High-Level Problem Solving Languages for Computer Aided Control Engineering

Åström, Karl Johan; Mattsson, Sven Erik

1987

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

CODEN: LUTFD2/(TFRT-3187)/1-020/(1987)

High-Level Problem Solving Languages for Computer Aided Control Engineering

Karl Johan Åström
Sven Erik Mattsson

STU project 85-4808

Department of Automatic Control
Lund Institute of Technology
March 1987

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

Document name
Report

Date of issue
1987-03-31

Document Number
CODEN:LUTFD2/(TFRT-3187)/1-020/(1987)

Author(s)
Karl Johan Åström
Sven Erik Mattsson

Supervisor

Sponsoring organisation
The National Swedish Board of Technical Development (STU contract 85-4808)

Title and subtitle
High-Level Problem Solving Languages for Computer Aided Control Engineering

Abstract

The purpose of this work has been to view CACE packages as high level problem solving languages. This approach gives a good unified way to view and analyse different packages. By collaborating with other groups we have been able to make a reasonable coverage. The notion of system is fundamental in control engineering and the representation of systems in CACE. The study showed that system representations are poorly delt with in existing CACE systems. A special study was therefore made of this problem. Different ways to describe systems have been investigated in the study. It has been shown that interconnected systems are conveniently described using object-oriented programming. A small prototype to test the ideas has also been implemented and tested.

Key words
Computer Aided Control Engineering; Computer Aided Control System Design; Software; System Representations

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
20

Recipient's notes

Security classification

Contents

1. Introduction

One of the most important conclusions from earlier projects is that it is fruitful to view a system for Computer Aided Control Engineering (CACE) as a high level problem solving language. The original plans for the project "High Level Problem Solving Languages for CACE" (STU project 85-4808) was to investigate existing CACE system from this point of view. In this project we could benefit from the collaboration with the Science and Engineering Research Council (SERC) in United Kingdom. Many design tools were reviewed by the UMIST group as part of their work. By participating in this work we got a good overview of the features of existing packages viewed as high level problem solving languages. We also did some experiments and efficiency studies which were fed into the work. A summary of the results is given in Munro et al (1986). System representations emerged as a key issue from these studies. The reason for this is that the notion of systems is an essential factor in all our work. Therefore we concentrated our efforts on system descriptions. In this work we could also draw on the previous experiences from Simnon (Elmqvist, 1975) and Dymola (Elmqvist, 1978). Åström developed a notion which allowed a generalization of Simnons system notion to cover hierarchically connected subsystems. This notion can also be extended to cover noncausal systems.

Systems can be represented in many different ways. There are graphical representations like block diagrams, signal flow diagrams and bond graphs. There are also mathematical representations like state space models and input-output relations which come in many different forms, matrix fractions, impulse responses, frequency responses. When working with control systems it is frequently useful to use several different representations of a system.

Only fairly primitive ways of representing systems are used in current CACSD systems. Typical examples are the Matlab derivatives where systems are described by matrices. A slightly more sophisticated representation is used in the simulation language Simnon. This representation recognizes that a system has the properties, inputs, outputs and states. Simnon also allows a system to be described as an interconnection of subsystems. However only flat interconnections are allowed.

A more flexible way to describe systems can be based on object-oriented programming. A general structural description of hierarchically connected systems can be constructed from simple ingredients by making a system an object with the properties Name, Inputs, Outputs, Subsystems and Connections.

It is also necessary to add behavioural descriptions to obtain a useful tool. This is done by creating new objects which describe behaviour. A system can then inherit both structure and behaviour. Behaviour can be characterized in many different ways. A state description is one of the simpler alternatives. This can be covered by introducing the object StateBehaviour with the properties States, StateTransitionMap and OutputMap. The behavioural descriptions should also allow a given system to be described by models of different complexity. Apart from the detailed quantitative descriptions it is also useful to be able to deal with qualitative descriptions of behaviour.

To implement these notions we had good input from a visiting scientist Dr Wolfgang Kreutzer. A simple prototype was implemented in ExperLisp

on a Macintosh. It is described in Åström and Kreutzer (1986). The prototype admits hierarchical system representation and symbolic manipulation of the system descriptions. The experiments with the prototype indicate that the approach is one way towards implementation of powerful CACE systems. This work also clearly indicated that object oriented programming was a very natural tool for this type of work. Some exploratory studies were also carried out using a SmallTalk implementation on the Macintosh.

