COMPUTER-AIDED CONTROL SYSTEMS ENGINEERING

edited by

M. JAMSHIDI

and

C. J. HERGET

COMPUTER-AIDED CONTROL SYSTEMS ENGINEERING

edited by

M. JAMSHIDI
CAD Laboratory for Systems
Department of Electrical and Computer Engineering
University of New Mexico
Albuquerque, NM
U.S.A.

and

C. J. HERGET
Lawrence Livermore National Laboratory
Livermore, CA
U.S.A.

1985

NORTH-HOLLAND
AMSTERDAM · NEW YORK · OXFORD

ISBN: 0 444 87779 7

Publishers:
ELSEVIER SCIENCE PUBLISHERS B.V.
P.O. Box 1991
1000 BZ Amsterdam
The Netherlands

Sole distributors for the U.S.A. and Canada:
ELSEVIER SCIENCE PUBLISHING COMPANY, INC.
52 Vanderbilt Avenue
New York, N.Y. 10017
U.S.A.

Library of Congress Cataloging in Publication Data
Main entry under title:

Computer-aided control systems engineering.

1. Automatic control--Data processing--Addresses, essays, lectures. 2. Computer-aided design--Addresses, essays, lectures. I. Jamshidi, Mohammad. II. Herget, Charles J.
TJ213.7.C64 1985 629.8'95 85-10138
ISBN 0-444-87779-7

PRINTED IN THE NETHERLANDS

PREFACE

It was only a few years ago that the only tools for analysis and synthesis available to the control engineer were paper, pencil, slide rule, spirule, and analog computer. The tools and the methods were simple enough that an engineer could easily master them in a relatively short time. However, over the past twenty-five years, control theory has evolved to a state where the digital computer has become a requirement for the control system engineer, and Computer-Aided Control System Design (CACSD) has emerged as an indispensable tool.

A good CACSD system draws on expertise from many disciplines including aspects of computer engineering, computer science, applied mathematics (e.g. numerical analysis and optimization), as well as control systems engineering and theory. The need for such a breadth is partially responsible for the paucity of high quality CACSD software today, and indeed, CACSD must be considered in its infancy. Professional societies, such as the IEEE Control Systems Society and IFAC, have recognized the need for increased professional activities in this field and have sponsored a number of Workshops and Symposia on this subject, the IEEE's being held in the United States and IFAC's being held in Europe and the United Kingdom. Most of these meetings did not have formal proceedings for the outside user to read, and papers on CACSD have not been readily available to the professional community. In recognition of the relevance of this topic, the IEEE Control Systems Society published a Special Issue on CACSD in the *Control Systems Magazine* (December 1982) and the IEEE has published a Special Section on CACSD in the *Proceedings of the IEEE* (December 1984).

It is the object of this volume, therefore, to make available a thorough and up-to-date account of Computer-Aided Control Systems Engineering by collecting together in one place some of the various papers presented at these workshops and symposia as well selected papers from some of the professional journals. We point out that we have used the term computer-aided control systems engineering in the title of this book instead of computer-aided control system design to emphasize the much broader facets of the problem which the control engineer faces. *I.e.*, aspects of modeling, analysis and simulation must always be considered in addition to design. In addition to the selected technical papers, there is a collection of software summaries describing CACSD packages from around the world.

The book consists of nineteen individual contributions from thirty-eight authors and the Software Summaries describe thirty-seven different packages. The papers fall into three general categories: (*i*) computer-aided control system design packages and languages, (*ii*) perspectives and expository looks at computer-aided control system design, and (*iii*) algorithms and techniques for CACSD.

The editors would like to take this opportunity and thank every one of the authors for their worthy contributions to CACSD in general and to this volume in particular. We wish to express our gratitude to Ms. Fran McFarland at Lawrence Livermore National Laboratory and

Ms. Gladys Ericksen at the University of New Mexico for their tireless assistance to the editors. We want to express our appreciation to Dr. Gerard Wanrooy and his staff at North Holland in Amsterdam for their collaboration. We also want to thank IEEE and Springer-Verlag for permitting us to use some articles on which they hold the copyright in this book. The first editor would also like to express his appreciation for the support he received from Dr. R. H. Seacat, Chairman of the Department of Electrical and Computer Engineering, and Dr. G. W. May, Dean of Engineering, at the University of New Mexico.

