European Journal of Engineering Education, 9 (1984) 135—151
Elsevier Science Publishers B.V., Amsterdam — Printed in The Netherlands

The Use of Computers in the Education of Control Engineers at ETH Zürich

M.A. Mansour, W. Schaufelberger, F.E. Cellier, G. Maier and M. Rimvall

1. Introduction

The education of control engineers is in a transient state. Computers are introduced for various purposes into the educational process. The goals of compulsory education in computers and control in the electrical engineering curriculum at ETH Zürich are described briefly in the following. Different ways of using computers to a considerable extent in control education are surveyed. The operating environment, consisting of hardware, software and laboratory processes, is also described because of its importance in the educational process.

2. The Curriculum in Electrical Engineering at ETH Zürich

The programme leading to the Diploma Degree in Electrical Engineering is a four-year programme plus a Diploma semester. There is also a one-year postgraduate programme in control and communication. The eight semesters of the four-year degree programme are organized as follows:

— semesters 1—3: propaedeutical education;
— semesters 4—6: general electrical-engineering education;
— semesters 7—8: projects and specialization.

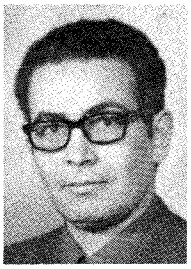
The curriculum is fairly rigid for the first three years. The total time spent in classroom and laboratory is ~3600 hours, of which 500 hours are devoted to nontechnical education. Projects are vital for the last year (50% of the total time). In addition, a diploma thesis has to be written to obtain the Diploma Degree.

2.1 Compulsory Computer Education in the Electrical Engineering Department

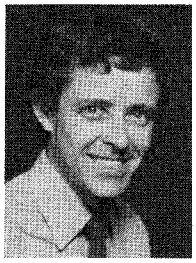
Every student in Electrical Engineering Department has to take the following three courses in computer science:

0304-3797/84/$03.00

François E. Cellier is Lecturer in Simulation Techniques at the Swiss Federal Institute of Technology (ETH) in Zürich, Switzerland. He received his Diploma in Electrical Engineering from the same university in 1972, an M.S. in Automatic Control in 1973, and a Ph.D. in Technical Sciences in 1979. His main scientific interests concern modelling and simulation methodology and the design of advanced software systems for simulation and computer-assisted modelling. His research activities range from the numerical aspects of simulation software (software/hardware interface) to the information-processing point of view (user/software interface). He has published more than 25 papers in this area.



Mohamed Mansour received his B.Sc. and M.Sc. in Electrical Engineering from the University of Alexandria, Egypt, in 1951 and 1953, respectively, and a Dr.Sc.Techn. in Electrical Engineering from ETH Zürich, Switzerland, in 1965, when he was awarded the silver metal of ETH. He was Assistant Professor in Electrical Engineering at Queen's University, Canada, from 1967 to 1968. He has been Professor and Head of the Department of Automatic Control at ETH Zürich since 1968, Dean of Electrical Engineering (1976–1978), and Director of the Institute of Automatic Control and Industrial Electronics, ETH Zürich, during 1976–1978 and 1980–1982 and since 1984. He is also President of the Swiss Federation of Automatic Control, and member of the Council and Treasurer of IFAC. His fields of interest are control systems, especially stability theory and digital control, the stability of power systems, and digital filters.



Georg Maier was born in 1954 in Schaffhausen (Switzerland). In 1979 he received his Diploma Degree in Electrical Engineering from ETH Zürich, and since then he has been working as an assistant in the Department of Automatic Control and Industrial Electronics of ETH Zürich. His main topics are real-time programming and real-time operating systems using the high-level language Modula-2. From 1979 to 1982 he was an active member of the Technical Committee on Real Time Operating Systems of the European Workshop on Industrial Computer Systems (EWICS-TC8). He is currently working on a Ph.D. thesis on methods for exception handling and synchronization in real-time programming.