The report is organized as follows. An assessment of some existing CACSD packages are given in Section 2. Requirements on system representations are given in Section 3. Sections 4 and 5 deal with representation of system structure and behavior. Some conclusions are drawn in Section 6.

2. Assessment of Some Existing Packages

A number of different design packages have been investigated. In this work we have drawn from the experiences of several other groups. We have had a direct collaboration with professor Neil Munros team at UMIST. In the development of the ECSTASY project they have compared a number of different software systems. We have also learned much by being a test site for the EAGLES project at the Lawrence Livermore National Laboratory (LLNL), California. Many ideas have also been exchanged with the group at ETH in Zürich and with the developers of CTRL-C, MATRIX$_X$ and PC-Matlab.

In the collaboration with the UMIST team we have looked into the following packages CTRL-C (Systems Control Technology, 1984), MATRIX$_X$ (Integrated Systems, Inc., 1984; Shah et al, 1985), ACSL (ACSL, 1986), DELIGHT (Polak et al, 1982), TSIM (Cambridge Control Limited, 1983), CLADP (Maciejowski and MacFarlane, 1982), CSS (Munro and Bowland, 1984), and the Lund packages (Åström 1983, 1985). The features examined include user interfaces, infrastructure, and tools. A summary of the findings are given in Munro et al (1986). The results have also been fed into the specifications of the ECSTASY project (SERC, 1986) which aims at a short range design. An assessment of existing packages is also given by Cellier and Rimvall (1986).

One thing that emerged from the studies was a need to look deeper into the representation of dynamical systems. This has also been reinforced by interaction with the vendors of CTRL-C and MATRIX$_X$. They are now trying to introduce graphics front ends but they have to do a lot of patching because they lack a good systems concept. The same is true for the developers of simulation software. See Cellier and Rimvall (1986). We therefore decided that it would be a good complement for us to take a deeper look at the representation of systems.

Section 2.1 contains a list of packages which are available at the department. The data structures available in the packages are indicated. Section 2.2 takes a closer look a the system representation concepts of some packages.

2.1 Available Packages

In our laboratory we have access to a number of software packages. First we have those developed at the department (Åström, 1983, 1985):

Simnon — *Simulation program for Nonlinear systems.* Interactive simulation program for nonlinear continuous and discrete time systems with facilities for optimization and use of experimental data. Simnon allows a system to be described as a flat interconnection of subsystems. There are two types of subsystems: continuous and discrete time. See Elmqvist (1975) and Elmqvist, Åström and Schönthal (1986).

Idpac — *Identification Package.* Interactive program for data analysis and identification of linear systems using parametric

and non-parametric (covariance spectra) methods. Has a data structure for ARMAX models. Signals, frequency responses, spectra etc. are treated as vectors and matrices. See Wieslander (1980a).

Synpac — *Synthesis Package.* Interactive program for design of feedback and feedforward controllers for multivariable linear continuous or discrete systems; Pole placement by state-feedback, LQG, KBF. Has a data structure for systems on state space form with covariance matrix descriptions of white noise inputs. The description of a system may contain multiple representations. Signals, frequency responses etc. are treated as vectors and matrices. See Wieslander (1980c).

Modpac — Analysis and transformations of models. See Wieslander (1980b).

Polpac — Polynomial oriented analysis and design package.

Lispid — System for off-line identification of nonlinear continuous time systems.

Dymola — A new simulator concepts with hierarchical structures and model types. See Elmqvist (1978).

We have also access to software developed outside the department. Besides the references given a general reference is ELCS (1987). ELCS stands for the Extended List of Control Software and is a collection of one-page summaries on computer-aided control engineering packages and related software libraries. The summaries are written by the software developers. ELCS contains more than 80 software descriptions.

ARGOS — *A Real-time Graphical Operating System.* Has facilities to create, display, control, manipulate and edit graphical structures. Tools for defining linear systems by graphical sketching of block diagrams have been implemented using ARGOS' facilities. See King and Gray (1985).

Blaise — General purpose command driven CACSD package which supports several data structures as real, complex and polynomial matrices, lists and macros. See Delebecque and Steer (1985) and Delebecque (1986).

CAMP — *Camp Aided Modeling Program.* A pre-processor that derives nonlinear state equations from bond graphs, block diagrams and their combination. See Granda (1983).

CES — *Swansea Control Engineering Station.* Supports creation and editing of block diagrams and signal flow diagrams in a Macintoshlike style. The concepts for describing behaviour are under design and focused on linear systems. See Barker et al (1986, 1987).

