An Interactive Control Systems Simulator

Kent S. Diehl, Bruce H. Krogh, and Mark L. Nagurka

ABSTRACT: An Interactive Control systems Simulator (ICS) for control engineering education is presented. ICS contains an interactive block diagram editor with which arbitrary linear control systems can be constructed for a library of predefined nonlinear plants. The time response of the constructed system is simulated with animated graphics, and data can be collected for display with the ICS plotting facility. This paper describes the structure and use of ICS and presents two examples.

Introduction

This paper describes an *Interactive Control systems Simulator* (ICS) developed at Carnegie-Mellon University (CMU) for use in control engineering education. The ICS environment includes: preprogrammed dynamic systems, or *plants*, for which the student can construct controllers; an interactive block diagram editor for specifying linear feedback control systems; a simulator that displays the time evolution of the plant dynamic response (via animation) while performing the time integration of the block diagram model; a comprehensive plotting package; a disk I/O system for saving block diagrams; and an extensive on-line help facility. ICS is menu-driven, employing a data tablet as the primary input medium. Students use a locator stylus to construct block diagrams, move between menu levels, and invoke commands.

ICS is implemented on the Hewlett-Packard 9836C (HP Pascal Operating System) personal scientific workstation [1]. Each workstation includes a high-resolution color display (512 by 390 pixels), a thermal printer, a graphics tablet, two 5.25-in. floppy disk drives, 1.25 Megabytes of RAM, and a 15-Megabyte Winchester disk drive. Over 40 of these workstations are available for student use in three laboratories at CMU.

ICS provides a reliable, easy-to-use *laboratory* of animated control experiments to complement control engineering courses taught at CMU. With ICS, students implement and test feedback control designs for a number of nonlinear plants. The block diagram editor provides a natural, efficient means for specifying controller designs, while the menu-driven interface makes the program easy to learn [2]. During simulations, the animation of the system response gives the student physical insight into the performance of the control design, thus adding to the laboratory flavor of this package.

ICS has two principal features that distinguish it from many other control systems simulation programs: an interactive block diagram interface and a library of predefined animated plants, which enhances the *educational* application of the program. Other block diagram interfaces that have been developed are described in [3]–[6]. A program using animation as an educational tool for control engineering is described in [3].

Students use ICS in conjunction with control systems analysis and design software, such as CACHE [7], a program also developed at CMU. CACHE provides extensive computational and graphics functions incorporating classical and modern control engineering methodologies. In a typical assignment, students use CACHE to perform the calculations necessary to design a controller for one of the ICS plants and then use ICS to implement and test their design. Although CACHE and ICS use similar data structures, students manually transfer data between the programs to maintain a clear distinction between the stages of linear controller design (with CACHE) and feedback control implementation (with ICS).

ICS, written in PASCAL, currently requires 280 Kbytes of memory for the binary code. Linked lists are used throughout the data base to circumvent fixed dimensionality constraints and to provide dynamic allocation for data structures created during a user session. For example, block diagrams are stored as dynamic structures: when a block is added to the block diagram, memory is allocated for this structure; when a block is deleted from a block diagram, memory is freed for future use. The modular program structure facilitates the addition of new types of blocks and preprogrammed plants.

The remainder of this paper describes the use and features of ICS. The next section provides an overview of the program, while the following section details the available blocks and explains how block diagrams are constructed. Each of the four predefined plants that are currently implemented are then described. The concluding section presents examples of controllers implemented in ICS for two of the predefined plants. The plant dynamics and blocks used in the examples are defined in the appendixes.

Structure of ICS (Main Menu Entries)

The ICS menu hierarchy, shown in Fig. 1, is structured to be self-explanatory to the novice user and efficient for the user familiar with the program. The structure and use of ICS are understood from the functions of the items on the main menu. In this section, we describe some of the menu items in the order in which they might be invoked in a typical controller-design session.

Selection of *Plant Lib* from the main menu moves the user to the plant library menu where the plant to be controlled is chosen. Block diagrams may be saved and retrieved from disk files using the *Filer*. To construct and modify block diagrams, the *Editor* menu item is selected. From the editor menu, several pop-up menus (not shown in Fig. 1) can be invoked for defining, browsing, and modifying attributes of blocks. The *Pre-Plot* menu level (also not shown in Fig. 1) is used to specify signals to be sampled during a simulation, and the *Post-Plot* menu provides an interactive interface for plotting these signals. On-line help is available for each menu level; to obtain help, the user selects *Help* followed by the menu item in question.

The *Modify* command allows the keyboard entry of the integration step, the final time of the simulation, the period at which signals are sampled for plotting, and the period at which the animation is updated. The *Run* menu item invokes the ICS simulator that performs a fourth-order Runge-Kutta time integration of the equations represented by the block diagram. If the block diagram includes one of the predefined plants, a dynamic animation is displayed during the simulation, including a numerical display of the elapsed time and values of the plant state and input variables.

ICS was designed to be "user friendly" by incorporating error-recovery routines to han-

This work was supported in part by the Center for Design of Educational Computing, Carnegie-Mellon University, and presented at IEEE Symposium on Computer-Aided Control System Design, Santa Barbara, California, March 13-15, 1985. Kent S. Diehl was with the Department of Electrical and Computer Engineering, Carnegie-Mellon University, and is now with Systems Control Technology, Palo Alto, CA 94303. Bruce H. Krogh is with the Department of Electrical and Computer Engineering and Mark L. Nagurka is with the Department of Mechanical Engineering, Carnegie-Mellon University, Pittsburgh, PA 15213.