C. Magnus Rimvall was born in Laholm, Sweden, in 1957. He received his Diploma Degree in Physics Engineering from Lunds Tekniska Högskola in 1982. He is currently a Ph.D. candidate at the Institute for Automatic Control and Industrial Electronics at ETH. His research interests include computer-aided control systems design and simulation language design. Mr. Rimvall is a member of IMACS and SCS.

Walter Schaufelberger is Professor of Control Engineering in the Electrical Engineering Department of the Swiss Federal Institute of Technology (ETH). He received his Diploma Degree in 1964 and his Ph.D. in 1969, both from ETH. He spent the academic year 1971—72 as a Visiting Lecturer at Queen's University, Kingston, Canada, and became Assistant Professor at ETH in 1972. He is interested in new teaching methods, such as group teaching and laboratory-oriented teaching. His research interests are in the fields of adaptive control and system identification.

First Semester: The Use of Computers (4 hours per week)
Introduction to programming using PASCAL; examples from data-processing and numerical analysis; development of simple programs for Apple computers.

Fourth Semester: Computer Techniques I (3 hours per week)
Advanced PASCAL programming, data structures, algorithms, compilers, debuggers, operating systems.

Fifth Semester: Computer Techniques II (3 hours per week)
Hardware of minicomputers and microcomputers, structures, technologies, interfaces, assembly-level programming, real-time programming.

In addition, there are compulsory courses in linear algebra and numerical mathematics.

2.2 Compulsory Control Education in the Electrical Engineering Department

Every student has to attend the following three control courses:

Fourth Semester: Control Systems I (3 hours per week)
Theory of linear continuous-time systems; system equations; input—output and state-variable representation; Controllability, observability, stability.

Fifth Semester: Control Systems II (2 hours per week)
Analysis and synthesis of continuous-time systems; stationary and transient behaviour; stability, sensitivity; compensation, state feedback, observer, optimal control.

Sixth Semester: Control Systems III (2 hours per week)
Description, analysis and synthesis of linear discrete-time systems; sampling; time and z-domain description; controllability, observability; stability, stationary and transient behaviour; realization of sampled data control; compensation, state feedback, observer.

2.3 Options in Control and Computer Science

There are many options in the seventh and eighth semesters for students interested in computers and control, for example:

- control theory (nonlinear systems and optimal control);
- control techniques (application-oriented);
- power electronics and drives;
- signals and systems;
- sensors;
- software engineering;
- real-time software;
- simulation.

Students have to take a minimum number of courses in the seventh and eighth semesters (credit system). The Control Group also offers a short course in PDP 11 programming.

3. The Use of Computers in Control Education

A brief summary of the principal uses of computers in control education is given here.

3.1 Classroom

Simulation studies or computer-controlled processes are used to illustrate the behaviour of control systems.

3.2 Exercises

Students have to design several control systems, sitting at terminals during the exercise periods.

3.3 Laboratory Experiments

Many small processes controlled by computers are available for one-afternoon exercises in the fifth and sixth semesters.

3.4 Student Projects

The main use of computers is in students' projects. Design and analysis of control systems or implementations are typical projects.

3.5 Graduate Programme

Much use of computers is also made at graduate level. Experimental verification of theoretically obtained results is often an integral part of a doctoral dissertation.

4. Hardware and Software for Educational Use: Overview and Experience

4.1 Hardware

The following computers are those used primarily in control education at ETH:

1 VAX 11/780
3 PDP 11/03's with process-control interface
1 PDP 11/34 with process-control interface
2 PDP 11/45's
1 PDP 11/60 with process-control interface
1 HP 1000 with process-control interface
1 VISOGYR industrial controller with real-time BASIC (Landis & Gyr Zug AG)
1 SESTEP programmable logic controller (Sprecher & Schuh)

All offices, laboratories and classrooms are connected by a local area network. This provides access to many additional computing facilities within ETH.

4.2. Software

The software used and the philosophy behind it will be described in some detail, because it is crucial for successful use of computers, and because the various control laboratories differ mainly in this respect. Two typical areas are first treated: languages and packages in use for simulation and for computer-aided control systems design. Thereafter, some aspects of educational practice in real-time programming are described.

