentice-Hall, Englewood Cliffs, NJ (1992)
8. Yamamoto, S., M. Nomura, and Y. Sano, *J. Chromatography,* **409**, 101 (1987) ❐

Computing

Continued from page 285.

of steps set out in the lab handout. The applications should be interactive and open-ended, requiring the students to use engineering analysis and methodology during the lab, *e.g.,* they should figure out which one of several possible control schemes for level and/or temperature in tank 1 of Figure 1 is "best" for their application.

- ▶ Integrating the RTSB computing with other course material, *e.g.,* using real-time Simulink software in the lab if the students have used Matlab/Simulink in other courses.
- ▶ The applications, *i.e.,* the process instrumentation and computer system, should be realistic enough that the relevance to course material and industrial requirements is obvious. However, the application should be more flexible and easier to program than most commercial systems designed for industrial operations.

REFERENCES

1. Matlab Real Time Workshop Toolbox. The Mathworks, Inc. http://www.mathworks.com (1995)
2. Labview for Windows. National Instrument. http://www.natinst.com (1993)
3. G2—Expert Systems Development Software, Gensym Corporation. http://www.gensym.com
4. Seborg, D.E., T.F. Edgar, and D.A. Mellicamp, *Process Dynamics and Control* (1989)
5. Ogunnaike, B.A., and W.H. Ray, *Process Dynamics, Modeling, and Control,* Oxford University Press, Inc., New York, NY (1994)
6. Fisher, D.G., "Process Control: An Overview and Personal Perspective," *CJChE,* **69**, Feb. (1991)
7. Badmus, O.O., D.G. Fisher, and S.L. Shah, *Computer Program for Generating Multivariable State Space Models from Process I/O Data* (1996)
8. House, R., "Choosing the Right Software for Data Acquisition, *IEEE Spectrum,* May (1995)
9. Bialkowski, W.L., and A.D. Weldon, "The Digital Future of Process Control: Its Possibilities, Limitations, and Ramifications," *TAPPI J.,* **77**(10), Oct (1994)

APPENDIX

The specifications of the computer-process interface used to generate the results presented in this paper are:

Hardware

- CIO-DAS16/F (primary data acquisition board with 16 single-ended (8 differential) 12-bit A/D, 2 D/A, 32 DIO and 3, 16-bit counters
- CIO-EXP16: Expansion board with 16 A.I. multiplexing and thermocouple signal conditioning
- CIO-DAC02: (add-on board with 2 12-bit D/A for voltage or current output) (Source: Computer Boards, Inc.)

Software

- Labview drivers (Source: National Instruments)
- Matlab/Simulink drivers (Source: Mathworks)
- C++ drivers (Source: Mathworks, Inc., and user written)

Host Computer

- PC (486 with 16Mb memory, 1.2Gb HDD)
- Running Microsoft Windows 3.1 ❐