CC — *Complete Control.* Analysis and design of linear control systems of the following types: classical, sampled data, multirate, state-space, multivariable, and optimal (LQR, KBF, LQG). Has data structures for transfer functions, transfer function matrices, state space systems, real matrices, simulation output, and frequency files. See Thompsson

	(1984, 1985). This package can also communicate with PC-Simnon.
CSS	*UMIST's Control System Software.* On-line analysis, design and simulation of linear (either continuous or sampled data) MIMO control systems. A closed loop system is described by four components: the plant G and three compensator blocks. These blocks could be defined in the form of polynomial system-matrices, state-space equations and transfer-function matrices. See Munro and Bowland (1984).
CTRL-C	Interactive program based on MATLAB for analysis and design of multivariable control systems. The only data structure available is the complex matrix. See Systems Control Technology (1984) and Little et al (1984).
ENPORT-6	Bond graph processor for nonlinear systems. See Rosencode (1985).
DELIGHT	*Design Laboratory with Interaction and Graphics for a Happier Tomorrow.* General purpose, optimization-based, interactive CAD-system. The basic language is UNIX and Fortran inspired with extensions to handle matrices. On top of this various application oriented features could be added. One such extension is DELIGHT.MIMO oriented at multivariable control system design. See Polak et al (1982), Nye and Tits (1982) and Nye (1983).
EAGLES and M	*Environment for Advanced Graphics (LLNL) Engineering Systems on workstation computers.* EAGLES is meant to be an environment to engineering tools with an emphasis on graphical interfaces. The M language processor is one part. The M language handles general expressions involving complex variables, polynomials, transfer functions and matrices of these types. The part of the M language dealing with matrix manipulation is quite similar to MATLAB. String manipulation is also supported. M has also Simnon-like constructs for defining ordinary differential and difference equations for dynamic simulation. See Lawver and Poggio (1985), Gavel et al (1985) and EAGLES (1986).
LSAP	*Linear Systems Analysis Package.* An interactive program with graphics for the classical analysis and design of linear control systems. Allows for the definition of rational transfer functions, either Laplace or Z-transforms. See Herget and Tilly (1985).
MATLAB	*Matrix Laboratory.* Interactive program for numerical linear algebra and matrix computations. The capabilities range from standard tasks such as solving linear equation systems and inverting matrices, through eigenvalue problems and matrix functions, to fairly sophisticated tools such as the singular value decomposition. The only data structure available is the complex matrix. See Moler (1980).
PC-MATLAB	IBM-PC Version of MATLAB with graphics and tools for identification and control systems design and analysis. See

Little and Moler (1987).

SANDYS — *Simulation and Analysis of Dynamic Systems.* Interactive program for simulation of dynamic systems described by ordinary differential and algebraic equations as well as discrete events. See ASEA (1984).

SUNS — *Sussex University Nonlinear Software.* A set of programs for determination of limit cycles, assessment of their stability and plots of the solutions waveforms. The program can handle various configurations from single loop relay systems to more complex nonlinearities and some cases MIMO configurations and sampled systems. The linear part is specified by giving a rational transfer function and the nonlinear part by selection from a menu. See Atherton et al (1985).

2.2 Examples of System Representations

Some examples of system representations used in current CACSD packages are given in this section.

Matrix Languages

Linear timeinvariant systems can be described using arrays. Such systems are conveniently handled in some matrix language like MATLAB (Moler, 1980), or its derivatives MATRIX$_X$ (Integrated Systems, Inc, 1984; Walker et al., 1984a,b; Shah et al, 1985), CTRL-C, (Systems Control Technology, 1984; Little et al., 1984). A system is represented as a matrix quadruplet in CTRL-C. In MATRIX$_X$ it is represented as a system matrix and an integer which gives the order of the system. It is, however, clear that it is not sufficient to only have matrices. A detailed discussion of this is found in Åström (1984). There are a few more data types like polynomials and transfer functions in Blaise (Delebecque and Steer, 1985), IMPACT (Rimvall, 1983, 1986; Rimvall and Cellier, 1984, 1985) and EAGLES, (Gavel et al, 1986). A more sophisticated data structure for systems was used in the Lund packages (Åström, 1983, 1985). Our experiences indicate that it would be very useful to have even more flexible concepts.