In addition the Control Group also maintains a large collection of subroutines for control systems design (AUTLIB) to be used with high-level programming languages.

5. Software for Simulation and Computer-Aided Control Systems Design

In the last few decades, digital simulation has emerged as one of the most important tools for research as well as development in control theory. However, as no universally usable simulation software is available, a large number of simulation packages are needed to cover all types of simulation and at the same time provide packages as user-friendly as possible. Therefore, a large number of simulation packages are available at ETH, many of which have been developed locally either in part or fully. In the following list of simulation packages

available and used at ETH Zürich, no reference to alternative products (superior or inferior) is made. Furthermore, the mention of any present in-house software project neither implies nor guarantees the future availability of any product.

5.1. ACSL (Advanced Continuous Simulation Language)

An easy-to-learn, well-structured interactive simulation language for continuous and sampled data systems, ACSL includes features such as macros, procedurals and nonlinear functions. In the interactive mode, the user can change model parameters and perform simulations using color graphics at terminals, as well as analyze a simulated system through its Jacobian matrix and eigenvalues.

Available from Mitchell & Gauthier Associates, Inc., ACSL is presently used in the first simulation lectures, and in research projects at ETH Zürich.

5.2. COSY (COmbined SYstems)

This is a simulation language for combined continuous and discrete systems, and offers a high-level input language. The syntactical language definition of COSY has been defined using a general-purpose parser. At present, the COSY definition describes a state-of-the-art simulation language with a structurability and versatility not available in any other language. Preprocessor and run-time systems are presently not available (see Cellier and Bongulielmi, 1979). A subset of COSY (SYSMOD) is currently implemented by Systems Designer Ltd. on command by the British Ministry of Defence (Baker and Smart, 1983).

The general-purpose parser is one part of the SYNTAX package. Using this package it is possible to test a syntax for ambiguity and consistency. After the syntax has been found to be correct, it is possible to test whether examples written in the language are syntactically correct. Furthermore, it is possible to plot syntax diagrams automatically. Most of the SYNTAX package was developed at ETH Zürich in the late 1970s; the plot-generating part originates from the Institute of Computer Science, ETH (Bongulielmi and Cellier, 1979).

5.3. CSMP-6000

A batch-oriented simulation language for continuous systems, this well-known IBM simulation language was the only language used in the first simulation lectures (continuous simulation) until two years ago. In the last two years, students have been able to choose between ACSL and CSMP.

5.4 DARE-ELEVEN (Differential Analyzer REplacement)

Most of this package, an interactive version of the DARE simulation

language, is written in assembler for the PDP 11/RT, making the package fast but badly maintainable and not portable.

Developed at the University of Arizona, Tucson, AZ, this language was used in one laboratory until a year ago, when it was replaced by DARE-INTERACTIVE.

5.5 DARE-P

This simulation language for continuous systems features a very versatile postprocessor for tabular and graphical output of simulation results. The connection between the simulation package and the postprocessor can be seen as one of the first attempts to connect a simulation package to a data base. The language was developed at the University of Arizona, Tucson, AZ.

5.6 DARE-INTERACTIVE

An extension to DARE-P, DARE-INTERACTIVE makes the language interactive in a fashion similar to that of DARE-ELEVEN. New features include color graphics, a run-time display and split-screen graphics. Future versions of this software will include additional modules for sensitivity analysis, replication and batch, and trend analysis. DARE-INTERACTIVE will eventually also provide access to a general data-base. It has been developed and remains under development by ETH Zürich, and runs under VAX/VMS.

5.7 DYMOLA

DYMOLA is a simulation processor to transform dynamic equations given in the form

$$F_i(x,\dot{x},\ddot{x},\ldots,u,\dot{u},\ldots) = 0 \qquad i = 1,\ldots,$$

to a system of first-order differential equations, and can be used as a preprocessor to SIMNON to prepare models. It was developed at the Lund Institute of Technology (LTH), Lund, Sweden, and runs under VAX/VMS.

