Computer Aided Tools for Control System Design

Åström, Karl Johan

1989

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-3207)/1–40/(1989)

Computer Aided Tools for Control System Design

K. J. Åström

Department of Automatic Control
Lund Institute of Technology
December 1989

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
FINAL REPORT

Date of issue
December 1989

Document Number
CODEN: LUTFD2/(TFRT-3207)/1–40/(1989)

Author(s)
K. J. Åström

Supervisor

Sponsoring organisation

Title and subtitle
Computer-Aided Tools for Control System Design

Abstract

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.

The paper is published in M. Jamshidi and C. J. Herget, *Computer Aided Control Systems Engineering*, North-Holland 1985, pages 3–40.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
40

Recipient's notes

Security classification

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.

Published in M. Jamshidi and C. J. Herget, *Computer Aided Control Systems Engineering*, North-Holland 1985, pages 3–40.

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

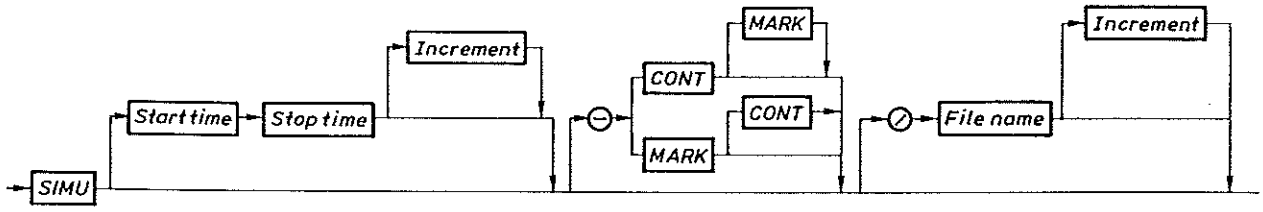

Figure 1
Syntax diagram for the command SIMU.

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.

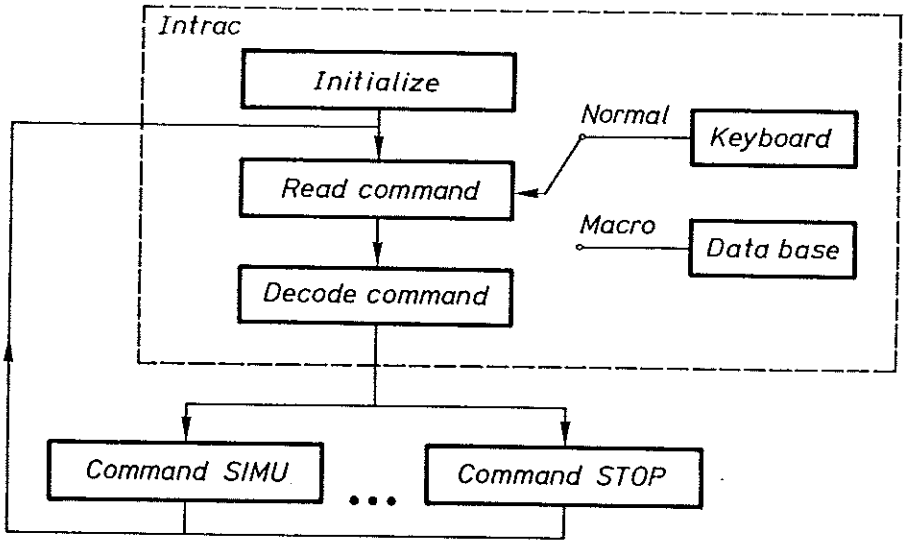

Figure 2
Skeleton flow chart for a command driven program.

command decoding, file handling and plotting. Intrac also contains facilities for creating macros and user defined functions. Macros may have formal arguments, local and global variables. They permit conditional and repeated execution of commands as well as nested use of macros. There are read and write commands, which can be used to implement menu dialogs. It is possible to mix command mode and question mode, since the execution of a macro may be suspended and resumed later. A description of Intrac is given in Wieslander and Elmqvist (1978) and Wieslander (1980a). The commands available in Intrac are listed in Appendix F.

To build a package using Intrac it is necessary to write the action routines i.e. the subroutines that performs the desired tasks. The commands are then entered in the command table of the command decoder. It is also easy to add a command to a package, to move commands between packages and to create special purpose packages. Intrac may thus be viewed as a tool for converting a collection of Fortran subroutines into an interactive package. Intrac has also been used to implement other packages by other groups.