Simnon

The system description used in the simulation language Simnon, (Elmqvist, 1975), includes system descriptions for continuous and discrete time systems. A continuous system corresponds to a state models described by an ordinary differential equation like

$$\begin{aligned} \frac{dx}{dt} &= f(x, u, t) \\ y &= g(x, u, t) \end{aligned} \tag{2.1}$$

where x is the state vector, u the input vector and y the output vector. The Simnon representation is

```
CONTINUOUS SYSTEM <system identifier>
INPUT <list of inputs>
OUTPUT <list of outputs>
STATE <list of states>
```

```
DER <list of derivatives>
TIME <variable>
Computation of outputs
Computation of derivatives
Parameter assignment
Initial value assignment
END
```

The standard state space model for a discrete time system is

$$\begin{aligned} x(t_{k+1}) &= f(x_k, u_k, t_k) \\ y(t_k) &= g(x_k, u_k, t_k) \end{aligned} \tag{2.2}$$

where $\{t_k\}$ is a sequence of sampling points. In Simnon such a system is described as

```
DISCRETE SYSTEM <system identifier>
INPUT <list of inputs>
OUTPUT <list of outputs>
STATE <list of states>
NEW <list of new states>
TIME <variable>
TSAMP <variable>

Computation of outputs
Computation of new values of the states
Update the TSAMP-variable
Modify states in continuous subsystems
Parameter assignment
Initial value assignment
END
```

Notice that this description is analogous to continuous systems. There is however a new variable TSAMP which gives the next time that the system should be sampled. In Simnon it is also possible to connect subsystems by using a connecting system which is described by

```
CONNECTING SYSTEM <system identifier>
TIME <variable>
Computation of inputs
Parameter assignment
END
```

The notation in Simnon is very natural for a control engineer. Long experience of using it has also shown that it is very easy to teach and use.

Discussion

The matrix based languages lack a proper system concept. This means that it is difficult to implement operations which are naturally viewed as operations on a system. We cannot make the natural abstractions used in system theory. Low level matrix operations have to be used instead. Simnon has a notion of systems. A drawback with this notion is however that it only admits flat interconnections. Particularly when dealing with large systems it would be desirable to have an hierarchical interconnection of subsystem. One possibility to do this was suggested in Åström (1984). A more flexible approach is suggested in this report. Since Simnon is a simulation language there are also only a limited number of system operations which are supported.

3. Requirements

We will now briefly discuss some key issues in system representation. An important requirement is that the descriptions introduced should admit hierarchical system representations. Another is that it should be convenient to express systems composed of regular patterns of similar components in a convenient way.

To discuss suitable system descriptions we must also know how they are typically used. Typical operations on a system may be to:

Combine several subsystems into a new subsystem.
Expand a system into its subsystem.
Find interconnected loops.
Compute steady state operating points.
Compute steady state input-output relations.
Simulate.
Linearize.
Describe region of validity of linearized model.
Analyse stability, reachability and observability.
Make a Kalman decomposition.
Compute system inverses.
Compute sensitivity functions.
Compute well-conditioned linear representations.
Find linear characteristics like:
  poles and zeros,
  transfer functions,
  frequency curves.
Transform system representations.
Perform and validate control design.
Make model reductions.
Fit parameters to experimental data.
Find graphical representations.

Some of these operations are conveniently done numerically. Others require formula manipulation. It is therefore essential that the system can support numerical as well as formal calculations.

To describe systems it is thus necessary to have a rich structure which makes it possible to describe hierarchical interconnections of subsystems where each subsystem in turn is composed of subsystems. The subsystems may be of different types. They may be described in terms of state models, as input-output relations like impulse responses or transfer functions. We also have a need for descriptions of different complexity.

4. System Structure

Representation of system structure is a key element when dealing with complex systems. Graphical representations, like block diagrams, signal flow diagrams and bond graphs are common for this purpose. They can be used to present details of subsystems as will as to give an overview of systems. In a block diagram description a subsystem is represented by a box and interconnections by lines between the boxes. See Figure 4.1. A line can represent a simple connection which tells that the variables at the connections are the same. It can also represent a more complex situation where several variables are involved. It is common practice to introduce arrows to indicate causality when this is possible. There may also be special symbols to denote simple operations like addition and multiplication of signals. There are many related descriptions like signal flow graphs and bond graphs.

