Computer Aided Modeling, Analysis and Design of Control Systems

A Perspective

Åström, Karl Johan

CODEN:LUTFD2/(TFRT-7251)/1-35/(1983)

COMPUTER AIDED MODELING, ANALYSIS AND DESIGN OF CONTROL SYSTEMS. - A PERSPECTIVE -.

KARL JOHAN ÅSTRÖM

DEPARTMENT OF AUTOMATIC CONTROL
LUND INSTITUTE OF TECHNOLOGY

FEBRUARY 1983

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
Box 725
S 220 07 Lund 7 Sweden

Document name
REPORT

Date of issue
February 1983

Document number
CODEN:LUTFD2/(TFRT-7251)/1-35/(1983)

Author(s)
Karl Johan Åström

Supervisor

Sponsoring organization

Title and subtitle
Computer aided modeling, analysis and design of control systems. - A perspective -.

Abstract

The paper summarizes experiences of development and use of interactive software for computer aided design of control systems. Different principles for interaction with users having wide ranges of experiences and knowledge are discussed. A comprehensive set of packages for modeling, identification, analysis, simulation and design are described. Problems associated with structuring, probability, maintainability, and extensibility are discussed. Experiences from development and use of the packages in teaching and industrial environments are discussed. Some views on future development of CAD for control systems are also given.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
35

Recipient's notes

Security classification

Distribution: The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

CONTENTS:

COMPUTER AIDED MODELING, ANALYSIS AND DESIGN OF CONTROL SYSTEMS

– A PERSPECTIVE

K.J. Åström

Department of Automatic Control
Lund Institute of Technology
S-220 07 Lund, Sweden

Abstract

The paper summarizes experiences of development and use of interactive software for computer aided design of control systems. Different principles for interaction with users having wide ranges of experiences and knowledge are discussed. A comprehensive set of packages for modeling, identification, analysis, simulation and design are described. Problems associated with structuring, portability, maintainability, and extensibility are discussed. Experiences from development and use of the packages in teaching and industrial environments are discussed. Some views on future development of CAD for control systems are also given.

Expanded version of papers presented at the 18th and 20th IEEE Conferences on Decision and Control, San Diego, CA and the GE-RPI workshop, Schenectady, NY.

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 explore the possibilities of developing the proper tools for using control theory cost effectively. This paper summarizes results and experiences from these projects.

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 design of man-machine interfaces, for interactive use of the computer. Graphics was important for rapid man-machine communication.

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

2. THE PROJECTS

A brief overview of the projects which were carried out are given in this section.

Goals

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.

Background

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.

Stepwise refinement

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.

Constraints

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 300 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.

Results

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 environments. Ideas on the use of graphics and interactive computing in future systems have also been developed. An overview of the results are given in the next sections.

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 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 gradually shift the initiative from the computer to the user as he becomes more proficient.

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 possible to create new user defined commands easily. It was thus possible to use the packages in ways which were not anticipated when they were designed. The decision to use commands instead of a question and answer dialog thus had far reaching consequences. A more detailed discussion of the different types of dialogs and of our experiences of them is given in Wieslander (1979). 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 (1981) although their conclusions are based on different hardware.

<u>Examples of commands</u>

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
   >EXIT
```

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 <u>subcommand</u> 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 CLSYS ← LOSS SYS

computes the optimal feedback gain L and the corresponding closed loop system CLSYS 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. It may also be useful to have a simple way of renaming the commands. We have experimented with short form commands and renaming mechanisms. These functions are, however, not implemented in our standard packages.

A similar mechanism may be used for commands which use 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

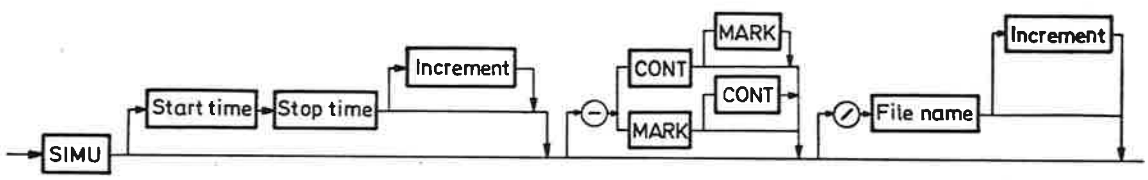

Fig. 1 Syntax diagram for the command SIMU.