5.8 EARLY DESIRE

This direct-execution simulation language belongs to the next generation of the DARE family. Developed at the University of Arizona, Tucson, AZ, this new package is not yet used frequently at ETH Zürich.

5.9 ENPORT-4

This interactive simulation language for models represented by bond-diagrams runs on the PDP 11; it was developed at MIT, Massachusetts, MA.

5.10 FORSIM-VI

A batch-oriented simulation language for solving partial differential equations, FORSIM-VI is used primarily by researchers from other departments; it is available from the Atomic Energy Commission of Canada at Chalk River, Ontario.

5.11 GASP-IV

This batch-oriented, FORTRAN-coded simulation package for combined continuous/discrete systems, developed by Pritsker & Associates in the 1970s, is a true subset of GASP-V. It is available, but used only in connection with GASP-V.

5.12 GASP-V

This batch-oriented simulation package for combined continuous/discrete systems provides special facilities for partial differential equations and discontinuities. GASP-V also contains a large number of different integration algorithms. GASP-V was used in the latter part of the simulation lectures (discrete simulation) until two years ago, when it was replaced by SLAM-II, another successor of GASP-IV, which provides additional features for discrete system modelling. GASP-V is still used in research for the analysis of strongly discontinuous and coupled algebraic/differential systems. GASP-V was developed from GASP-IV at ETH Zürich in the late 1970s, and has been distributed worldwide.

5.13 GASP-V/INTERACTIVE

This is an interactive version of the GASP-V package. Through the use of MIDGET and an interactive postprocessor from the DARE family, the user can enjoy a very flexible version of the powerful GASP-V package. GASP-V/I was and is being developed at ETH Zürich, and runs under VAX/VMS.

5.14 GASP-VI

An extension to the GASP-V package, GASP-VI includes discrete processes, a more versatile list-processing mechanism, and enhanced output facilities. It is under development at ETH Zürich (Cellier and Rimvall, 1982).

5.15 MICRODARE-IV

This fix-point version of the DARE simulation program is used for laboratory instrumentation. It was developed at the University of Arizona, Tucson, AZ.

5.16 SDL (Simulation Data Language)

A portable relational data base specially adapted to storage and retrieval of data from simulations, SDL can easily be connected to any simulation program providing for a FORTRAN interface. Developed by

Pritsker & Associates, Lafayette, Indiana, IL, SDL is used in the latter part of the simulation lectures (discrete simulation) in connection with SLAM-II.

5.17 SIMNON

SIMNON, an interactive simulation program for continuous and sampled data systems, features a very natural model description as well as an easy-to-use dialogue form. It was one of the first packages to include terminal graphics to support interactive analysis/synthesis/optimization. Developed at the Lund Institute of Technology (LTH), Lund, Sweden, SIMNON runs under VAX/VMS.

5.18 SLAM-II

SLAM-II is a batch-oriented simulation language using a PERT network description of discrete models to be simulated. As SLAM-II is a superset of GASP-IV, it can also be used for combined continuous/discrete simulation with only part of the system described by a network. SLAM-II was developed by Pritsker & Associates. It is presently used in the latter part of the simulation lectures (discrete simulation).

5.19 THTSIM

An interactive simulation language for models represented by bond-diagrams, THTSIM runs on the PDP 11. It was developed at Twente University, The Netherlands.

5.20 TRNSYS (Transient System Simulation Program)

This is a batch-oriented simulation language specially designed by the Solar Energy Laboratory, Wisconsin, WI, to simulate solar-heating systems. A simplified coded input gives access to several premodelled components and output facilities.

Although there is a large common area between the fields of simulation software and computer-aided control systems design (CACSD) software, the two fields are here treated separately. By CACSD systems we understand packages providing a maximally user-friendly interface to algorithms, data bases and graphics packages. Some of the CACSD systems mentioned therefore also contain simulation features.