To capture descriptions like block diagrams it is necessary to introduce the notions of subsystems and interconnections. In this report we will only consider systems with well defined inputs and outputs. Such a system can be regarded as an abstraction of a box with input and output terminals. It can be represented as an object with the variables

```
Name
Inputs
Outputs
Subsystems
Connections
```

The inputs and outputs can be simple variables but they could also be objects with properties like units, range, etc. A primitive connection is a pair of input and output terminals. The connection implies that the corresponding terminal variables are the same. To avoid ambiguity the name of the associated system is also given. The notation `(Regulator sp)` denotes the terminal sp of the regulator. The system in Figure 4.1 can be represented as follows

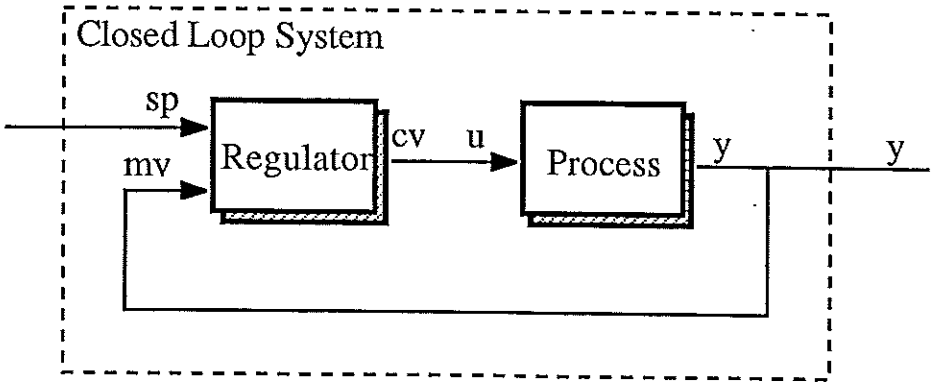

Figure 4.1 Example of a hierarchical block diagram. The ClosedLoopSystem is composed of two subsystems Process and Regulator.

```
Name ClosedLoopSystem
Input r
Outputs y
Subsystems Regulator Process
Connections r (Regulator sp)
(Regulator cv) (Process u)
(Regulator mv) (Process y)
y (Regulator y)
```

The regulator has the representation

```
Name Regulator
Input mv sp
Outputs cv
Subsystems nil
Connections nil
```

It thus has two inputs, mv the measured value and sp the set point. It has one output the controlled variable cv. The regulator has no subsystems and consequently no connections. In such a case the corresponding variables will be not be given.

The process in Figure 4.1 has the representation

```
Name Process
Input u
Outputs y
```

This notation is simple, natural and quite powerful.

Methods

A system structure has a number of associated operations. Examples of basic low level constructor and destructor function are

```
MakeSystem
AddInputs
DelInputs
```

Basic query and selector functions of the type

```
Inputs?
Inputs
```

are also needed to work with a system structure. The function `Inputs?` returns true if the subsystem has inputs and the function `Input` returns all inputs to a given subsystem. These functions operate on one level only. There are also primitive display functions like

```
ShowSystem
```

and a system editor which admits a structured editing of a system.

The functions discussed so far relate to a particular system only. For a system with subsystems it is also of interest to find all attributes of the system and of all associated subsystems. This is done by the functions

```
AllInputs
AllOutputs
AllSubsystems
AllConnections
```

It is sometimes also desirable to show the attributes hierarchically corresponding to the the subsystem hierarchy. This is done by the functions

```
HierarchyOfInputs
HierarchyOfOutputs
HierarchyOfSubsystems
HierarchyOfConnections
```

Several functions are useful in order to explore the structure of a system. There are some auxiliary functions which are useful to explore the connections. The functions

```
InputsConnectedTo
OutputsConnectedTo
```

return the systems which are connected to the input and output terminals of a given system. The function

```
ContainedIn
```

gives all the systems which contain a given system as a subsystem.

A loop is a closed path obtained by scanning a connection of subsystems in the direction defined by the input-output causality. The functions

```
Loop?
Loop
AllLoops
```

tell if a given subsystem is contained in a loop, gives a loop associated with a given subsystem and all loops associated with a given system. These functions can be used to trace a given collection of subsystems.

System Operations

It is convenient to have a number of operations which act on a system and generate new systems. This is accomplished by the functions

```
Aggregate
Disaggregate
```

The function `Aggregate` applies to a collection of subsystems and gives a new subsystem. The appropriate connections are generated from the aggregated subsystems.

There are also a number of other system operations which are useful to form composite systems from simple ingredients. Typical examples are

```
Invert
ParallelConnect
SeriesConnect
FeedbackConnect
```

These operations will have to operate on many different properties of a system. They will create new system with the appropriate properties.