The structure with a common user interface for all packages is advantageous for the user because the interaction and the macro commands are the same in all packages. This simplifies learning and use of the packages. The advantages of interactive computing environments for problem solving are now widely recognized. They are built into systems like Basic, Apl, Lisp and Smalltalk. There are also several systems, which provide interactive features in static computing environments. Speakeasy, CTS are typical examples which are similar to Intrac.

Descriptions of control systems problems require flexible data structures. Many problems may be characterized in terms of arrays only. Arrays will go a long way to describe linear systems in state space form and to describe signals. Many problems can be solved using a matrix language like MATLAB, Moler (1981) or its derivatives Matrix$_X$, Walker et al. (1982), CTRL-C, Little et al. (1984), or Impact, Rimvall and Cellier (1984). It is, however, clear that it is not sufficient to only have matrices. A detailed discussion of this is found in Aström (1984).

For simple systems with only one data type, like matrices, all data may be stored in a stack or in a simple array. A more sophisticated data structure was used in the Lund packages. Our experiences indicate that it would be very useful to have a more flexible system. It is probably a good idea to build a system around some general database system. The need for multiple descriptions of a system is one special problem which is conveniently solved using databases. A typical example is when a system is represented both as a transfer function and as a state equation. Small systems are not much of a problem because it is easy to transform from one form to another. Such computations may however be extensive for large systems. To obtain a reasonable efficiency it is then necessary to store the different descriptions. It may also be desirable to have models of different complexity for the same physical object as well as linearized models for different operating conditions. Since it is very difficult to visualize all possible combinations a priori, it is a useful to have a database system which admits modifications of the structure of the data.

System descriptions

Since dynamical systems is a fundamental notion, its representation becomes a key issue. Many different representations of systems are used in control theory. The ordinary differential equation model

$$\begin{cases} \dfrac{dx}{dt} = f(x, u, t) \\ y = g(x, u, t) \end{cases} \tag{1}$$

where x is the state vector, u the input vector and y the output vector, is a common case. Often the fundamental form of the equations is not (1), where the derivative is solved explicitly but rather

$$\begin{cases} F\left[\dfrac{dx}{dt}, x, u, t\right] = 0 \\ G(x, y, u, t) = 0 \end{cases} \tag{2}$$

The following discussion is restricted to systems of type (1). Other issues arise when operating with models of type (2). This is treated in depth in Elmqvist (1978, 1979a, 1979b). Partial differential equations and differential equations with time delay are also

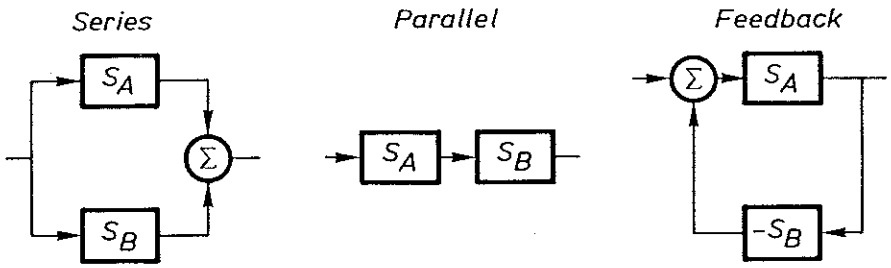

Figure 3
The basic system interconnections.

For more complex systems it is desirable to have appropriate notations for interconnected hierarchical systems. These notations should be such that details of the subsystem can be hidden and that signals and variables at the lower levels can be accessed in a well controlled fashion.

The system description introduced by Elmqvist (1977) in the simulation language Simnon has been very easy to operate with and very easy to teach. Elmqvist introduced the classes of continuous and discrete time systems defined as follows.

```
CONTINUOUS SYSTEM   <system identifier>
[INPUT    <simple variable>*]
[OUTPUT   <simple variable>*]
[STATE    <simple variable>*]
[DER      <simple variable>*]
[TIME     <simple variable>*]

[INITIAL
[Computation of initial values for state variables]

[Computation of auxiliary variables]
[Computation of output variables]
[Computation of derivatives]
[Parameter assignment]
[Initial value assignment]
END

DISCRETE SYSTEM  <system identifier>
[INPUT    <simple variable>*]
[OUTPUT   <simple variable>*]
[STATE    <simple variable>*]
[NEW      <simple variable>*]
[TIME     <simple variable>*]
TSAMP     <simple variable>*]

[INITIAL
[Computation of initial values for state variables]
[Computation of initial values for output variables]
[Computation of initial values for the TSAMP-variable]
SORT
```

situations it is desirable to have access to all variables. This can be achieved by using EXPORT ALL or some similar construction.