0272-1708/86/0400-0020 $01.00 © 1986 IEEE

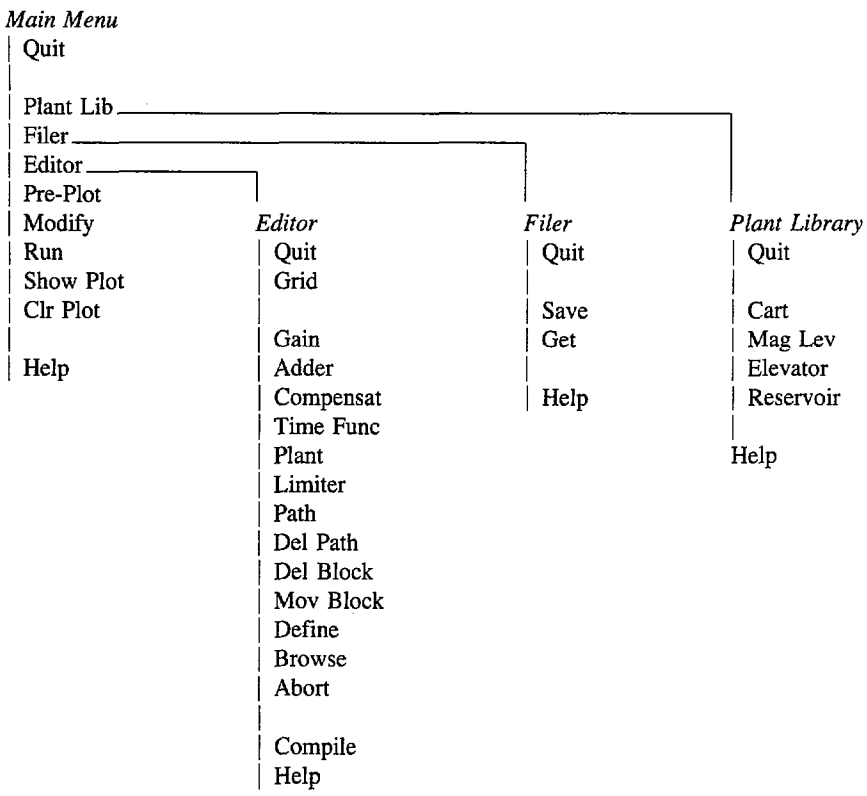

Fig. 1. ICS menu hierarchy—menu selections are made using the data tablet stylus.

dle typing errors, meaningless sequences of menu selections, and numerical saturation. For example, if the student specifies an unstable system that would lead to numerical overflow, the error-recovery mechanism terminates the simulation automatically without aborting the program.

Block Diagrams

With ICS, the student can implement linear feedback controllers using the block diagram editor. Block diagrams are constructed by selecting blocks from the editor menu and placing them in a design viewport. Blocks are color-coded to correspond to entries in the menu frames. Blocks are placed on the screen with the locator stylus and connected together by drawing paths between them. Blocks may be deleted, moved, connected, disconnected, defined, browsed (inspected), or modified at any time. This complete flexibility in the construction of block diagrams makes ICS easy to learn and use.

The six types of blocks are: *plant* blocks, representing one of the predefined plants (discussed below and in the following section); *compensator* blocks, defined by arbitrary linear dynamic equations in state-space form; *time function* blocks, which act as signal generators; matrix *gain* blocks; *limiter* blocks, for clipping scalar signals; and *adder* blocks, for adding signals together. The *plant, compensator, gain,* and *adder* blocks are, in general, multi-input/multi-output. A signal path joining two blocks is a vector with dimensions determined by the output and input dimensions of the connected blocks. The characteristics of the blocks are summarized in the Table.

A *plant* block represents the predefined plant that is to be controlled. The nonlinear differential equations representing each plant's dynamics are preprogrammed in ICS. The linearized dynamics, with respect to a user-specified operating point, are provided by ICS as the **A** and **B** matrices for the plant, representing the linearized process matrix and input matrix, respectively. These matrices are provided for design purposes only and are not used by the simulator. The plant output matrix **C** is an identity matrix; that is, the output of a plant block is the plant state vector.

After building a block diagram, the user invokes the ICS *compiler*. The *compiler* checks blocks for undefined fields and verifies that connected blocks have compatible input and output dimensions. If a block is found that has not been completely specified, an error message is written to the user and compilation stops. If there is an inconsistency in the input/output dimensions of two connected blocks, then the path between these two blocks is redrawn in red to indicate the error. If the compiler finds no errors in the block diagram, then a message is written to the user indicating that the block diagram is syntactically correct. At this point, the user may proceed to simulate the dynamic system represented by the block diagram.

The Predefined Plants

A key feature of ICS, making it particularly suitable for educational purposes, is the library of predefined plants. Each plant in-

Table
Summary of Block Elements

Block	Fields	Number of Inputs	Input Dimension	Number of Outputs	Output Dimension
Plant	matrices **A, B, C**	0,1	number of columns in matrix **B**	any	number of rows in matrix **C**
Compensator	matrices **A, B, C** vector **X_ICs**	0,1	number of columns in matrix **B**	any	number of rows in matrix **C**
Gain	matrices **K**	1	number of columns in matrix **K**	any	number of rows in matrix **K**
Gain	**K**	1	number of columns in matrix **K**	any	number of rows in matrix **K**
Limiter	Limits	1	1	any	1
Time Function	$f(t)$	0	—	any	1
Adder	none	any	any	same as inputs	same as inputs

cludes preprogrammed nonlinear dynamics for simulations, linearized dynamics about a user-specified operating point for controller design, and graphics routines for animating the plant time response during simulations. Each plant has a set of parameters with default values that may be modified by the user. These parameters include physical parameters of the plant model, the nominal state for linearization, and the initial state for simulation. A brief description of each plant and its animation is given in this section.