```
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.

<u>Macros</u>

The 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 in storage 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.

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. By having commands for reading the keyboard and for writing on the terminal it is also possible to implement menu driven dialogs using macros.

An interactive CAD program based on a command dialog with a macro-facility may be viewed as an <u>extendable high level problem solving language</u>.

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 all 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 command decoding, file handling and plotting. Intrac also contains the macro facility. 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.

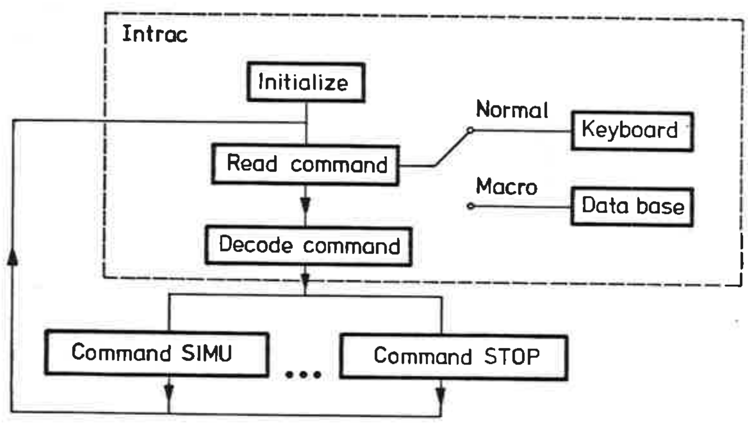

Fig. 2 Skeleton flow chart for a command driven program with a macro facility.

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.

How to choose commands?

The selection of commands is one of the major issues when designing a CAD package. The commands determine how useful a package is and how easy it is to learn. It is important that commands are complete in the sense that they allow use of a wide range of techniques in an area. Otherwise the designer will only try those approaches for which commands are available. Commands should also have a considerable expression power so that a control system designer can do what he wants with a few commands. The commands should also reflect the natural concepts from a theoretical point of view. This would make it easy for a user well versed in control theory to use a package. The commands should also be few and simple so that they are easy to learn and remember. This is of course in conflict with requirements on completeness and expression power. Selection of commands is thus a good exercise in engineering design.

Based on experiences from our projects we have arrived at some design principles. A set of basic commands which correspond to the elements of the theory and which allow coverage of a certain problem area are first determined. Simplifications and extensions are then generated using the macro facility.

4. PROGRAM PACKAGES

The initial experiments indicated that interactive program packages could be powerful problem solving tools. The necessity of considering a wide range of problems when developing the tools was also apparent. If this is not done it is easy to arrive at specialized solutions which are difficult to generalize and extend. To work with programs of reasonable size a family of interactive program packages for modeling, identification, simulation, analysis and design of control systems were developed. The packages are all based on the common user interface Intrac which was discussed in Section 3. The different packages are listed in Table 1 which also summarizes some data about them. Brief descriptions of the different packages are given below. The commands used in the packages are listed in the appendices. There are also a large number of macros available for all packages.