Since system interconnections are often visualized graphically there should be facilities for representing and manipulating system interconnections graphically as well as textually. Interesting ideas in this direction have been proposed by Elmqvist (1982).

It would also be desirable to have the notion of system type and facilities for creating instances of the type. This would give a simple way of generating special classes of systems. Linear systems can then be defined as

```
type LINEAR_STATE_SPACE_SYSTEM
INPUT  u: vector
OUTPUT y: vector
STATE  x: vector
DERIVATIVE  dx: vector
A B C D : matrix
y  = C*x + D*y
dx = A*x + B*u
END
```

A similar construction can be used for linear polynomial systems. Instance of linear systems can then be created by

```
S: LINEAR_STATE_SPACE_SYSTEM
```

The parameters can be accessed as

```
S.A = matrix (1 2 ; 3 4)
```

It is a nontrivial design issue to decide when and how dimension compatibility should be checked. This has to do with how arrays are implemented. From the user point of view it would, however, be desirable to define a linear system as was done above without a need for specifying the dimensions.

In some cases it is also desirable to be able to hide a system description so that a user of the system can only make operations like simulation. An example from teaching is in courses on system identification, where it is desirable for students to find the properties of an unknown system, or in courses on adaptive control, when it is desirable to check that an algorithm works on an unknown system. The possibility to hide details of a system description would also be a possibility to get controlled access to industrial models. This can be achieved by using the mechanisms introduced in Ada, where the declarations and a body of a procedure are separated, see DOD (1983).

System operations

Apart from interconnections there are many other operations that are desirable to perform on systems, e.g. computation of equilibrium values, simulation, linearization,

methods. It was, however, discovered that almost all methods could be obtained by combinations of correlation analysis, spectral analysis, least squares and maximum likelihood estimation. Commands were thus constructed to give primitives for these operations and the special methods were then implemented as macros which used the primitive commands. This approach is also a pedagogical way to structure the problem area.

Idpac can be viewed as a convenient way of packaging the research in systems identification that has been done at our department for a period of 15 years. Idpac has gone through several steps of development. It grew out of the software described in Aström et al. (1965). The latest version is described in Wieslander (1980b). The paper Aström (1980) gives the relevant theory for the parametric identification methods. It also contains a comprehensive set of examples of using Idpac. A summary of the commands are given in Appendix A. Descriptions of some of the Idpac macros are given in Gustavsson (1979). Typical examples of using Idpac are given in Gustavsson and Nilsson (1979).

Modpac

There are many ways to describe a control system. Nonparametric methods in the time and frequency domain can be used. Parametric descriptions like state equations, rational transfer functions and fractions of matrix polynomials may also be used. There are also many ways in which state equations can be transformed. For digital control it is necessary to go between continuous time and discrete time representations. All these problems can be handled by Modpac. The package also has facilities for finding the Kalman decomposition of a system and for calculating observers. Modpac is described in Wieslander (1980c). A list of the commands in Modpac is given in Appendix B.

Simnon

Simnon is a package for interactive simulation of nonlinear continuous time systems with discrete time regulators. The package also includes noise generators, time-delays, a facility for using data files from Idpac as inputs to the system and an optimizer.

Simnon allows a system to be described as an interconnection of subsystems. There are two types of subsystems, continuous time systems and discrete time systems. These were described in detail in Section 4. This makes Simnon well suited for simulation of digital control systems. Simnon has two abstract datatypes continuous system and discrete systems to describe the subsystems and a third type connecting system to describe the interconnenctions. The characteristics of Simnon are illustrated by an example.

Listing 1 gives a description of a feedback loop consisting of a continuous time process called PROC and a digital PI regulator called REG. The process is an integrator with input saturation. The interconnections are described by the connecting system CON.