Currently, the plant library contains four predefined systems: a movable *cart* balancing an inverted pendulum; a *levitation* system consisting of a ferromagnetic sphere acting under the influence of a voltage-controlled electromagnet; an *elevator* positioned by an armature-controlled DC motor; and a two-tank *reservoir* system with two controlled inputs and two outputs having nonlinear flow-head characteristics.

The *cart*, shown in Fig. 2, consists of an inverted pendulum mounted on a movable cart. The inverted pendulum is modeled as a massless rod with a point mass located at one end. The other end pivots freely on the cart. This plant has four state variables: position, velocity, pendulum angle, and pendulum angular velocity. The position of the cart is controlled by applying an external horizontal force to the base of the cart. The user specifies the mass of the cart, the point mass of the pendulum, and the length of the pendulum. A typical control objective for this plant is to track a reference position signal while balancing the pendulum. This plant is used in the first example of the following section and the nonlinear dynamic model is given in Appendix I.

The *levitation* system consists of a ferromagnetic sphere to be levitated by an electromagnet. The animated display for this system is shown in Fig. 3. The inductor is modeled as a nonlinear element in a circuit containing a resistor, a constant DC voltage, and a controllable voltage source [8]. The user specifies the value of the resistor, constants associated with the inductor, the mass of the sphere, and its initial and nominal positions. From these data, the program determines the value of the constant DC voltage and the nominal current. The state variables for this plant are the sphere's position and velocity, and the inductor current. The typical control objective for this plant is to levitate the ball at a prescribed distance from the magnet. This experiment proved to be very useful for demonstrating concepts of state-variable feedback control based on linearization about nonzero operating points.

The *elevator* system contains an armature-controlled DC motor whose shaft turns a pulley that supports an elevator, as shown in Fig. 4. The motor speed is controlled by varying the armature voltage. The motor model includes the armature inductance and resistance, shaft inertia, bearing friction, and motor constants such as the back emf constant [9]. The elevator mass may be specified, as well as the pulley mass and radius. The state variables are the armature current and the elevator position and velocity. This plant is useful for demonstrating regulator design principles to control the elevator's position, with disturbance rejection, as in the case when the elevator mass is not known *a priori*.

The fourth plant is a two-tank *reservoir* system with one linear and one nonlinear output-flow characteristic. Each tank has a controllable input flow rate. The tanks are situated on different levels so that the output of the upper tank flows into the lower tank, as shown in Fig. 5. The bottom areas of both tanks and the two output flow resistances may be specified. The two state variables for

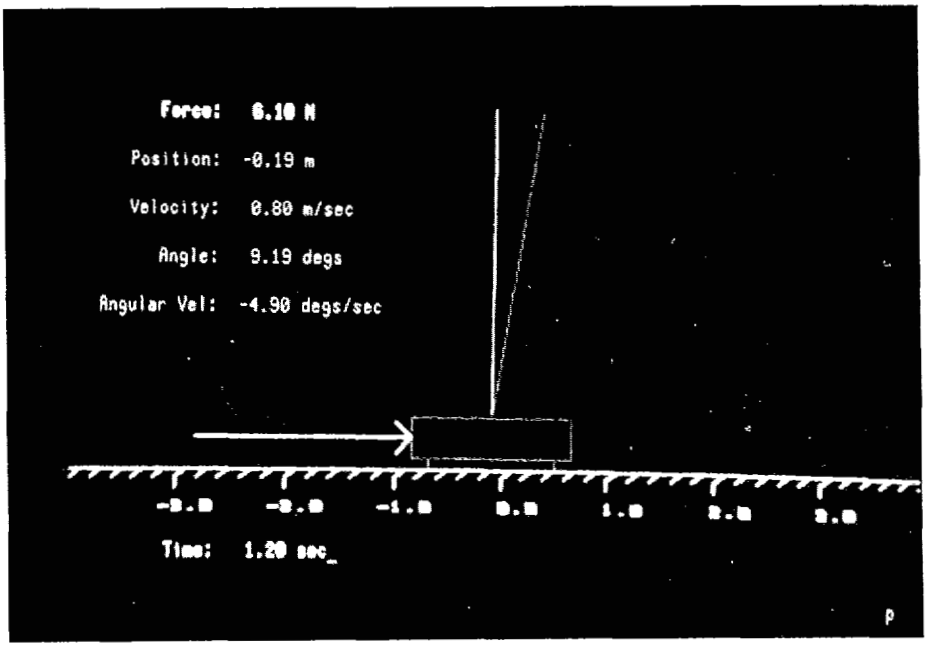

Fig. 2. Animation for the Cart-Pendulum Plant.