5. System Behavior

Only topological properties of a system i.e. structure and interconnections, have been discussed so far. To describe a system it is also necessary to describe how it behaves. Systems behavior is a very rich field. Examples of categories of behavior are

```
Static
Qualitative
StateSpace
StochasticStateSpace
StochasticInputOutput
LinearStateSpace
TransferFunction
TransientResponse
DescribingFunction
```

These categories can be described as objects. The static behavior can be described by a nonlinear function. Several methods are needed to work with static behavior e.g.

```
FindOutput
FindInput
Linearize
MaxGain
MinGain
DegreeOfLinearity
OperatingRange
```

Qualitative behavior attempts to describe some gross properties of a system like gain, time constants, and estimate of largest dynamic gain, some measure of nonlinearity, and some measure of how deterministic a system is. Associated with these properties we also need methods to obtain these properties from the more detailed representations. The ideas developed for automatic tuning of regulators are quite useful for this purpose. See Åström and Hägglund (1984). It is quite useful to allow qualitative system descriptions because it allows qualitative reasoning about system properties. In large systems composed of many subsystems we may e.g. neglect an interconnection if a system with a very low gain is connected in parallel with a system with a very high gain. In a simulation where we are exploring phenomena in a time scale of minutes we may use static models for subsystems whose time constants are less that one tenth of a second.

It is also very useful to introduce the `ValidityRange` property to indicate the region of validity of the model. This can be described as a subset of the product of the input spaces and the state spaces. With such a feature it is possible to write a simulation program which will raise an exception if the state of the system goes outside the region of validity during a simulation.

6. Conclusions

The purpose of this work has been to view CACE packages as high level problem solving languages. This approach gives a good unified way to view and analyse different packages. Because of budgetary constraints the project was smaller than originally planned. By collaborating with other groups we have however been able to make a reasonable coverage. A study of different packages has shown that system representations is a key issue which is poorly delt with in existing systems. A special study was therefore made of this problem. Different ways to describe systems have been investigated in the study. It has been shown that interconnected systems are conveniently described using object-oriented programming. A small prototype to test the ideas has also been implemented and tested.

Our experiences indicate that the system description proposed is natural and easy to work with. A complete system for describing system structure can be implemented following the ideas outlined.

A graphics user interface of the type described in Elmqvist and Mattsson (1986) and Mattsson et al (1986) is a natural interface to the descriptions. Exploratory work along these lines is under way.

The work has been influential in shaping the future of the CACE project.

Acknowledgements

The results of this paper were obtained as part of the project Computer Aided Control Engineering (CACE) at Lund Institute of Technology. We are grateful to the National Swedish Board of Technical Development (STU) who has supported this project under contract 85-4808. We also would like to thank Wolfgang Kreutzer and Karl Erik Årzén for many useful discussions.

References

ACSL (1986): *ACSL - Advanced Continuous Simulation Language – Reference Manual*, Fourth Edition, Mitchell and Gauthier, Assoc., Inc., Concord, Mass., USA.

ASEA (1984): "SANDYS – Användarhandledning, H6000," ASEA, Västerås, Sweden.

ÅSTRÖM, K.J. (1983): "Computer Aided Modelling, Analysis and Design of Control Systems—A perspective," *IEEE Control Systems Magazine*, Vol. 3, No. 2, May 1983, 4–16.

ÅSTRÖM, K.J. (1984): "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.

ÅSTRÖM, K.J. (1985): "Computer Aided Tools for Control System Design—A perspective," in M. Jamshidi and C.J. Herget (Eds.): *Computer-Aided Control Systems Engineering*, North-Holland, pp. 3–40.

ÅSTRÖM, K.J. and HÄGGLUND, T. (1984): "Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins," *Automatica* **20**, No. 5, 645–651.

ÅSTRÖM, K.J. and W. KREUTZER (1986): "System Representations," *Proceedings of the IEEE Control Systems Society Third Symposium on Computer-Aided Control Systems Design (CACSD)*, Arlington, Virginia, September 24–26, 1986, Also available as internal report TFRT-7330.

ATHERTON, D.P., O.P. MCNAMARA, M.D. WADEY and A. GOUCEM (1985): "SUNS: The Sussex University Nonlinear Control Systems Software," *Proceedings of the 3rd IFAC/IFIP International Symposium CADCE'85 on Computer Aided Design in Control and Engineering Systems*, The Technical University of Denmark, July 31 – August 2, 1985, pp. 173–178.

BARKER, H.A., M. CHEN, C.P. JOBLING and P. TOWNSEND (1986): "Interactive Graphics for the Computer-Aided Design of Dynamic Systems," IEEE International Conference on Systems, Man and Cybernetics, Atlanta, Georgia, October 14–17, 1986.