Table_1 – Examples of program sizes

	Number of commands	Source code lines	Program size kbytes
Intrac	17	7 000	90
Idpac	39	37 000	470
Modpac	37	41 000	570
Simnon	24	25 000	360
Synpac	46	43 000	630
Polpac	32	32 000	460

Idpac

Idpac is a package for data analysis and identification of linear systems having one output and many inputs. Time series analysis of ARMA and ARIMA models is a special case. The package has commands for manipulation and plotting of data, correlation analysis, spectral analysis and parametric system identification. There are also commands for model validation and simulation. The basic techniques used for parameter estimation are the least squares method and the maximum likelihood method. By using the macro facility it is however possible to generate commands for most of the parameter estimation methods which are proposed in literature. It was actually in the development of Idpac that the power of the macro concept became apparent. In the early Idpac versions there were many commands necessary to cover the available identifications 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 Åström et al (1965). The latest version is described in Wieslander (1980b). The paper Åström (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 is 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. This makes Simnon well suited for simulation of digital control systems. 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.

The following annotated dialog illustrates how Simnon is used.

```
CONTINUOUS SYSTEM PROC
"Integrator with input saturation
Input u
Output y
State x
Der dx
upr=if u<-0.1 then -0.1 else if u<0.1 then u else 0.1
dx=upr
END


DISCRETE SYSTEM REG
"PI regulator with anti-windup
Input yr y
Output u
State i
New ni
Time t
Tsamp ts
e=yr-y
v=k*e+i
u=if v<ulow then ulow else if v<uhigh then v else uhigh
ni=i+k*h*e/ti+u-v
ts=t+h
k:1
ti:1
h:0.5
ulow:-1
uhigh:1
END


CONNECTING SYSTEM CON
"Connecting system for simulation of process PROC
"with PI regulation by system REG
yr[REG]=1
y[REG]=y[PROC]
u[PROC]=u[REG]
END
```

Listing 1 - Simnon description of a simple control loop consisting of a continuous time process and a discrete PI regulator.

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	Draw y with automatic scaling and yr with the
SHOW yr	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. 3. 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. 3. These values are so large that the integral is never reset. The simulation, shows in thin lines in Fig. 3,

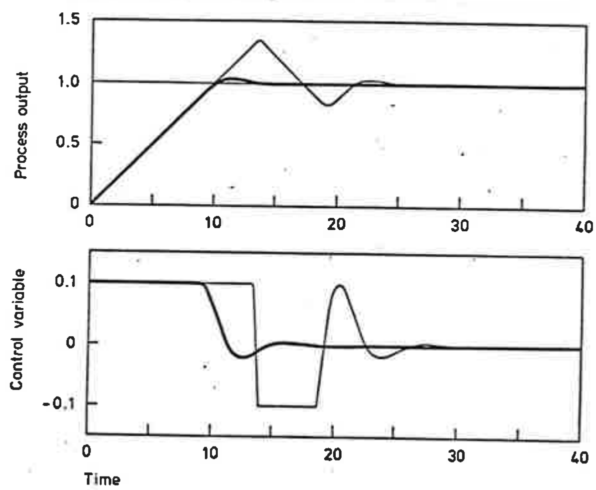

Fig. 3 Results of simulation of process with PI regulator. Thin lines show results with ordinary regulator and thick lines show results for regulator with anti-windup.

thus correspond to a regulator without wind-up. The actual actuator limitations correspond to ulow=-0.1 and uhigh=0.1. The commands

```
PAR ulow:-0.1
PAR uhigh:0.1
```

change the parameters and the command SIMU now generates the curves shown in thick lines in Fig. 3. Notice the drastic improvements due to the nonlinearity in the regulator.

The first version of Simnon was implemented in an MS project. Simnon has gone through several stages of development. See Elmqvist (1975) and (1977). A list of the commands in Simnon is given in Appendix C.

<u>Synpac</u>

Synpac is a state space oriented design package. It includes facilities for calculating state feedback and Kalman filters for continuous and discrete time LQG problems. It also has facilities for transforming continuous time problems into discrete time problems.