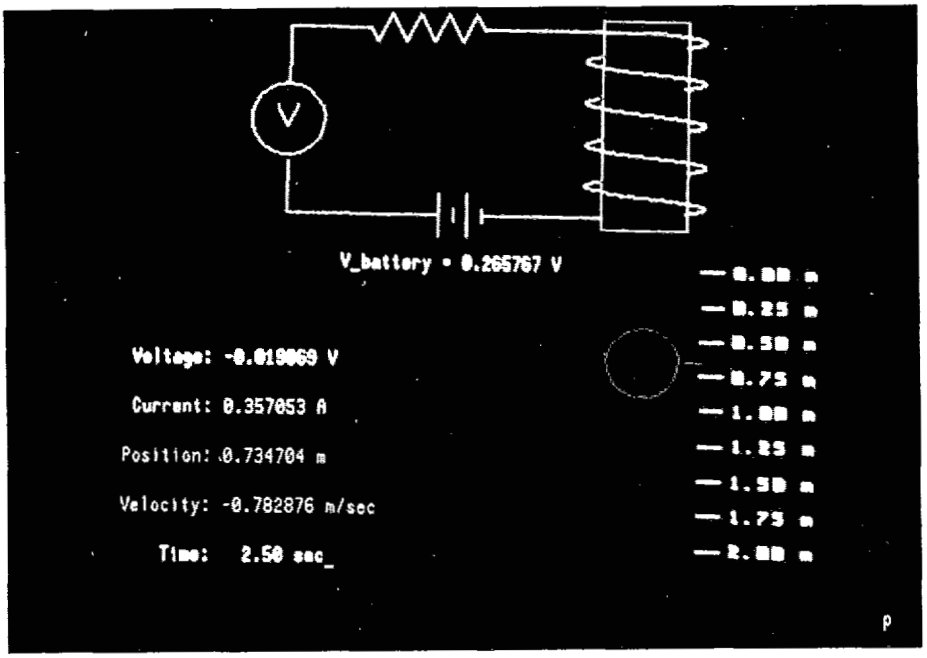

Fig. 3. Animation for the Magnetic Levitation Plant.

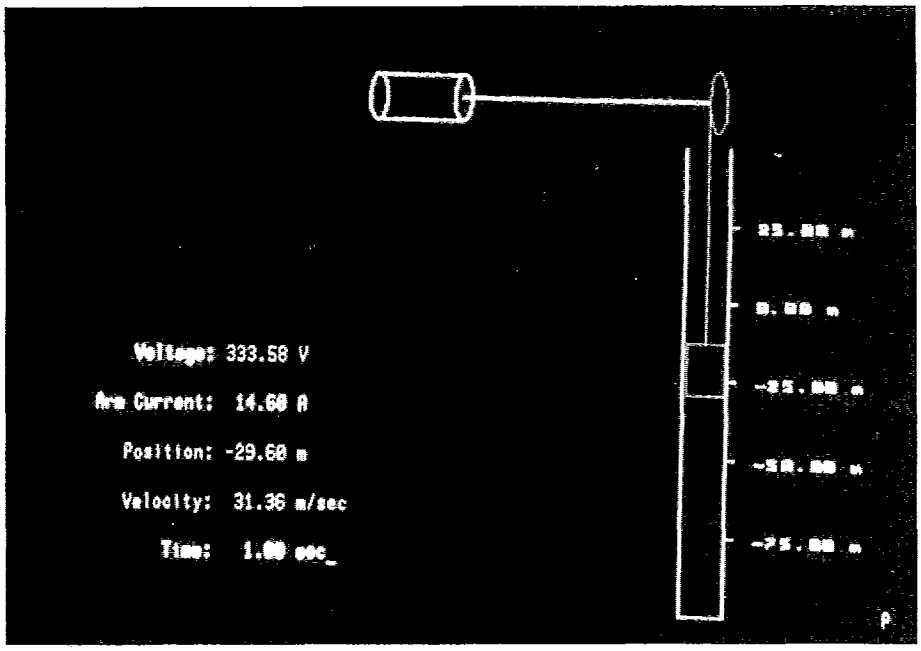

Fig. 4. Animation for Elevator Plant.

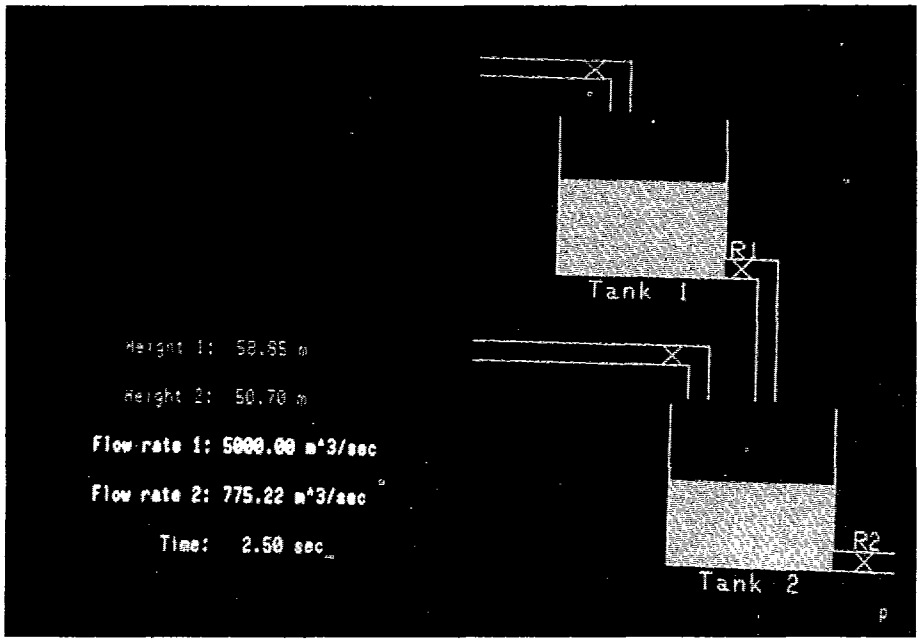

Fig. 5. Animation for Two-Tank Plant.

this plant are the liquid heights in the tanks. This multi-input/multi-output plant is used to illustrate a number of concepts including disturbance rejection, regulation, optimal control design, and problems with bounded state and control variables (e.g., the water levels and input flows cannot be negative).