BARKER, H.A., M. CHEN, P.W. GRANT, C.P. JOBLING and P. TOWNSEND (1987): "The Development of an Intelligent Man-Machine Interface for Computer-Aided Design, Simulation and Implementation of Control Systems," IFAC World Congress, Munich, Germany, 1987.

CAMBRIDGE CONTROL LIMITED (1983): "TSIM–User's Guide, Reference Manual," Cambridge Control Limited, Cambridge, UK.

CELLIER, F.E. and M. RIMVALL (1986): "Computer-Aided Control System Design Techniques and Tools," CERL–Report: 86/04, Dept. of Electrical & Comp. Engineering, University of Arizona, Tucson, Arizona.

DELEBECQUE, F. and S. STEER (1985): "The Interactive System Blaise for Control Engineering," *CADCE '85, 3rd IFAC/IFIP Symposium on*

Computer Aided Design in Control and Engineering Systems, Lyngby, Denmark, July 31–August 2, 1985, pp. 44–46.

DELEBECQUE, F. (1986): "Some Remarks About the Design of an Interactive CACSD Package: The Blaise Experience," *IEEE Control Systems Society, Third Symposium on Computer-Aided Control Design*, Arlington, Virginia, September 23–26, 1986.

EAGLES (1986): "EAGLES/Controls – Documentation," Lawrence Livermore National Laboratory, Livermore, California, USA.

ELCS (1987): "The Extended List of Control Software, ELCS," Number 3, February 1987, (Eds.) D.K. Frederick, C.J. Herget, R. Kool and M. Rimvall, Dept. of Automatic Control, ETH, CH-8092 Zürich, Switzerland.

ELMQVIST, H. (1975): "SIMNON - User's Manual," CODEN: LUTFD2/ TFRT-3091, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ELMQVIST, H. (1978): "A Structured Model Language for Large Continuous Systems," Ph.D Thesis CODEN: LUTFD2/TFRT-1015, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ELMQVIST, H. (1985): *LICS - Language for Implementation of Control Systems*, Report CODEN: LUTFD2/TFRT-3179. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ELMQVIST, H., K.J. ÅSTRÖM and T. SCHÖNTHAL (1986): *SIMNON – User's Guide for MS-DOS Computers*, Department of Automatic Control Lund Institute of Technology, Lund, Sweden.

ELMQVIST, H. and S.E. MATTSSON (1986): "A Simulator for Dynamical Systems Using Graphics and Equations for Modelling," *Proceedings of the IEEE Control Systems Society Third Symposium on Computer-Aided Control Systems Design (CACSD)*, Arlington, Virginia, September 24–26, 1986.

GAVEL, D.T., C.J. HERGET, and B. S. LAWYER (1986): "The M Language—An Interactive Tool for Manipulating Matrices, Systems and Signals.," Dynamics and Controls Group, Engineering Research Division, Lawrence Livermore National Laboratory, Livermore, California.

GRANDA, J.J. (1983): "A Guide to Using Camp," Department of Mechanical Engineering, California State University, Sacramento, California.

HERGET, C.J. and D.M. TILLY (1985): "Linear Systems Analysis Program," in Jamshidi, M. and C.J. Herget (Eds.): *Computer-Aided Control Systems Engineering*, North-Holland, Amsterdam, The Netherlands, pp. 53–60.

INTEGRATED SYSTEMS, INC (1984): "MATRIX$_X$ User's Guide, MATRIX$_X$ Reference Guide, MATRIX$_X$ Training Guide, Command Summary and On-line Help," Integrated Systems, Inc., Palo Alto, California.

KING, R.A. and J.O. GRAY (1985): "A Flexible Data Interpreter for Computer Aided Design & Simulation of Dynamic Systems," *CADCE '85, 3rd IFAC/IFIP Symposium on Computer Aided Design in Control and Engineering Systems*, Lyngby, Denmark, July 31–August 2, 1985, pp. 87–91.

LAWVER, B. and P. POGGIO (1985): "EAGLES Requirements," Computer Systems Research Group, Engineering Research Division, Lawrence Livermore National Laboratory, Livermore, California.

LITTLE, J.N., A. EMAMI-NOEINI, and S.N. BANGERT (1984): "CTRL-C and Matrix Environments for the Computer-Aided Design of Control Systems," in Bensoussan and Lions (Eds.): *Analysis and Optimization of Systems*, Lecture Notes in Control and Information Sciences, Springer-Verlag, Berlin, Also in Jamshidi, M. and C.J. Herget (eds) Computer-Aided Control Systems Engineering, North-Holland, Amsterdam, The Netherlands, 111–124.