5.21 IMPACT (Interactive Mathematical Program for Automatic Control Theory)

IMPACT is the newest CAD project at ETH Zürich. It is to replace and enhance INTOPS. The package will be fully interactive, using a simple but versatile command language similar to that of MATLAB. Supported data structures will include matrices, polynomial matrices and linear as well as nonlinear system descriptions. Access to a data base will be provided. Several new polynomial algorithms are to be included, as well as a multitude of control-theory algorithms.

The package is implemented in Ada in a portable manner. The block structure of the package makes extensions relatively simple; in particular, very general data-base and graphics-device interfaces exist (Rimvall, 1983a; Cellier and Rimvall, 1983).

5.22 INTOPS (INTeractive OPerationS)

INTOPS was developed at ETH Zürich during the 1970s and incorporates many partly externally produced control algorithms. This FORTRAN-coded software system gives the user access to a fairly large number of algorithms for control theory in an interactive manner (several of these algorithms are also included in the AUTLIB library). This software is implemented on a PDP 11/34 under RSX. Some parts require a Hewlett-Packard 2648 terminal for execution.

5.23 MATLAB (MATrix LABoratory)

Developed at the University of New Mexico, NM, MATLAB provides an easy-to-use, interactive interface to the matrix-manipulation algorithms of LINPACK and EISPACK. It is probably the most widely used program at ETH Zürich at the present time.

5.24 MIDGET (Menu-driven Interactive Development system for Generic Engineering Tasks)

MIDGET was developed in 1983 at ETH Zürich. It is a software package standing between the user and the VMS operating system of our VAX computer. MIDGET simplifies access to several software packages and performs several accounting functions. MIDGET can thus be considered a special-purpose operating system implemented on the basis of VMS. Highlights of the MIDGET system are as follows.

(a) MIDGET gives access to several development systems (in particular, a development system for the construction of new development systems makes extensions to MIDGET an easy task). Generally, a development system provides the user with a number of commands (typically 5—30) adequate for solving a particular task. These commands are presented in a menu form; the use of "syspics" (pointers to the specific problem part being treated) speeds up processing considerably. At the present time, development systems exist for many of the simulation languages mentioned here and for the syntax package. Systems are planned for the development of larger software packages (in FORTRAN, PASCAL or Ada), which will include an automatic updating feature.

(b) A MIDGET internal-accounting system adiministers several functions not supported by the standard operating system. For example, it is possible to split any user account into several logical subaccounts; each user obtains access to a directory (logical work-area) of his own

after typing in a private password. This is especially practical for accounts on which several students work: the disk quota is used optimally, and some file security is ensured.

(c) A jump facility gives the user easy access to his directory tree.

(d) Automatic terminal initialization makes it possible for other packages to adjust automatically to the terminal type used.

MIDGET is presently implemented on a VAX under the VMS operating system. In principle, however, a system similar to MIDGET would be implementable on any interactive system (Rimvall, 1983b).

Table I indicates at which levels of education the various packages for simulation and computer-aided design are used. Furthermore, a distinction is made between whether a package is used as a tool only, or whether development of the package itself is undertaken at ETH.

TABLE I

ETH Software Packages for Use in Simulation and Computer-Aided Design [a]

Language or software package	Use at various levels						Use as tool	Development at ETH
	Semesters 5 and 6	Semesters 7 and 8	Laboratory	Student project	Graduate prgramme	Research		
ACSL	0	2	0	2	1	1	2	1
COSY	—	—	—	2	0	2	—	2
CSMP	0	2	0	1	0	1	2	0
DARE-11	0	0	0	0	0	0	0	0
DARE-P	1	0	0	2	1	2	2	2
DARE-INT	0	0	2	2	0	2	2	2
DESIRE	0	0	0	0	0	0	0	0
DYMOLA	0	1	0	0	0	0	1	0
ENPORT-4	0	0	0	0	0	0	0	0
FORSIM	0	0	0	1	0	1	1	1
GASP-IV	0	0	0	0	0	0	0	0
GASP-V	0	0	0	1	2	2	2	2
GASP-V/I	0	0	0	1	0	2	2	2
GASP-VI	—	—	—	2	0	2	—	2
MICRODARE	0	0	0	1	0	0	1	1
SDL	0	1	0	1	0	1	2	1
SIMNON	1	1	0	2	2	0	2	0
SLAM-II	0	2	0	1	0	1	2	0
THTSIM	0	0	0	0	1	0	1	0
TRNSYS	0	0	0	0	1	1	1	1
IMPACT	—	—	—	2	0	2	—	2
INTOPS	2	0	0	0	0	0	2	0
MATLAB	1	0	0	2	2	0	2	0
MIDGET	2	1	1	2	2	2	2	2