Examples

As an example, consider the problem of moving the cart to a reference position while balancing the inverted pendulum. We assume that only the position and velocity of the cart are measurable. The student is required to implement a controller using state-variable feedback and integral control with an observer to estimate the missing state variables. For this example, we outline the steps a student would follow to develop and test a control design for this system. The dynamic equations and definitions of the blocks are given in Appendix I.

Using ICS, the student first specifies the values of the nominal state and physical parameters for the cart plant. To determine gains for state variable plus integral feedback, the student obtains from ICS the linearized model of the cart by browsing the **A** and **B** matrices of the plant block. Specifying pole locations for the closed-loop system as $\{-2, -3, -4, -3 \pm j\}$, the student uses CACHE to solve for the integral and state-variable feedback gain matrices [10]. A reduced-order observer is also designed to estimate the third and fourth state variables [11]. In this example, the pole locations for the observer were selected to be $\{-20, -20\}$.

Following this numerical design work, the student uses ICS to implement and evaluate the controller. The block diagram, shown in Fig. 6, is constructed using the ICS block diagram editor. After the block diagram has been checked by the compiler for syntax errors, signals to be plotted are specified by selecting the appropriate components of input and output vectors for blocks in the diagram. For this example, we have chosen to look at the cart position. The student runs the simulation and observes the effects of the controller via the animation. When the simulation is completed, plots of the previously specified signals may be displayed. Figure 7 shows the time history of the cart position, which asymptotically approaches the reference input value. A hard copy of these plots and the block diagram may be made on the thermal printer for future reference, and the block diagram data can be saved on a floppy disk.

As a second example, a full-order observer was designed to estimate the state of the *levitation* plant, based on measurements of the ball's position. The equations and parameters for this system are given in Appendix II. The block diagram is shown in Fig. 8. The observer poles are placed at $\{-5, -6, -7\}$. State-variable feedback control is used to stabilize the ball's position with the poles for the closed-loop linearized system placed at $\{-2, -3 \pm j\}$. The ball's nominal position is 0.2 m and initial position is 0.25 m. Figure 9 shows that the ball reaches its nominal position within 4.0 sec.

Concluding Remarks

Students at CMU have responded favorably to the use of ICS as a laboratory of computer-animated experiments for graduate and undergraduate courses in dynamic systems and controls. The concepts of linearization, stability, state-variable feedback, asymptotic observers, and disturbance rejection are developed in a sequence of design problems. ICS is used in conjunction with CACHE, a control system analysis and design program also developed at CMU [7], to

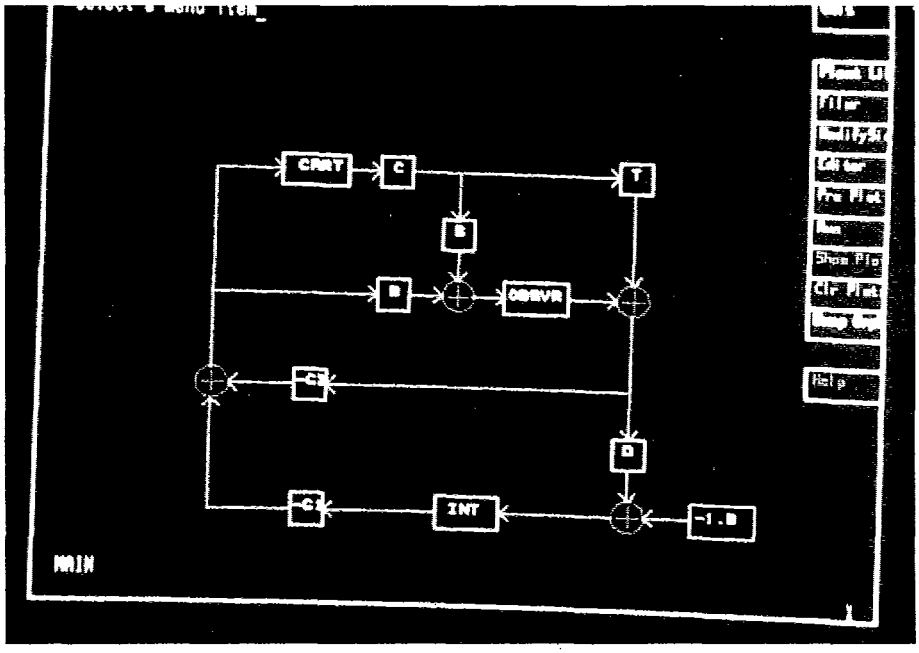

Fig. 6. Block diagram solution for cart example.