M. Jamshidi
Albuquerque, NM
U.S.A.

C. J. Herget
Livermore, CA
U.S.A.

CONTRIBUTOR LIST

William F. Arnold, III
Naval Weapons Center
China Lake, California, USA

K. J. Åström
Lund Institute of Technology
Lund, Sweden

S. N. Bangert
Systems Control Technology, Inc.
Palo Alto, California, USA

S. P. Bingulac,
University of Belgrade
Belgrade, Yugoslavia

François E. Cellier
University of Arizona
Tucson, Arizona, USA

Michael J. Denham
Kingston Polytechnic
Kingston-upon-Thames, England

Howard Elliott
University of Massachusetts
Amherst, Massachusetts, USA

A. Emami-Naeini
Systems Control Technology, Inc.
Palo Alto, California, USA

Michel A. Floyd
Integrated Systems, Inc.
Palo Alto, California, USA

G. F. Franklin
Stanford University
Stanford, California, USA

Dean K. Frederick
Rensselaer Polytechnic Institute
Troy, New York, USA

Charles J. Herget
Lawrence Livermore National Laboratory
Livermore, California, USA

M. Jamshidi
University of New Mexico
Albuquerque, New Mexico, USA

Russel P. Kraft
Mechanical Technologies, Inc.
Latham, New York, USA

Alan J. Laub
University of California
Santa Barbara, California, USA

Eugene A. Lee
The Aerospace Corporation
El Segundo, California, USA

Gordon K. F. Lee
Colorado State University
Fort Collins, Colorado, USA

Larry L. Lehman
Integrated Systems, Inc.
Palo Alto, California, USA

J. N. Little
Systems Control Technology, Inc.
Palo Alto, California, USA

A. G. J. MacFarlane
Cambridge University
Cambridge, England

J. M. Maciejowski
Cambridge University
Cambridge, England

D. Q. Mayne
Imperial College
London, England

R. Morel
Los Alamos National Laboratory
Los Alamos, New Mexico, USA

W. T. Nye
University of California
Berkeley, California, USA

R. V. Patel
Concordia University
Montreal, Canada

W. R. Perkins
University of Illinois
Urbana, Illinois, USA

E. Polak
University of California
Berkeley, California, USA

Magnus Rimvall
Swiss Federal Institute of Technology
Zurich, Switzerland

Tahm Sadeghi
Fairchild Republic Company
Farmingdale, New York, USA

Chr. Schmid
Ruhr University
Bochum, Federal Republic of Germany

J. Schotik
University of New Mexico
Albuquerque, New Mexico, USA

Sunil C. Shah
Integrated Systems, Inc.
Palo Alto, California, USA

P. Siegel
University of California
Berkeley, California, USA

H. Austin Spang, III
General Electric R&D Center
Schenectady, New York, USA

Diane M. Tilly
Lawrence Livermore National Laboratory
Livermore, California, USA

P. P. J. van den Bosch
Delft University of Technology
Delft, The Netherlands

P. J. West
University of Illinois
Urbana, Illinois, USA

T. Wuu
University of California
Berkeley, California, USA

T. C. Yenn
University of New Mexico
Albuquerque, New Mexico, USA

TABLE OF CONTENTS

Section 1. PACKAGES

COMPUTER-AIDED CONTROL SYSTEMS ENGINEERING
M. Jamshidi and C.J. Herget (Editors)
© Elsevier Science Publishers B.V. (North-Holland), 1985

COMPUTER AIDED TOOLS FOR CONTROL SYSTEM DESIGN

K.J. Aström

Department of Automatic Control
Lund Institute of Technology
Lund, Sweden

The paper describes experiences of development and use of interactive software for computer aided design of control systems. The experiences are drawn from a comprehensive set of packages for modeling, identification, analysis, simulation and design, which have been in use for about a decade. Problems associated with structuring, portability, maintainability, and extensibility are discussed. Experiences from uses of the packages in teaching and industrial environments are discussed. Views on future development of CAD for control systems are also given.

1. INTRODUCTION

Thirty years ago pencil, paper, slide rules and analog computers were the major tools for analysis and synthesis of control systems. The methods and the tools were so simple that an engineer could master both problems and tools. Many new methods for analysis and design of control systems have emerged during the last 30 years. These methods differ from the classical techniques. They are more sophisticated analytically and their use require extensive calculations. An extensive subroutine library is required to apply these methods to a practical problem. Even if such a library is available it is a major effort to write the software necessary to solve a particular problem. This means that modern control theory is costly to use. Another drawback is that the problem solver interacts with his tools (the computer) via intermediaries (programmers). This easily leads to confusion and mistakes. The intensive interaction between problem formulation and solution is also lost.

Based on experience from industrial application of modern control theory in the early sixties it was clear to me that modern control theory could be used very successfully in a research laboratory or at a university. It was, however, equally clear that the methods would not be widely used in normal engineering practice unless the proper tools were developed. A number of projects were therefore carried out in order to efficiently develop the proper tools for using control theory cost. This paper summarizes some of the software developed in the project and experiences from its use.

The projects were based on the idea of combining an engineer's intuition and overview with digital computing power. The approach included development of design techniques and man-machine interfaces, for interactive use of the computer. Graphics is of course a major ingredient.