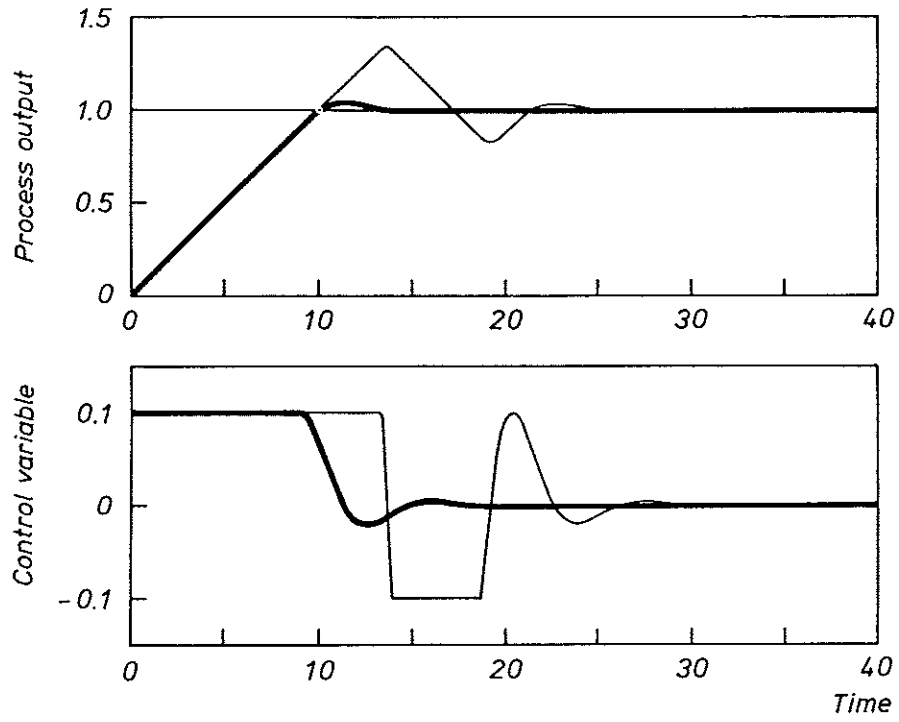

Figure 4
Results of simulation of process with a PI regulator.
Thin lines show results with ordinary regulator and thick lines show results for regulator with anti-windup.

The following annotated dialog illustrates how Simnon is used.

Command	Action
SYST PROC REG CON	Activate the systems.
AXES H 0 100 V -1 1	Draw axes.
PLOT yr y[proc] u[reg]	Determine variables to be plotted.
STORE yr y[proc] u[reg]	Select variable to be stored.
SIMU 0 100	Simulate.
SPLIT 2 1	Form two screen windows.
ASHOW y SHOW yr	{ Draw y with automatic scaling and yr with the same scales in first window.
ASHOW u	Draw u with automatic scaling in second window.

The result is shown by the curves in thin lines shown in Fig. 4. These curves show that there is a considerable overshoot due to integral windup. The regulator REG has anti-windup. The state of the regulator is reset when its output is equal to ulow or uhigh. The limits were set to ulow = -1 and uhigh = 1 in the simulations shown with thin lines in Fig. 4. These values are so large that the integral is never reset. The simulation, shows in thin lines in Fig. 4, thus correspond to a regulator without wind-up. The actual actuator limitations correspond to ulow = -0.1 and uhigh = 0.1. The commands

```
 1:   MACRO DESIGN ALPHA
 2:     ALTER Q1 3 3 ALPHA
 3:     FOR H = 0.5 TO 5 STEP 0.5
 4:       SAMP DSYS ← CSYS H
 5:       TRANS Q DSYS ← CSYS H
 6:       TRANS R DSYS ← CSYS H
 7:       OPTFB L ← DSYS
 8:       KALFI K ← DSYS
 9:       CONNECT CLSYS ← DSYS K L
10:       SIMU Y X ← CLSYS UREF
11:       PLOT X(1) X(7) X(8) XE(1) U
12:     NEXT H
13:   END (MACRO)
```

Line numbers have been introduced only to be able to describe the algorithm. A macro with the name DESIGN and the parameter ALPHA is defined on line 1. The macro definition ends on line 13. The 3,3 element of the matrix Q1 is assigned the value ALPHA on line 2. Line 3 is a repetition statement which repeats the commands 4 through 11 for sampling periods 0.5 to 5 with an increment of 0.5. The system description is sampled on line 4 and the criterion and the covariances are transformed on lines 5 and 6. The command on line 7 computes the optimal state feedback matrix L and the command on line 8 computes the Kalman filter gain. The command on line 9 forms a closed loop system composed of the original system the Kalman filter and the state feedback. The command on line 10 simulates the closed loop system with a reference input UREF. The command on line 11 plots state variables 1, 7 and 8, the estimate of the first state and the control signal. The following dialog illustrates how the macro may be used

```
EDIT FILE CSYS
INPUT UREF ← STEP
DESIGN 3
DESIGN 8
```

The system file is first edited. A step is generated as a command signal and the macro design is executed with parameters 3 and 8.

The example illustrates some Synpac commands. It also shows how a macro may be used to create a special purpose command.