An example illustrates some features of Synpac. Consider the standard LQG problem

$$dx = Axdt + Budt + dv$$
$$dy = Cxdt + de$$

where {v} and {e} are Wiener processes with joint incremental covariances

$$\operatorname{cov}\begin{bmatrix} dv \\ de \end{bmatrix} \begin{bmatrix} dv \\ de \end{bmatrix}^T = \begin{bmatrix} R_1 & R_{12} \\ R_{12}^T & R_2 \end{bmatrix}.$$

Let the control problem be to minimize

$$J = \lim_{T\to\infty} \frac{1}{T} E \int_0^T \left[x^T(t)Q_1x(t) + 2x^T(t)Q_{12}u(t) + u^T(t)Q_2u(t) \right] dt.$$

Furthermore assume that a digital regulator will be used and that sampling periods from 0.5 to 5 s are of interest.

Assume that a system description which contains the matrices A, B, C, R_1, R_{12}, R_2, Q_1, Q_{12} and Q_2 has been introduced in a file called CSYS, and that the design parameter i.e. the parameter which will be modified in the design, is the 3,3 element of the matrix Q_1. The following macro then executes the design

```
 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 Åström (1963) A test version was made as an MS project. The current version is described in Wieslander (1980d). 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.

Portability

The programs were initially written in FORTRAN for a minicomputer PDP-15. A considerable effort was also devoted to development of subroutine libraries and programming standards. See Elmqvist et al (1976), Wieslander (1977) and Cowell (1977). The advantage of using a large main frame computer for program development was soon apparent. The program development was therefore moved to a Univac 1108 at the Lund University Computing Center. More powerful program development tools like Pfort could then be used. See Ryder (1975). Since there was a considerable interest from external groups to use the programs, a substantial effort went into making the software portable This included development of Fortran routines for file, character and string handling. A plotting library in Fortran was also developed. These routines are interfaced with a well-defined small set of installation dependent routines. A result of the efforts is that the packages are indeed portable. Packages are currently running on the following computers: PDP-15, PDP-11, DEC-10, VAX-11/780, NOVA-3, Nord-100, ECLIPSE, IBM-1800, IBM-360, CDC-1700, CDC-6400, HP-3000, Honeywell, SEL-32, Univac 1108, PRIME-750.

5. LARGE SYSTEMS

When working with the projects it was found that there were certain problems where interactive computing is not feasible. These problems typically involve large systems where it can easily happen that the computing time required is so large that it does not make sense to wait for the results at the terminal. We experienced this in connection with identification and simulation of large systems.

Lispid

For identification of large systems we found that it was better to use a batch program which allows an interactive start up and an interactive inspection of the results. LISPID is an example of such a program. This program allows estimation of parameters in linear stochastic systems with arbitrary parameterization and in special types of nonlinear systems. See Källström et al (1976).

Dymola

Another problem was also encountered in connection with modeling of large systems. It is straightforward to write down and check the balance equations. It is however a major effort to reduce the equations obtained to forms that are suitable for simulation and control design. The language Dymola which admits a simple description of a large hierarchical system was therefore developed. See Elmqvist (1978), (1979a) and (1979b). 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.

6. 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. 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 and 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 forthcoming book on computer control, Åström 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 (1981).

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 Åström and Källström (1976) and Källström and Åström (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 used to design control laws for digital flight control systems. These control laws have also been flight tested. See Åström and Elgcrona (1976) and Folkesson et al (1982).

The programs are used by a number of industries in Sweden and to a limited extent outside Sweden. To spread knowledge about the packages we have given a number of courses on the use of interactive computing. It has been our experiences that the average engineer can use of the tools quite well after a one week course. The macro facility is very useful because it makes it possible to tailor a few special commands for the standard needs of each user.

7. FUTURE WORK

Computer aided design of control systems is still in its early stages. There are a number of packages like ours. An overview of some packages are found in Atherton (1981), Edgar (1981), Edmunds (1979), Frederick (1982), Hashimoto and Takamatsu (1981), Lemmens and van den Boom (1979), Munro (1979), Rosenbrock (1974), Tyssö (1981) and Wieslander (1979b). More references are also found in these papers. Special workshops and symposia devoted to CAD for control systems have been organized by IFAC, GE-RPI, and IEEE CSS. See Mansour (1979), Leininger (1982), Spang and Gerhart, (1981), Herget and Polak, (1982). Computer aided tools are also popular in many other fields e.g. mechanical design and VLSI design. The seminal work on computer graphics by Newman and Sproull (1979) and the text Foley and van Dam (1982) contain much material and many rererences.