The paper is organized as follows. A brief overview of the projects is given in Section 2. Interaction principles are discussed in Section 3. Data structures are discussed in Section 4. The comprehensive set of program packages which is one result of the projects is described in Section 5. Some special problems associated with large systems are discussed in Section 6. Experiences from use of the packages in university and industrial environments are presented in Section 7. Sections 8 and 9 give suggestions for future work and conclusions.

2. THE PROJECTS

The objectives of the projects were to make advanced methods for modeling, analysis and design of control systems easily accessible to engineers, researchers and students and to explore the potentials of interactive computing for control system design, see Åström and Wiesl nder (1981). When the projects were initiated around 1970 we had extensive experience of analog simulation, programming in Fortran, Basic, and APL. There was common consensus about the power of digital computation and the superiority of the man-machine interaction in analog simulation. We were familiar with the ease of debugging and running programs in an interactive implementation like APL. But we were also aware of the limited portability of such programs and of the difficulties of extending such systems. The software was developed in close interaction with the users. A system outline was sketched. The ideas were discussed in seminars. A system of moderate size was implemented and tested by several users. The system was then modified. In the initial phases we were also quite willing to scrap a system and start all over again. As the projects progressed we got a much better feel for what could be done and how it should be done. It also became clear that a fairly comprehensive package was necessary to evaluate the ideas. Such packages were also developed. They went through many revisions to improve portability, modularity and efficiency. The packages matured and reached a stable state around 1979. From then on we have only made moderate maintenance and updating. Uses of the packages have increased rapidly since then.

What can be done with interactive computing depends much on the available hardware. Since the hardware has undergone a revolutionary development over the past ten years it is useful to describe what was available in the projects. When the activity was started, in 1971, we had access to a DEC PDP 15 with 32 kbytes of core memory, a 256 kbytes disk and a storage oscilloscope. After a few years the activity was moved to large mainframe computers. We are currently using a DEC Vax-11/780 with 2 Mbyte of fast memory and a 600 Mbyte disc for most of the work. Our sponsoring agency (STU) also introduced constraints by insisting that the programs should be portable and useful to industry. One way to achieve this was to use standard Fortran.

The projects have resulted in a comprehensive set of program packages for modeling, identification, analysis, simulation and design of control systems. We have several years experience of using these packages in different university and industrial environments. Ideas on the use of graphics and interactive computing in future systems have also been developed.

3. INTERACTION PRINCIPLES

When designing a system for man-machine interaction it is important to realize that there is a wide range of users, from novices to experts, with different abilities and demands. For a novice who needs a lot of guidance it is natural to have a system where the computer has the initiative and the user is gently led towards a solution of his problem. For an expert user it is much better to have a system where the user keeps the initiative and where he gets advice and and help on request only. Attempts of guidance and control by the computer can easily lead to frustration and inefficiency. It is highly desirable to design a system so that it will accomodate a wide range of users. This makes it more universal. It also makes it possible to grow with the system and to gradually shift the initiative from the computer to the user as he becomes more proficient. Many aspects on interaction principles and their implementation are found in Barstow et al. (1984).

To obtain an efficient man-machine interface it is desirable to have hardware with a high communication rate and a communication language with a good expression power. When our projects were started we were limited to a teletype and a storage oscilloscope. There were also limited experiences of design of man-machine interfaces. The predominant approach was a question-and-answer dialog, see e.g. Rosenbrock (1974).

In our projects it was discovered at an early stage that the simple question-and-answer dialog was too rigid and very frustrating for an experienced user. The main disadvantage is that the computer is in command of the work rather than the user. This was even more pronounced because of the slow input-output device (teletype) which was used initially.

Our primary design goal was to develop tools for the expert. A secondary goal was to make the tools useful also for a novice. To make sure that the initiative would remain with the user it was decided to make the interaction command oriented. This was also inspired by experiences from programming in APL. Use of a command dialog also had the unexpected effect that it was easy to create new user defined commands. The packages could thus be used in ways which were not anticipated when they were designed. The decision to use commands instead of a question and answer dialog had far reaching consequences. A more detailed discussion of the different types of dialogs and of our experiences of them is given in Wieslander (1979a,b). Today there is a wide range of experiences of designing man-machine interfaces in many different fields. Our own conclusions agree well with those found in Newman and Sproull (1979), and Foley and van Dam (1983) although their conclusions are based on different hardware.