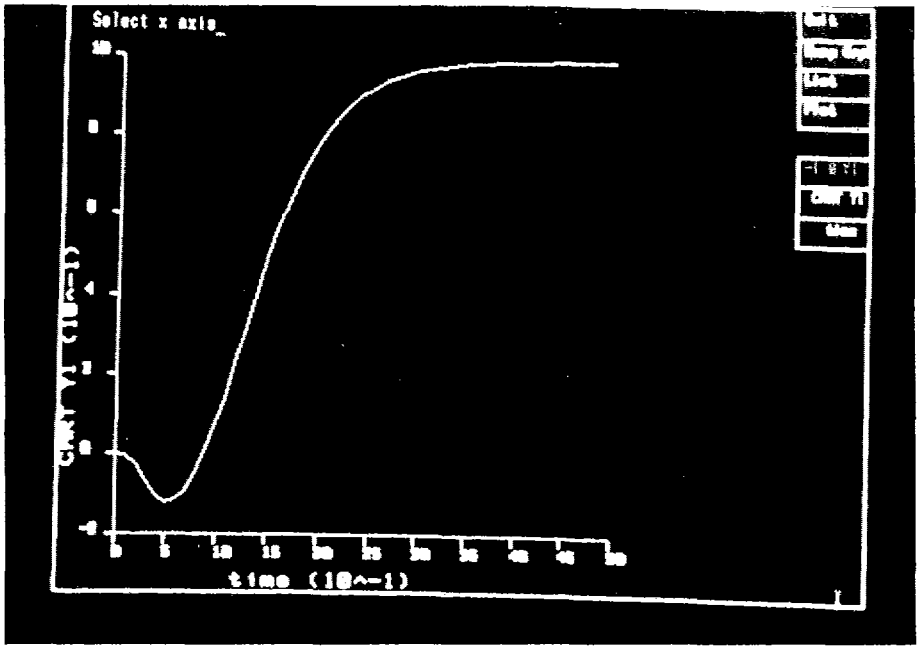

Fig. 7. Plot of cart position vs. time for cart example.

provide insight into the power and limitations of linear control design techniques applied to the nonlinear plants. The block diagram editor and animated graphics help the students develop their intuition for the physical structures of the systems and controller designs.

Readers interested in obtaining the ICS user documentation and floppy disks containing the ICS source code should contact Professor Bruce H. Krogh, Department of Electrical and Computer Engineering, Carnegie-Mellon University, Pittsburgh, PA 15213 [phone: (412) 578-2472]. A videotape demonstrating the features of ICS is also available.

Appendix I: Cart-Pendulum Plant

The nonlinear dynamics of the cart and inverted pendulum plant are given by:

$$\ddot{x} = \beta[\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta] + \beta F/(lm_b)$$

$$\ddot{\theta} = [g \sin\beta - \ddot{x}\cos\theta]/l$$

where m_b is the mass at the end of the inverted pendulum, l the length of the inverted pendulum, $\beta = m_b l/(m_b + m_c)$ the center of mass of the system, where m_c is mass of the cart, x the cart position (positive to the right), θ the angle on the inverted pendulum from vertical (positive clockwise), and F the force (control) applied to the cart.

The following is a summary of the blocks in Fig. 6 for this example:

CART EXAMPLE

Plant

'cart' $A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -9.81 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 9.81 & 0 \end{bmatrix}$

$B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -0.5 \end{bmatrix}$

$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Gains

'c' $K = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

's' $K = \begin{bmatrix} 0 & 121.324 \\ 0 & 1630.99 \end{bmatrix}$

'B' $K = \begin{bmatrix} 4.0774 \\ 4.1274 \end{bmatrix}$

'-G2' K =

[82.365 65.021 364.35 160.04]

'-G1' K = [48.930]

'D' K = [1 0 0 0]

'T' $K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -4.0775 \\ 0 & -41.775 \end{bmatrix}$

Compensators

'OBSER' $A = \begin{bmatrix} -40 & 1 \\ -400 & 0 \end{bmatrix}$

$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$

'INT' A = [0], B = [1], C = [1]

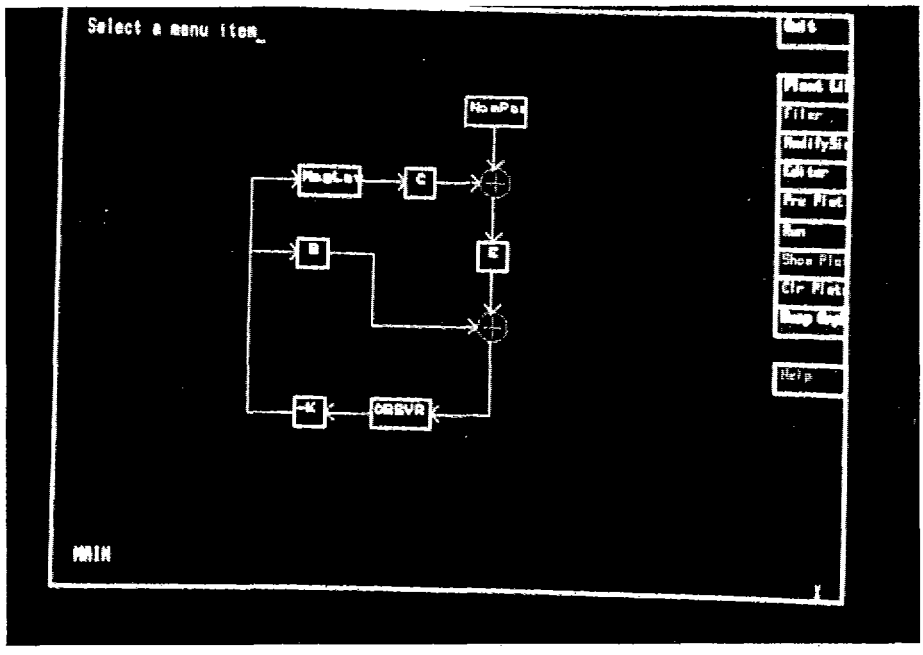

Fig. 8. Block diagram solution for levitation example.