The field is in a state of rapid development due to an increased understanding of the technology and the drastic development of computer and graphics hardware. It is safe to predict that future computer aided design tools will be much more powerful than the packages described in this paper. Some speculations on future development are given in this section.

Computer hardware

The virtual memory requirement for each package is less than 1 Mbyte. A package can run on a computer having a primary storage of 128 kbytes. They require fast floating point operations for reasonable efficiency. The packages could be implemented on a computer like the IBM personal computer with an Intel 8087 floating point processor. A good Fortran compiler is required to do this with a reasonable effort.

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, PERQ and Sun 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 storage scope is very limited. Curves may be shown but they can not be erased individually. Bit mapped graphics is faster and much more flexible. Individual picture elements may be changed instantaneously. This makes it possible to zoom, scroll and pan a picture. Color and animation add extra dimensions. Imaginative use of color graphics is still in its infancy in CAD packages for control systems. Interesting ideas have been proposed by Polak (1982) in connection with applications of optimization techniques. Animation has not been used much. It is clear that we have a lot to learn from designers of video games. See Perry et al (1982).

The information content of each symbol is related to the expression power of the commands. Our experience indicate clearly the need for having a CAD language with considerable expression power. It would be nice to describe all operations using notations similar to those used in system theory. An expression parser is needed. Macros are very useful 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).

Short form commands and default values are two simple techniques for increasing the efficiency of the man-machine dialog. More sophisticated techniques like conceptual dependency are found in semantically oriented programs for processing natural language. See Schank (1975) and Schank and Abelson (1977). It would be interesting to explore how these ideas can be incorporated in CAD systems. Another possibility is to have an operator communication which is more oriented towards graphics. Interesting ideas in this direction are demonstrated in Elmqvist (1982).

Numerical algorithms and design tools

The numerical algorithms for the design primitives are key elements in the software. There have been major advances in numerical software for linear algebra over the past years. A substantial effort has gone into subroutine packages such as Eispack, Garbow et al (1977) and Smith et al (1976), and Linpack, Dongarra et al (1979), which are now available in the public domain. A similar effort has not yet been devoted to the numerical calculations required for analysis and design of control systems, although libraries are maintained at many departments. The numerical problems that arise in automatic control are however starting to receive attention from numerical analysts. See van Doren (1981), Hammarling (1982) and Laub, (1980). This is crucial for the future development.

Most data processing in current packages is inspired from numerical analysis. The powers of non-numeric data processing have not been exploited. It would be highly desirable to have facilities for symbolic manipulation. This can e.g. be used for model simplification, generation of code for computing equilibrium points, generation of simulation code, linearization, etc. If symbolic manipulations are included it is also possible to generate 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. It is thus likely that future systems will be implemented using several programming languages. One indication of this is the design package ISER-CSD which is written in Fortran and Pascal. See Suleyman (1980). This system is, however, restricted to one computer system.

Another possibility is to base the interaction on an existing language with an interactive implementation like Apl, Lisp or Logo. 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.

The programming language Ada, Dod (1980), which will be available in a few years time is another interesting alternative. The basic subroutine libraries can conveniently be implemented as packages in Ada. A wide range of libraries can be expected to be available for Ada. Since Ada supports the concept of tasks it will also be possible to apply ideas from concurrent programming. A good programming environment which will be a substantial help in software development is also planned for Ada. The deciding factor will probably depend on how well Ada will be accepted.

Some computations, such as simulation and identification, are quite demanding computationally. The problem solving would be more efficient and convenient if the the user could perform several tasks like plotting, editing and report writing in parallel. This mode of operation is particularly useful for a system with windowing. See Goldberg et al (1983).