Synpac was the first package that was implemented. It was based on the Fortran programs described in Aström (1963). A test version was made as an MS project. The current version is described in Wieslander (1980d). An introduction is presented in Gutman et al. (1984). A list of the commands in Synpac is given in Appendix D.

Polpac

Polpac is a polynomial oriented design package for multi-output single-input systems. It includes algorithms for pole placement, minimum variance control, and LQG control. The package allows classical design using root loci and Bode plots. Root loci may be drawn with respect to arbitrary parameters. A list of the commands in Polpac is given in Appendix E.

hierarchical system was therefore developed, see Elmqvist (1978, 1979a,b). Experimental software, which operates on the basic system description and generates simulation programs e.g. in Simnon, and linearized system equations have also been developed. We believe that this is an important step towards effective methods for dealing with large systems.

7. EXPERIENCES OF USING THE PACKAGES

The packages have been used at our department, at other universities, and in industry. The early use of the packages provided very good feedback to their development. There has been a continuous dialog between users and implementors at all stages of the development. Very valuable input was provided by visitors to the department and from industrial use of the packages. They often had different ideas on how to use the programs.

All staff members of the department and a large number of the students who have done MS or PhD dissertations have used the packages. The programs have been used in some of our advanced courses and in courses for industrial audiences. They are now being introduced also in elementary courses. The bottleneck for this has been the availability of a sufficient number of graphic terminals. By using the packages it has been possible to focus on concepts and ideas in the lectures and to work with realistic examples with considerable detail in exercises and projects.

The simulation language Simnon is used as a standard language for documenting models. The availability of a library of realistic models of different complexity is of course very beneficial in teaching. Simnon has been used in an interesting way in a recent book on computer control, Aström and Wittenmark (1984), which makes extensive use of simulation. All simulation results are implemented as Macros in Simnon which are accessable from the student terminals. This means that the students may conveniently check the results and also look into effects of variations of data. Simnon has also been used in many applied projects at the institute. A typical example is a study on modeling and simulation of a wind turbine, Bergman et al. (1983).

We have found Idpac to be a very good tool to teach system identification. It is possible for the students to gain a lot of experience by working with real data. Idpac has also been used in many industrial projects. Typical examples are given in Aström and Källström (1976) and Källström and Aström (1981). Trouble shooting in the paper industry has e.g. been another interesting application area, see Lundqvist and Nordström (1980) and Johansson et al. (1980).

Similarly we have found that Synpac is an excellent tool for teaching LQG design. The students can work with realistic problems with reasonable effort. Synpac has also been

The personal computers which are projected to appear within a few years have specifications like: a primary memory of 2 Mbytes, a secondary memory 100 Mbytes, a computing speed of one megaflop/s and a price less than 20k$, see Dertouzos and Moses (1980). These computers are also expected to have a high resolution bit mapped color graphics display. With computers like this it is possible to have single user work-stations with packages which are much more sophisticated than all our current packages. The existence of computers like Apollo, Lisa, Mackintosh, Sun, and Iris make the predictions quite credible.

There has been a drastic development of the computer output devices. A teletype is capable of writing at a speed of 10 ch/s (110 Baud). A regular terminal connected to a 19.2 kBaud channel can write a screen i.e. 80 x 24 ch in a second. A good vector graphics terminal can refresh up to 100 000 long vectors or a million short vectors per second. A high resolution bit mapped display may refresh 512 x 512 pixel frames at rates of 60 frames/s (15 Mbit/s).

The input devices have unfortunately not developed at the same rate. We still have ordinary keyboards, see Montgomery (1982). A very good typist may type at a rate of 8 ch/s. A normal engineer types considerably slower. Pointing devices like roll balls, mouses and touch panels have been invented. These devices may perhaps be used to increase the input rate indirectly by combining the rapid output rate with feedback via the picking device (dynamic menus). Speech input is another possibility. There are, however, no indications of a more drastic increase in the input rate.

The renaissance of graphics

Graphics has played a major role in engineering. The first books used in engineering education were books of drawings of machines by Leonardo da Vinci. Graphical representations have been used extensively ever since. Graphics in the forms of Bode diagrams, Nichols charts, root loci, block diagrams and signal flow diagrams are important tools in classical control theory. Modern control theory has, however, not been much influenced by graphics. This can partly be explained by lack of proper tools for graphics. The situation may change drastically in the future because good graphics hardware will be available at a reasonable cost.