[a] The symbols indicate the following: 2, much used; 1, used; 0, hardly ever used; —, not applicable/not available.

6. Practical Education in Real-Time Programming

6.1 Goals

In the Department of Automatic Control undergraduate student projects in the field of real-time programming are carried out regularly (2—5 per year) as an opportunity to get to know and use a real-time programming language environment extensively.

6.2 Project organization

Students should take part in the PDP 11 short course before their project starts. Good knowledge of PASCAL is assumed. The course in real-time software is recommended. Two recently developed real-time tools, namely Modula-2 (Wirth, 1980) (together with the real-time operating system kernel MODEL written in Modula-2 (Maier, 1982)) and PORTAL (Lienhard, 1978; Nägeli, 1981) are used on the PDP 11 computers under RT-11 (Modula-2) or RSX-11 (PORTAL).

Typical projects are the design and implementation of a control task for one of the more complex laboratory processes. Through some ten student projects, a model railway has been connected to a minicomputer (PDP 11/03), and two large program packages written in Modula-2 and PORTAL have been developed to control and supervise up to ten trains simultaneously. Another couple of projects have been carried out on a power-system model.

6.3 Experience

Normally, students have been able to learn Modula-2 in a few days starting from their knowledge of PASCAL. The time to get started with PORTAL has been slightly longer, because compared to PASCAL this language includes more special features than Modula-2.

It has become evident that the design of a real-time program is much more complex and time-consuming than the design of a sequential program: besides all commonly known problems of sequential programming, the following tasks cannot be neglected:

- getting started with the real-time operating system and with the real-time aspects of the programming language to be used;
- understanding synchronization (normally, in a real-time system several actions have to be carried out in parallel: therefore, a real-time program consists of several parallel processes which must be coordinated with each other by means of synchronization operations);
- learning interface programming (in many real-time systems, non-standard devices (e.g., A/D and D/A converters) are used: their handling needs special knowledge in system programming, e.g., interrupt handling).

The applicability of Modula-2 and MODEB as well as of PORTAL indicates that there is no need for a large number of synchronization concepts. The design of real-time programs becomes simpler if only a few powerful synchronization operations are available.

The efficiency of the programming environment used influences the programmers' productivity to a high degree. The advantages of the separate compilation of Modula-2 have been proved: a module of 500 lines can be compiled and linked with a set of other modules to an executable program within about two minutes. The PORTAL implementation on the PDP 11 does not provide separate compilations. Therefore, the whole program must be recompiled after each change, which takes about three-quarters of an hour for 3000 lines. Delays of this order of magnitude are hardly acceptable and may be the reason that such a tool is not used widely.

For both Modula-2 and PORTAL a symbolic debugger is available and has been recognized as a very important tool during the test phase.

6.4 Conclusions

A real-time operating system or a real-time language is a prerequisite for a practical education in real-time programming. The applications should be coded in a higher-level, PASCAL-like programming language. Alternatives to Modula-2 and MODEB or PORTAL are one of the numerous PASCAL dialects with real-time enhancement (e.g., Micro Power PASCAL) or, within a few years hence, Ada.

Besides the programming language, a suitable language environment in terms of editing, compiling, debugging, etc. is essential. Students should get to know a powerful system, so that they themselves will be able to estimate the influence of another language environment programmer's productivity.