Implementation tools

Our packages were developed from scratch. Future packages may be expected to use ready made modules to a much larger extent.

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 (1980) and one of its extension $Matrix_X$, Walker et al (1982). It is, however, clear, that it is very useful to also have polynomials, rational functions and good general structures for linear and nonlinear systems. In some cases it is also valuable to describe systems as hierarchical interconnections of subsystems.

For simple systems with only one data type, like matrices, all data may be stored in a stack or in a simple array. In our packages which used more sophisticated data structures the data was stored in files. Our experiences indicate that it would be very useful to have a more flexible system. It is probably a very 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.

In our packages we had to develop our own graphics interface. A few simple routines which were compatible with Tektronix 4010 systems were used. The situation will be much better when standards like the Graphical Kernel System (GKS) or raster graphics extensions of SIGGRAPH Core materialize. See Foley and van Dam (1982) and Anon (1982).

8. 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.

9. 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.

10. REFERENCES

Abelson, H (1982): Logo for the Apple II. Byte/McGraw-Hill Peterborough, New Hampshire.

Anonymous (1982): Graphical Kernel System (GKS) - Functional Description. Draft International Standard ISO/DIS 7942 Version 7.02, August 9, 1982. Available through American National Standards Institute Inc. New York, N.Y.

Armstrong, E S (1980): ORACLS - A design system for linear multivariable control. Marcel Dekker, New York.

Åström, K J (1963): On the choice of sampling rates in optimal linear systems. IBM Research Report RJ-243.

Åström, K J, Bohlin, T and Wensmark, S (1965): Automatic construction of linear stochastic dynamic models for stationary industrial processes with random disturbances using operating records. Report TP 18.150, IBM Nordic Laboratory, Sweden.

Åström, K J and Källström, C G (1976): Identification of ship steering dynamics. Automatica 12, 9-22.

Åström, K J (1979): Reflections on theory and practice of automatic control. Dept of Autmatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-7178)/1-26/(1979).

Åström, K J (1980): Maximum likelihood and prediction error methods. Automatica 16, 551-574.

Åström, K J and Elgcrona, P O (1976): Use of LQG theory and Synpac in design of flight control systems. Reports SAAB, Linköping, Sweden.

Åström, K J and Wieslander, J (1981): Computer aided design of control systems - Final Report STU Projects 73-3553, 75-2776 and 77-3548. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3160)/1-23/(1981).

Åström, K J and Wittenmark, B (1984): Computer Control Theory. Prentice Hall, Englewood Cliffs, N J. (to appear).

Atherton, D P (1981): The role of CAD in education and research. IFAC Congress VIII, Kyoto, Japan.

Barr, A and Feigenbaum, E A (1982): The Handbook of Artificial Intelligence. Vol II. W. Kaufmann Inc. Los Altos, Calif.

Bergman, S Mattson, S E and Östberg A B (1981): A modular simulation model for a wind turbine system. AIAA-81-2558. Second AIAA Terrestrial Energy Systems Conference, Colorado Springs. To appear in AIAA Journal.

Cowell, W (Ed)(1977): Portability of numerical software. Lect. Notes in Comp. Sci., Vol. 57, Springer-Verlag, New York.

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

Dod (1980): Reference Manual for the Ada Programming Language. Defense Advanced Research Projects Agency. United States Department of Defense.

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, Tyssö, 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 Compurter 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 (1982): Fundamentals of interactive computer graphics. Addison Wesley, Reading, Mass.

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

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).

Garbow, B S., et al (1977): Matrix eigensystem routines - Eispack Guide Extension. Lect. Notes in Comp. Sci., Vol. 51, Springer-Verlag, New York.

Goldberg, A J Robson, D and Ingalls, D H H (1983): Smalltalk-80: The Language and its Implementation. Addison-Wesley, Reading, MA.

Gustavsson, I (1979): Processidentifiering - overheadbilder (Process identification - transparencies). Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, CODEN: LUTFD2/(TFRT-7166)/1-248/(1979).

Gustavsson, I and Nilsson, A-B (1979): Övningar för Idpac (Exercises for Idpac). Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-7169)/1-55/(1979).

Hammarling, S (1982): Some notes on the use of orthogonal similarity transformations in control. NPL Report DITC.

Hashimoto, I and Takamatsu Y (1981): New results and the status of computer aided process control systems design in Japan. Engineering Foundation Conference on Chemical Process Control-II, Sea Island, Georgia.

Herget, C J and Laub, A J ed (1982): Proc IEEE CSS Workshop on Computer Aided Control System Design. Berkeley, Calif. IEEE CSM 2:4. Special Issue on Computer-Aided Design of Control Systems.