The man-machine interface

A high bandwidth information transmission is required for an efficient man-machine communication. This implies a high rate of transmission of symbols and a high information content in each symbol. The user interfaces in our packages were designed for teletypes combined with graphic terminals having storage screens and data rates of 4800 Baud. These were the only tools available at reasonable cost when our design was frozen. A

code for realization of the control laws. Symbolic calculations were not used in our packages because of the limited computing facilities available. It is, however, feasible in future packages.

When transfering our packages we have noticed that their power increases considerably if an experienced user is around. The possibilities of providing the packages with a rule based expert system or an advisory system, Barr and Feigenbaum (1982), is therefore very appealing. It is an interesting research problem to find out if expert knowledge in identification, analysis and design of control systems can be incorporated in the packages.

Implementation languages

The source code for the smallest package described in this paper is about 30 000 lines of source code. A future package may be an order of magnitude larger. A good programming environment and efficient software tools are necessary to develop and maintain such systems. Fortran was used in our packages to make them portable. It is however unlikely that future systems will be written only in Fortran.

Fortran libraries like Eispack, Linpack, (hopefully also a control package), and some graphics package will probably be used. Although Intrac was written in Fortran it is not convenient to do so. Pascal would be much more convenient, particularly if we want to include formula manipulation and the other features that we may expect in a future system. An expression parser is needed. Macros and user defined procedures are very useful in order to increase the efficiency of the man-machine dialog. More flexible control structures and more powerful commands than those used in Intrac would be desirable. One possible extension is the system Delight which is based on the language RATTLE developed by Nye et al (1981). Other possibilities are to replace Intrac by languages with an interactive implementation like Apl, Lisp or Logo or an interpretive threaded language like Forth, see Winston and Horn (1981), Abelson, (1982) and Kogge, (1982). Systems based on Lisp will be extendable automatically, symbolic manipulations are also easy to implement. There are good programming environments for Lisp which have been used to implement very large systems. Natural language interfaces and expert systems are also often written in Lisp.

Programs like Idpac and Delight, which handle the interpretations of the commands and the man-machine interaction, have many features in common with operating systems like UNIX, Kernighan and Pike (1983). They may be viewed as an interpretative programming language whose data types are files. Such programs can be implemented as extensions to an operating system. The software Honey-X developed by Honeywell is such a system which is based on Multics, see Anon. (1982). The advantage of such an approach is that the major part of the code is the ordinary operating system. Only a small portion has to be

9. CONCLUSIONS

Interactive computing is a powerful tool for problem solving. An engineer can come to the work station with a problem and he can leave with a complete solution after a few hours. The results are well documented in terms of listings, text and graphs. The problem solver can obtain the solution by himself without relying on programmers as intermediaries. Our projects have shown that the productivity in analysing and designing control systems can be increased substantially by using these tools. We believe that interactive computer aided design tools is one possibility to make modern control theory cost effective.

Computer aided design of control systems is still in its infancy. A small number of systems have been implemented in a few places. There are many possible future developments which are mainly driven by the computer development. Packages of the type we have been experimenting with can easily be fitted into the personal computers or work stations that will be available in a few years time. The bit mapped high resolution color displays that will be available on these computers offer new possibilities for an efficient man-machine dialog. With the drastic increase in computer capacity, that is forth coming, it is also possible to make much more ambitious projects. Applications of computer aided design also appear in many other branches of engineering. Cross fertilization between the fields will most likely lead to a rapid development.

10. ACKNOWLEDGEMENTS

The work reported in this paper has been supported by the Swedish Board of Technical Development for many years. This support is gratefully acknowledged. The projects, which the paper draws upon, have been true team efforts. Many members of the department have contributed to discussions of command structures, implementation, testing and evaluation. Particular thanks are due to Johan Wieslander and Hilding Elmqvist, who generated many of the important ideas, and to Tommy Essebo and Thomas Schönthal, who did a major part of the programming of the packages.

Dertouzos, M L, and Moses, J (1980): The Computer Age: A twenty year view. MIT Press, Cambridge, Mass.

DOD (1983): Reference Manual for the Ada Programming Language. ANSI/MIL-STD-1815A--1983, United States Department of Defense, Washington, DC.

Dongarra, J J, Moler, C B, Bunch, J R, and Stewart, G W (1979): LINPACK - Users' guide. SIAM, Philadelphia.

Edgar, T F (1981): New results and the status of computer-aided process control system design in North America. Engineering Foundation Conference on Chemical Process Control-II, Sea Island, Georgia.

Edmunds, J M (1979): Cambridge linear analysis and design programs. IFAC Symposium on Computer Aided Design of Control Systems, Zurich, 253-258.