Laboratory processes and models form another element of a successful education in real-time programming: experiments on laboratory processes cannot be replaced by theory or simulation.

7. Laboratory Processes and Experiments

We here provide a brief overview of many experimental applications. The goals to be reached are described for different courses, and related typical experiments are summarized. Additional details may be found in Mansour and Schaufelberger (1981).

7.1 Undergraduate Control Laboratory (fifth and sixth semester)

This feature consists of simple one-afternoon experiments to introduce sequencing control and classical control methods, of which examples are now given.

(1) Traffic lights. A model of a street intersection with five lanes in the city of Zürich is controlled by a computer. In automated operation, the system has six different states. Students write a LASIC program (Logical BASIC) for fully automated operation of the corner, and carry out experiments on the installation.

(2) High-bay warehouse. Wooden blocks can be stored and retrieved from a fully automated model warehouse with 360 storage places. Experiments with the crane-position control subsystem (preprogrammed in PASCAL) are carried out by students, using different control algorithms.

(3) Coin-exchanging machine. A fully automated coin-exchanging machine using small balls as coins is available. Students write real-time PASCAL programs for simple changing strategies, and operate the installation. Several balls may reside concurrently at different stages of the plant.

(4) Speed control of a DC motor. A 4 kW DC motor is controlled by thyristors. Students are required to investigate discrete proportional—integral—differential controllers programmed by assistants. Students can change the controller parameters and the sampling time, and investigate the system both theoretically and practically.

(5) Heating and ventilation control systems. A model of a room is available in this experiment. An industrial controller is used for temperature and air-flow control.

(6) Three-basin level control system. On—off and proportional—integral controllers are used to control two water levels in this system. Students verify the operation of all parts and the model obtained theoretically. They then operate the system with a preprogrammed solution.

(7) "Helicopter" model. A two-degree-of-freedom model of a helicopter with two drives is controlled by analogue or digital control.

7.2 Undergraduate Student Projects

Typical goals at this stage are the design and implementation of state-variable feedback and output feedback controllers for various processes. Students have much more time available for a project than for a laboratory assignment, so that they can carry out the entire design cycle.

(8) Three-mass—spring system. Output feedback controllers have been designed and implemented on this sixth-order system.

(9) DC servo. State-variable feedback control has been implemented successfully on a fourth-order servo by using an observer. Two systems are controlled simultaneously by use of a real-time operating system.
(10) Inverted pendulum. An inverted pendulum is controlled by various control algorithms. Swing-up strategies have been developed for a single-stick pendulum, and stable controllers have been realized for all equilibrium states of a two-stick pendulum.
(11) Model railway. A model railway has been connected to a PDP 11/03 and is used for projects in real-time programming. Program packages written in Modula-2 as well as in PORTAL have been implemented to control and supervise up to ten trains simultaneously.

7.3 Graduate Control Laboratory

Some experiments have been especially designed to complement the theoretically oriented lectures in some of the graduate control courses, i.e., identification, adaptive control and control by microprocessors. The experiments in adaptive control have been summarized by Schaufelberger (1977).

(12) Adaptive control of DC generators. Several adaptive-control methods (model reference, self-tuning) are used to control small DC generators with varying turbine speed.
(13) On-line identification. Several programs are available for on-line identification of different electromechanical systems.
(14) Speed and tension control in a tape transporting machine. Design and implementation of a controller consisting of two proportional—integral controllers and a static decoupling network is another example of a laboratory assignment at this stage.

7.4 Graduate Student Projects

The implementation of advanced control strategies on nontrivial examples is the goal at this stage.
(15) Speed and tension control in a tape transporting machine. State-variable feedback control with observer has been implemented on this two-input/two-output system.
(16) Positioning of antennas. Adjustment of antennas for optimum data transmission as an example of extremalizing control has been designed and implemented.

7.5 Research Projects

Results as obtained in theoretical investigations in research work are often implemented to obtain insight into the feasibility of the results obtained.