Johansson, B L, Karlsson, H and Ljung, E (1980): Experiences with computer control based on optical sensors for pulp quality of a two state TMP plant. Preprints Process Control Conf. Canadian Pulp and paper Ass. Halifax, Canada.

Kogge, P M (1982): An Architectural trail to threaded-code systems. Computer 15:3 22-32.

Källström, C G, Essebo, T and Åström, K J (1976): A computer program for maximum likelihood identification of linear multivariable stochastic systems. Proc 4th IFAC Symposium on Identification and System Parameter Estimation, Tblisi, USSR.

Källström, C G and Åström, K J (1981): Experiences of system identification applied to ship steering. Automatica 17, 187-198.

Laub, A J (1980): Survey of computational methods in control theory. In Erisman, A M, et al (Eds.), Electric power problems. The mathematical challenge, SIAM, Philadelphia, pp 231-260.

Leininger, G (ed) (1982): Computer aided design of multivariable technological systems. Preprints second

IFAC symposium on Computer Aided Design of Multivariable Technological systems. West Lafayette, Indiana, USA.

Lemmens, W J M and Van den Boom, A J W (1979): Interactive computer programs for education and research: a survey. Automatica, 15, 113-121.

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.

Moler, C (1980): Matlab users' guide. Report Department of Computer Science, University of New Mexico.

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): Optimization-based computer-aided-design of control systems. Proc JACC. University of Virginia.

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., Lect. Notes in Comp. Sci., 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.

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

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

Van Doren, P (1981):A generalized eigenvalue approach for solving Riccati equations. SIAM J Sci. Stat. Comput. 2121-135.

Walker, R, Gregory, C and Shah, S (1982): $Matrix_X$ A data analysis, system identification, control design and simulation package. IEEE CSM 2:4 30-37.

Wegner, P (1968): Programming Languages, Information Structures, and Machine organization. McGraw-Hill, New York.

Wieslander, J (1973): Computer aided control system design. IEE, Publ. no 96.

Wieslander, J (1977): Scandinavian control library. A subroutine library in the field of automatic control. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3146)/1-38/(1977).

Wieslander, J (1979a): Interaction in computer aided analysis and design of control systems. PhD thesis, Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-1019)/1-222/(1979).

Wieslander, J (1979b): Design principles for computer aided design software. Preprints, IFAC Symposium on CAD of Control Systems, Zurich, 493.

Wieslander, J (1980a): Interactive programs - General guide. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3156)/1-30/(1980).

Wieslander, J (1980b): IDPAC commands - User's guide. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3157)/1-108/(1980).

Wieslander, J (1980c): MODPAC commands - User's guide. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3158)/1-81/(1980).

Wieslander, J (1980d): Synpac commands - User's guide. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3159)/1-130/(1980).

Wieslander, J and Elmqvist, H (1978): INTRAC, A communication module for interactive programs. Language manual. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3149)/1-60/(1978).

Wilkinson, J H and Reinsch, C (1971): Linear algebra. Springer-Verlag, Berlin.

Winston, P H and Horn, B K P (1981): Lisp. Addison-Wesley, Reading, Mass.

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
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 tests
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 B - Modpac commands

1. Utilities

AGR - Edit an aggregate file
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
NIC - Display a frequency response in a Nichols diagram
NYQ - Display a frequency response in a Nyquist diagram
PLEV - Display eigenvalues etc in the complex plane
PLOT - Plot curves with linear scales

3. Matrix operations

ALTER - Alter elements in a matrix
EIGEN - Compute eigenvalues of a matrix
ENTER - Enter a matrix element by element
EXPAN - Generate a matrix from sub-matrices
MATOP - Perform matrix operations
REDUC - Extract a submatrix
UNITM - Generate a unit matrix
ZEROM - Generate a zero matrix

4. Polynomial operations

POCONV - Polynomial image - polynomial file conversion
POLY - Generate or edit a polynomial
POLZ - Compute and plot the zeroes of a polynomial
ZERPOL - Create a polynomial from its zeroes

5. System operations

CONT - Convert to continuous time form
KALD - Do a Kalman decomposition
SAMP - Convert to discrete time form
SPSS - Compute the frequency response
SSTRF1 - Convert from state space to transfer function
SYST - Generate a system description
SYSTR - Do a general coordinate transformation
TBALAN - Transform to balanced form
TCON - Transform to controllable form
TDIAG - Transform to diagonal form
THESS - Transform to Hessenberg form
TOBS - Transform to observable form
TRFSS1 - Convert from transfer function to state space

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 D - Synpac commands

1. Utilities