Elmqvist, H (1975): SIMNON, an interactive simulation program for nonlinear systems. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-7502).

Elmqvist, H, Tyssø, A, and Wieslander, J (1976): Scandinavian control library. Programming. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3139)/(1976).

Elmqvist, H (1977): SIMNON - An Interactive Simulation Program for Nonlinear Systems. Simulation '77, Montreux, Switzerland, June 1977.

Elmqvist, H (1978): A Structured Model Language for Large Continuous Systems. Ph.D. Thesis. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-1015)/1-226/(1978).

Elmqvist, H (1979a): Dymola - A Structured Model Language for Large Continuous Systems. Summer Computer Simulation Conference, Toronto, Canada, July 1979.

Elmqvist, H (1979b): Manipulation of Continuous Models Based on Equations to Assignment Statements. Simulation of Systems '79. Sorrento, Italy, September 1979.

Elmqvist, H (1982): A graphical approach to documentation and implementation of control systems. Proc 3rd IFAC/IFIP Symposium on Software for Computer Control, SOCOCO 82, Madrid, Spain.

Folkesson K, Elgcrona, P O, and Haglund, R (1980): Design and experience with a low-cost digital fly-by-wire system in the SAAB JA37 Viggen A/C. Proc 13th International Council of the Aeronautical Sciences, Seattle, WA.

Foley, J D, and van Dam, A (1983): Fundamentals of interactive computer graphics. Addison Wesley, Reading, Mass.

Frederick, D K (1982a): Computer packages for the simulation and design of control systems. Lecture notes, Arab school on science and technology.

Frederick, D K (1982b): Simnon reference manual. Automation and Control Laboratory, Corporate Research and Development, General Electric Company, Schenectedy, New York.

Furuta, K, and Kajiwara, H (1979): CAD system for control system design. J of the Society of Instrument and Control Engineers, Japan, 18 (9). (In Japanese).

Gale, W A, and Pregibon, D (1983): Using expert systems for developing statistical strategy. Proc Joint Statistical Meetings, Toronto.

for the computer-aided design of control systems. In Bensoussan and Lions (Eds.): Analysis and Optimization of Systems. Springer Lecture Notes in Control and Information Sciences, Springer, Berlin.

Lundqvist, S O, and Nordström, H (1980): The development of a control system for a pulp washing plant through the use of dynamic simulation. Preprints IFAC Conf. on Instrumentation and automation in the paper, rubber, plastics and polymerisation industries. Gent, Belgium.

Mansour, M editor (1979): Preprints first IFAC Symposium on CAD of Control systems, Zurich. Pergamon Press.

Moler, C B (1981): MATLAB user's guide. Report Department of Computer Science, University of New Mexico.

Montgomery, E B (1982): New keyboard concepts. IEEE Computer 15:3, 11-18.

Munro, N (1979): The UMIST control system design and synthesis suites. IFAC Symposium on Computer Aided Design of Control Systems, Zurich, 343-348.

Newman, W M, and Sproull, R F (1979): Principles of interactive computer graphics. McGraw-Hill, New York.

Nye, W, Polak, E, Sangiovanni-Vincentilli, A, and Tits, A (1981): An optimization-based computer-aided-design system. Proc ISCAS, April 24-27.

Perry, T, Truxal, C, and Wallich, P (1982): Video games: the electronic big bang. IEEE Spectrum 19:12, 20-33.

Polak, E (1981): Interactive software for computer-aided-design of control systems via optimization. Proc. 20th IEEE Conf. on Decision and Control, San Diego, CA, December 16-18, pp 408-412.

Rimvall, M, and Cellier, F (1984): IMPACT Interactive mathematical program for automatic control theory. In Bensoussan and Lions (Eds.): Analysis and Optimization of Systems. Springer Lecture Notes in Control and Information Sciences, Springer, Berlin.

Rosenbrock, H H (1974): Computer-aided control system design. Academic Press, New York.

Ryder, B G (1975): The Pfort verifier: User's guide. CS Tech. Rept. 12, Bell Labs.

Schank, R C (1975): Conceptual Information Processing. North Holland. Amsterdam.

Schank, R C, and Abelson, R P (1977): Scripts, plans, goals and understanding. Lawrence Erlbaum Associates, Hillsdale NJ.

Smith, B T, et al. (1976): Matrix eigensystem routines - Eispack guide. 2nd ed., Lecture Notes in Computer Science, Vol. 6, Springer-Verlag, New York.