Examples of commands

The structure of the commands we introduced will now be described. The general form of a command is

```
NAME LARG1 LARG2...  ←  RARG1 RARG2...
```

A command has a name. It may also have left arguments and right arguments. The arguments may be numbers or names of objects in a data base. In our packages the objects are implemented as files because this is a simple way to deal with objects having different types. A few examples of commands are given to further illustrate the notion of a command. The command

```
MATOP S  ←  A * B + C
```

simply performs the matrix operation expressed to the right of the arrow. The command

```
POLOP S  ←  A * B + C
```

performs the same operation on polynomials. The command

```
INSI U 100
   >PRBS 4 7
   >X
```

generates an input signal of length 100 called U. The command has options to generate several input signals. The options are selected by additional subcommands. PRBS is a subcommand which selects a PRBS signal. The optional arguments 4 and 7 indicate that the PRBS signal should change at most every fourth sampling period and that its period should be 2^7-1. The subcommand EXIT denotes the end of the subcommands. The command

```
DETER Y  ←  SYST U
```

generates the response of the linear system called SYST to the input signal U. The command

```
ML PAR  ←  DAT N
```

fits an ARMAX model of order N to the data in the file called DAT and stores the parameters in a file called PAR. The command

```
OPTFB L S  ←  LOSS SYS
```

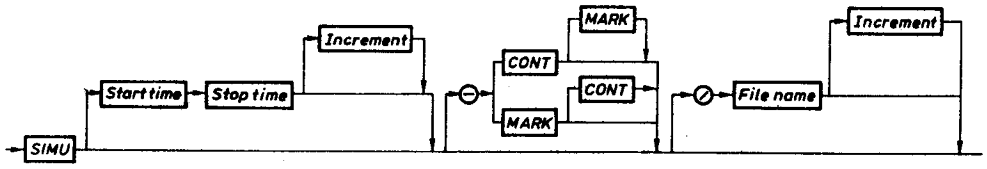

Figure 1
Syntax diagram for the command SIMU.

computes the optimal feedback gain L and the solution S to the steady state Riccati equation for the system SYS and the loss function LOSS.

Short form commands and default values

In a command dialog it is highly desirable to have simple commands. This is in conflict with the requirement that commands should be explicit and that it may sometimes be desirable to have variants of the commands. These opposite requirements may be resolved by allowing short forms of the commands. The standard form for the simulation command is SIMU. If no other command starts with the letter S it is, however, sufficient to type S alone. Another interesting possibility is to correlate a given command with the command list and to choose the command from the list which is closest to the given one. This automatically gives short form commands. The scheme will also be insensitive to spelling errors. It is, however, also dangerous because totally unexpected commands will be obtained. It is also useful to have a simple way of renaming the commands. Such mechanisms are now available in many systems. We have experimented with short form commands, command correlation, and renaming mechanisms. These functions are, however, not implemented in our standard packages.

Similar mechanisms may be used for commands with arguments by introducing a default mechanism so that previous values of the arguments are used unless new values are specified explicitly. The concept is illustrated by an example.

The syntax diagram for the command SIMU is shown in Fig. 1. The diagram implies that any form of the command which is obtained by traversing the graph in the directions of the arrows is allowed. For example the command

```
SIMU 0 100
```

simulates a system from time 0 to time 100. If we want to repeat the simulation a second time with different parameters it suffices to write

```
SIMU
```

The arguments 0 and 100 are then taken as the previously used values.

It follows from Fig. 1 that start and stop times and the initial time increment may be specified. It is also possible to mark curves by the argument MARK. A simulation may also be continued by using the end conditions of a previous simulation as initial values. This is done by the command extension CONT. The results of a simulation may also be stored in a file by appending /filename.

Macros and user defined procedures

Commands are normally read from a terminal in a command driven system. It is, however, useful to have the option of reading a sequence of commands from a file instead. Since this is analogous to a macro facility in an ordinary programming language the same nomenclature is adopted. See e.g. Wegner (1968). The construction

```
MACRO NAME
   Command 1
   Command 2
   Command 3
END
```

thus indicates that the commands 1, 2 and 3 are not executed but stored in memory. The command sequence is then activated simply by typing NAME.

Macros are convenient for simplification of a dialog. Command sequences that are commonly used may be defined as macros. A simple macro call will then activate a whole sequence of commands. The macro facility is also useful in order to generate new commands. Macros may also be used to rename commands. This is useful in order to tailor a system to the needs of a particular user. By introducing commands for reading the keyboard and for writing on the terminal it is also possible to implement menu driven dialogs using macros.

The usefulness of macros may be extended considerably by introducing commands to control the program flow in a macro, facilities for handling local and global variables and by allowing macros to have arguments. We will then have a mechanism for making user defined procedures.

Error checking

It is important in interactive systems to have test for avoiding errors. It is thus useful to check data types and to test problems for consistency whenever possible.

Implementation

It is straightforward to implement a command driven interactive program. The structure used in our packages is shown in Fig. 2. The main loop reads a command, decodes it and performs the required actions. All parts of Fig. 2 except the action routines are implemented as a package of subroutines called Intrac. These subroutines perform