```
CONV   - Conversion of data to internal standard format
DELET  - Edit system description
EDIT   - Symbolic text editor
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
NIC    - Display a frequency response in a Nichols diagram
NYQ    - Display a frequency response in a Nyquist diagram
PLEV   - Display eigenvalues etc in the complex plane
PLOT   - Plot curves with linear scales
```

3. Time series operations

```
CONC   - Concatenate two time series
CORNO  - Generate a correlated noise 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
STAT   - Compute
VECOP  - Do vector operations on a time series
```

4. Matrix operations

```
ALTER  - Alter elements in a matrix
EIGEN  - Compute eigenvalues of a matrix
ENTER  - Enter a matrix element by element
EXPAN  - Generate a matrix from sub-matrices
MATOP  - Perform matrix operations
REDUC  - Extract a submatrix
UNITM  - Generate a unit matrix
ZEROM  - Generate a zero matrix
```

5. System conversion and analysis

```
CONT   - Convert to continuous time form
POLES  - Compute the poles of a system
SAMP   - Convert to discrete time form
SIMU   - Simulate the time response of a system
SPSS   - Compute the frequency response of a system
SYSOP  - Generate a system from its subsystems
SYST   - Generate a system description
TRANS  - Convert a criterion from continuous time
         to discrete time form
```

6. Design

```
FEEDF  - Design feedforward control
```

KALFI - Compute a Kalman filter gain
LUEN - Compute a Luenberg observer
OPTFB - Compute a linear quadratic state feedback
PENLT - Reduce a penalty function to standard form
PPLAC - Pole placement for single input systems
RECON - State reconstruction for single input systems
REDFB - Compute an output feedback

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 - Dead-beat strategy
MIVRE - Minimum variance control
POLPLA - Polynomial synthesis

APPENDIX F – Intrac commands

1. Input and output

READ - Read string or variable from keyboard
SWITCH - Utility command
WRITE - Write string or variable on terminal

2. Assignment

DEFAULT - Assign default values
FREE - Release assigned global variables
LET - Assignment of variables and global parameters
STOP - Stop execution and return to OS

3. Control of program flow

FOR..TO - Loop
NEXT V
LABEL L - Declaration of label
GOTO L - Transfer control
IF..GOTO - Transfer control

4. Macro

END - End of macro definition
FORMAL - Declaration of formal arguments
MACRO - Macro definition
RESUME - Resume execution of macro
SUSPEND - Suspend execution of macro