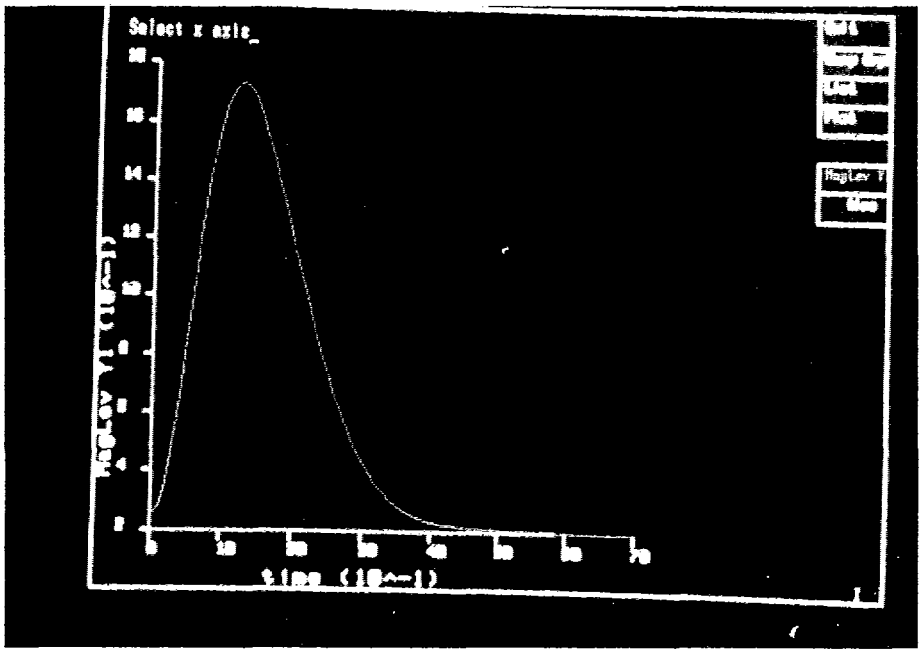

Fig. 9. Plot of ball position vs. time for levitation example.

Time Function

'−1.0' $f(t) = -1$

Appendix II: Levitation Plant

The nonlinear equations for the levitation system are given by

$$m\ddot{x} + (L_0/2)(a + x)^{-2}\dot{q}^2 - mg = 0$$

$$[L_1 + L_0/(a + x)]\ddot{q} - L_0(a + x)^{-2}\dot{x}\dot{q} + R\dot{q} = V_s + V_b$$

where R is the resistance, m the mass of the ferromagnetic sphere, $L(x) = L_1 - L_0/(a + x)$ the coil inductance, L_0 and L_1 the positive constants, V_s the controllable voltage source, V_b the nominal constant voltage (in series with V_s), q the current, and x the position of the ball from the bottom of the inductor (positive distance is measured downward).

The following is a summary of the blocks as defined in Fig. 8 for this example:

LEVITATION EXAMPLE

Plant

'Lev' $A = \begin{bmatrix} 0 & 1 & 0 \\ 65.4 & 0 & -147.65 \\ 0 & 0.4300 & 0.0582 \end{bmatrix}$

$B = \begin{bmatrix} 0 \\ 0 \\ 0.2912 \end{bmatrix}$ $C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Gains

'c' $K = [1 \quad 0 \quad 0]$

'B' $K = \begin{bmatrix} 0 \\ 0 \\ 0.0291 \end{bmatrix}$

'E' $K = \begin{bmatrix} 17.9417 \\ 107.8590 \\ 6.3101 \end{bmatrix}$

'−k' $K = [126.310 \quad 5.5585 \quad -272.67]$

Compensator

'OBSER'

$$A = \begin{bmatrix} -17.942 & 1 & 0 \\ -42.459 & 0 & -147.65 \\ -6.3102 & 0.43004 & 0.0582 \end{bmatrix}$$

$$B = C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

References

[1] Hewlett-Packard Company, *Pascal 2.1 User's Manual for the Hewlett-Packard Series 200 Computers*, 1983.

[2] K. S. Diehl, Carnegie-Mellon University, *ICS Users Guide Version 1.0*, 1984.

[3] D. K. Frederick, R. P. Kraft, and T. Sadeghi, "Computer-Aided Control Systems Analysis and Design Using Interactive Computer Graphics," *IEEE Contr. Syst. Mag.*, vol. 2, no. 4, pp. 19-23, Dec. 1982.

[4] M. Buchner and B. Sandridge, "A Graphics Interface for Control System Design," *2nd IEEE CSS Symposium on CADSD, Santa Barbara*, Mar. 1985.

[5] M. Jamshidi, Editor, "Special Issue: Computer-Aided Design of Control Systems," *IEEE Contr. Syst. Mag.*, vol. 2, no. 4, Dec. 1982.

[6] C. J. Herget and A. J. Laub, Editors, "Special Issue on Computer-Aided Control System Design," *IEEE Proceedings*, vol. 72, no. 12, Dec. 1984.

[7] J. M. Mason, C. P. Neuman, and B. H. Krogh, "CACHE: An Interactive Control Systems Analysis and Design Package," *IEEE J. on Educat.*, vol. 28, no. 3, pp. 143-149, Aug. 1985.