(17) Adaptive control. Identification and adaptation techniques can easily be tested by changing the masses of the three-mass—spring system manually.
(18) Power-systems model. A computer-controlled power-systems model with three generators is under construction jointly with other groups in the department. It operates at 4 kW and will be used for experimental work on new control algorithms in power systems.

8. Conclusions

Simulation, computer-aided control system design and the implementation of control systems by real-time languages are important aspects of control engineering. The proper use of computers in engineering education requires considerable expenditure on installations and on manpower to develop and maintain such installations and program packages and languages.

The infrastructure available in this area to the Control Group of ETH Zürich has been described to some extent in this paper to provide information about the actual situation in Zürich. We are conviced that efforts in this area are justified by the improvements in teaching and research that have resulted from the increased use of computers.

References

Simulation and Computer-Aided Control Systems Design

Baker, N.J.C. and Smart, P.J. (1983). "The SYSMOD simulation language", in W. Ameling, ed., *Proceeding of the First European Simulation Conference, ESC'83.* Berlin: Informatik-Fachberichte, Springer.

Bongulielmi, A.P. and Cellier, F.E. (1979). "On the usefulness of using deterministic grammars for simulation languages", *Proceedings of the SWISSL Workshop, St. Agata (Italy)* (to appear in *Simuletter*).

Cellier, F.E. and Bongulielmi, A.P. (1979). "The COSY simulation language", *Proceedings of the IMACS Congress on Simulation of Systems.* Amsterdam: North-Holland.

Cellier, F.E. and Rimvall, M. (1982). "The GASP-VI simulation package for process-oriented combined continuous and discrete system simulation", *Proceedings of the 10th IMACS World Congress on Simulation and Scientific Computation, Montreal (Canada).*

Cellier, F.E. and Rimvall, M. (1983). "Computer aided control systems design", in W. Ameling, ed., *Proceedings of the First European Simulation Conference, ESC'83.* Berlin: Informatik-Fachberichte, Springer.

Rimvall, M. (1983a). *IMPACT, Interactive Mathematical Program for Automatic Control Theory: A Preliminary User's Manual.* Zürich, Switzerland: Institute for Automatic Control and Industrial Electronics, ETH Zürich.

Rimvall, M. (1983b). *MIDGET Users' Guide.* Zürich, Switzerland: Institute for Automatic Control and Industrial Electronics, ETH Zürich.

Real-Time Programming

Maier, G. (1982). "MODEB: a real time operating system kernel written in the high-level programming language Modula-2" *3rd IFAC/IFIP Symposium on Software for Computer Control, SOCOCO'82, Madrid, Spain, 5—8 October 1982, Preprints,* pp. 15—21.

Maier, G. and Kuster, F. (1981). "Echtzeitprogrammierung in einer höheren Programmiersprache (Modula-2): steuerung einer Modelleisenbahn", *Zusammenfassung der Referate des 4. DECUS München Symposiums an der Universität Konstanz, 18—20 March 1981.*

Nägeli, H.H. (1981). "Programmieren mit PORTAL: eine Einführung", Landis & Gyr Zug AG.

Wirth, N. (1980). "Modula-2", *Instituts für Informatik, ETH Zürich, Bericht 36.*

Wirth, N. (1982). *Programming in Modula-2.* Berlin: Springer.

Laboratory Process and Experiments

Huguenin, F., Kraus, F., Kuster, F., Maier, G., Maletinsky, V., Schaufelberger, W., Scheuren, J.J. and Toedtli, J. (1980). "Automatische Steuerungen und Regelungen mit Mikrorechnern", *Fachgruppe für Automatik, ETH Zürich, Report 80-01.*

Mansour, M. and Schaufelberger, W. (1981). "Digital computer control experiments in the control group of ETH Zürich", *8th IFAC World Congress, Kyoto.*

Schaufelberger, W. (1977). "Laboratory experiments in adaptive control", *IFAC Symposium on Trends in Automatic Control Education, Barcelona.*