Spang, H A, III, and Gerhart, L (Eds.) (1981): Preprints GE-RPI, Workshop on control design, Schenectady, N.Y.

Suleyman, C (1981): Interactive system for education and research in control system design. IEEE International Conference on Cybernetics and Society, Atlanta, Georgia.

Tyssø, A (1979): CYPROS: Cybernetic program packages. IFAC Symposium on Computer Aided Design of Control Systems, Zurich, 383-389.

Tyssø, A (1981): New results and the status of computer aided process control systems design in Europe. Engineering Foundation Conference on Chemical Process Control-II,

APPENDIX A - IDPAC COMMANDS

1. Utilities
CONV - Conversion of data to internal standard format
DELET - Delete a file
EDIT - Edit system description
FHEAD - Inspect and change file parameters
FORMAT - Conversion of data to symbolic external form
FTEST - Check existence of a file
LIST - List files
MOVE - Move data in database
TURN - Change program switches

2. Graphic output
BODE - Plot Bode diagrams
HCOPY - Make hard copy
PLMAG - Magnify plot and allow changes of data
PLOT - Plot curves with linear scales

3. Time series operations
ACOF - Compute autocorrelation function
CCOF - Compute cross correlation function
CONC - Concatenate time series
CUT - Extract a part of a time series
INSI - Generate time series
PICK - Pick equidistant time points
SCLOP - Do scalar operations on a time series
SLIDE - Introduce relative delays between time series
STAT - Compute statistical characteristics
TREND - Remove a trend
VECOP - Do vector operations on a time series

4. Frequency response operations
ASPEC - Compute an auto spectrum
CSPEC - Compute a cross spectrum
DFT - Discrete Fourier Transform
FROP - Operate on frequency responses
IDFT - Inverse Discrete Fourier Transform

5. Simulation and model analysis
DETER - Deterministic Simulation
DSIM - Simulation with noise
FILT - Compute a filter system
RANPA - Pick parameters from a random distribution
RESID - Compute residuals with statistical test
SPTRF - Compute the frequency response of a transfer function

6. Identification
LS - Least Squares identification
ML - Maximum Likelihood identification
SQR - Least Squares data reduction
STRUC - Least Squares structure definition

APPENDIX C - SIMNON COMMANDS

1. Utilities

EDIT	- Edit system description
GET	- Get parameters and initial values
LIST	- List files
PRINT	- Print files
SAVE	- Save parameter values and initial values in a file
STOP	- Stop

2. Graphic output

AREA	- Select window on screen
ASHOW	- Plot stored variables with automatic scaling
AXES	- Draw axes
HCOPY	- Make hard copy
SHOW	- Plot stored variables
SPLIT	- Split screen into windows
TEXT	- Transfer text string to graph

3. Simulation Commands

ALGOR	- Select integration algorithm
DISP	- Display parameters
ERROR	- Choose error bound for integration routine
INIT	- Change initial values of state variables
PAR	- Change parameters
PLOT	- Choose variables to be plotted
SIMU	- Simulate a system
STORE	- Choose variables to be stored
SYST	- Activate systems

APPENDIX E - COMMANDS IN POLPAC

1. Utilities

CONV	- Conversion of data to internal standard format
DELET	- Delete a file
EDOT	- Edit system description
FHEAD	- Inspect and change file parameters
FORMAT	- Conversion of data to symbolic external form
FTEST	- Check existence of a file
LIST	- List files
MOVE	- Move data in database
TURN	- Change program switches

2. Graphics

BODE	- Plot Bode diagrams
HCOPY	- Make hard copy
LOCPLOT	- Plot root locus diagrams
NIC	- Plot Nichols diagrams
NYQ	- Plot Nyquist diagrams
PLEV	- Plot eigenvalues and allow editing
PLOT	- Plot curves with linear scales

3. System and polynomial operations

INSI	- Generate a data file
POLOP	- Evaluate algebraic polynomial expressions
POLSYS	- Create a system file or a polynomial file
POLY	- Generate or edit a polynomial
POLZ	- Compute and plot the zeros of a polynomial
SIMU	- Simulate a system
SYSOP	- Build a system from subsystems

4. Analysis

PROP	- Compute bandwidth, rise time, error coefficients
ROTLOC	- Compute the root locus
ROUTH	- Compute and display Routh's tableau
TRFFR	- Compute frequency response of a transfer function
TRFSIM	- Simulate

5. Synthesis

DEADBE	- Compute dead-beat strategy
MIVRE	- Compute minimum variance control
POLPLA	- Make a pole placement design