[8] H. H. Woodson and J. R. Melcher, *Electromechanical Dynamics*, New York: Wiley, 1968.
[9] B. J. Kuo, *Automatic Control*, Englewood Cliffs, NJ, Prentice-Hall, 1982.
[10] H. W. Smith and E. J. Davis, "Design of Industrial Regulators, Integral Feedback and Feedforward Control," *IEEE Proceedings*, vol. 119, no. 8, pp. 1210-1216, Aug. 1972.
[11] D. L. Luenberger, "An Introduction to Observers," *IEEE Trans. Automat. Contr.*, vol. AC-16, no. 6, pp. 596-602, Dec. 1971.

Bruce H. Krogh received the B.S. degree in mathematics and physics from Wheaton College, Wheaton, Illinois, in 1975 and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois, Urbana, in 1978 and 1982, respectively. From 1980 to 1981, he was a Systems Engineer with Westinghouse Electric Corporation involved in the development of algorithms for energy control center software. At present, he is an Assistant Professor in the Department of Electrical and Computer Engineering, Carnegie-Mellon University, Pittsburgh, Pennsylvania. His research interests include modeling, simulation and computational methods for real-time supervisory feedback control of dynamic systems with applications to robotics, manufacturing automation, and large-scale networks.

Mark L. Nagurka was born on December 14, 1956, in Bristol, Pennsylvania. He received his B.S. and M.S. degrees in mechanical engineering and applied mechanics from the University of Pennsylvania in 1978 and 1979, respectively, and his Ph.D. degree in mechanical engineering from the Massachusetts Institute of Technology in 1983. Following graduation from MIT, he joined the faculty in the Department of Mechanical Engineering at Carnegie-Mellon University and is researching the dynamics and control of mechanical systems. In February 1985, Professor Nagurka received the Ralph R. Teetor Educational Award of the Society of Automotive Engineers in recognition of significant contributions to teaching, research, and student development.

Kent S. Diehl received a B.S. degree in physics and mathematics in 1983 and an M.S.E.E. degree in 1985, both from Carnegie-Mellon University. He is continuing his work in CACSD packages with Systems Control Technology in Palo Alto, California.

Conf. Calendar continued from p. 19

I. Troch, % Technical University Vienna, Karlsplatz 13, A-1040 Wien, Austria.

IEEE Control Systems Society Third Symposium on Computer-Aided Control System Design
Sept. 24–26, 1986
Quality Inn, Pentagon City
Arlington, Virginia

Three copies of 700-word abstract due by March 30, 1986.

General Chairman:
David W. Porter
Business and Tech. Systems, Inc.
Aerospace Building, Suite 440
10210 Greenbelt Road
Seabrook, MD 20706
Phone: (301) 794-8800

IEEE Industrial Electronics Society Conference, Sept. 29–Oct. 3, 1986, Hyatt Regency, Milwaukee, Wisconsin. Technical Program Chairman: Dr. Guy O. Beale, Vanderbilt University, Box 1698, Station B, Nashville, TN 37235, phone: (615) 322-2212.

Allerton Conference on Communication, Control, and Computing, Oct. 1–3, 1986, Allerton House, near Monticello, Illinois. Send 100-word abstract before July 28, 1986, to Allerton Conference, % Prof. Michael C. Loui, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1101 W. Springfield Ave., Urbana, IL 61801, phone: (217) 333-2595.

1986 International Symposium on Information Theory, Oct. 5–8, 1986, University of Michigan, Ann Arbor, Michigan. Program Chairman: Prof. Stuart C. Schwartz, Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA.

1986 International Conference on Systems, Man, and Cybernetics, Oct. 14–17, 1986, Pierremont Plaza Hotel, Atlanta, Georgia. Send three copies of one-page abstract by April 15, 1986, to Program Chairperson: Nancy M. Morris, Search Technology, Inc., 25-B Technology Park/Atlanta, Norcross, GA 30092, USA, phone: (404) 441-1457.

6th International IFAC/IFIP/IMEKO Conference on Instrumentation and Automation in the Paper, Rubber, Plastics, and Polymerization Industries, Oct. 27–29, 1986, Akron, Ohio. Contact: Prof. A. Kaya, The University of Akron, Mechanical Engineering Dept., Akron, OH 44325, USA, phone: (216) 375-7735.

International Symposium on Robotics: Modeling, Control, and Education, Nov. 12–14, 1986, University of New Mexico, Albuquerque, New Mexico. Contact: Mo Jamshidi, CAD Lab Systems/Robotics, EECE Dept., University of New Mexico, Albuquerque, NM, 87131, USA, phone: (505) 277-0300.

IFAC Symposium on Components, Instruments, and Techniques for Low Cost Automation and Applications, Nov. 27–29, 1986. Contact: LCA'86, Departamento de I. S. y Automática, Universidad Politécnica de Valencia, P.O. Box 22012, 46071–Valencia, Spain, phone: (34) (6) 360 40 41, Telex: 62808 UPV.

IFAC/IFIP/IMACS International Symposium on Theory of Robots, Dec. 3–5, 1986, Vienna, Austria. Contact: Dr. P. Kopacek, ÖPWZ, P.O. Box 131, A-1014 Vienna, Austria.

IEEE Control Systems Society 1986 Conference on Decision and Control
Dec. 10–12, 1986
Athens, Greece
General Chairman:
Prof. Anthony Ephremides
Electrical Eng. Dept.
University of Maryland
College Park, MD 20742, USA

IFAC Symposium on Automation and Instrumentation for Power Plants, Dec. 15–17, 1986, Bangalore, India. Contact: Dr. M. Ramamoorty, % IFAC 86, The Institution of Engineers (India), Karnataka State Centre, Ambedkar Veedhi, Bangalore–560 001, India.