LITTLE, J. and C. MOLER (1987): "A Preview of MATLAB," The MathWorks, Inc., Sherborn, MA, USA.

MACIEJOWSKI, J.M. and A.G.J. MACFARLANE (1982): "CLADP: The Cambridge Linear Analysis and Design Programs," *Control Systems Magazine* **2**, 4, Dec 1982, 3–8, Also in Jamshidi, M. and C.J. Herget (Eds.) (1985). Computer-Aided Control Systems Engineering, North-Holland, Amsterdam, The Netherlands, pp. 125–147.

MATTSSON, S.E., H. ELMQVIST and D.M. BRÜCK (1986): "New Forms of Man-Machine Interaction," Final Report 1986-09-30, STU project 84-5069, STU program: Computer Aided Control Engineering, CACE, Report CODEN: LUTFD2/TFRT-3181, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

MUNRO, N. and B.J. BOWLAND (1984): "Computer Aided Control System Design Software – User's Guide," Control Systems Centre, UMIST, Manchester, UK.

MUNRO, N., J.M. EDMUNDS, G. BOWE, S. GOODFELLOW, W. SWINDELLS and D. LOMAS (1986): "Examination of various control system design software facility," Control Systems Centre, UMIST, Manchester, UK.

MOLER, C.B. (1980): "MATLAB - User's Guide," Department of Computer Science, University of New Mexico, Albuquerque, USA.

NYE, W.T. and TITS, A. (1982): "DELIGHT for Beginners," Memorandum No. UCB/ERL M82/55, Electronics Research Laboratory, College of Engineering, University of California, Berkeley.

NYE, W.T. (1983): "DELIGHT. An Interactive System for Optimization-Based Engineering," Memorandum No. UCB/ERL M83/33, Electronics Research Laboratory, College of Engineering, University of California, Berkeley.

POLAK E., P. SIEGEL, T. WUU, W.T. NYE and D.Q. MAYNE (1982): "DELIGHT.MIMO: An Interactive Optimization-Based Multivariable Control System Design Package," *IEEE Control Systems Magazine* **2**, No. 4, Dec 1982, 9–14, Also in Jamshidi, M. and C.J. Herget (Eds.) (1985): Computer-Aided Control Systems Engineering, North-Holland, Amsterdam, The Netherlands, pp. 139–147.

RIMVALL, M. (1983): "IMPACT, Interactive Mathematical Program for Automatic Control Theory, User's Guide," Department of Automatic Control, Swiss Federal Institute of Technology, ETH-Zentrum, Zürich, Switzerland.

RIMVALL, M. (1986): "Man-Macine Interfaces and Implementation Issues in Computer-Aided Control System Design," Ph.D Thesis, Department of Automatic Control, Swiss Federal Institute of Technology, ETH-Zentrum, Zürich, Switzerland.

RIMVALL, M and F. CELLIER (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.

RIMVALL, M. and F.E. CELLIER (1985): "A Structural Approach to CACSD," *Computer-Aided Control Systems Engineering*, North-Holland, Amsterdam, The Netherlands, pp. 149–158, In Jamshidi, M. and C.J. Herget (Eds.).

ROSENCODE (1985): "ENPORT-6 – User's Manual," Rosencode Associates, Inc., Lancing, Michigan, USA.

SERC (1986): "Control and Instrumentation – Subcommittee Review," August 1986, Science and Engineering Research Council Engineering Board, Information Engineering Committee, Control and Instrumentation Subcommittee, Swindon, UK.

SHAH, S.C., M.A. FLOYD AND L.L. LEHMAN (1985): "MATRIX$_X$: Control and Model Building CAE Capability," in Jamshidi, M. and C.J. Herget (Eds.): *Computer-Aided Control Systems Engineering*, North-Holland, Amsterdam, The Netherlands, pp. 181–207.

SYSTEMS CONTROL TECHNOLOGY (1984): "CTRL-C A Language for the Computer-Aided Design of Multivariable Control Systems, User's Guide," Systems Control Technology, 1801 Page Mill Road, Palo Alto, CA.

THOMPSON, P.M. (1984): "User's Guide to Program CC, Version 2," Electrical Engineering Department, 116-81, California Institute of Technology, Pasadena, California.

THOMPSON, P.M. (1985): "Program CC: Technical Information," *Proceedings of the 2nd IEEE Control Systems Society Symposium on Computer-Aided Control System Design (CACSD)*, Santa Barbara, California, March, 13–15, 1985.

WALKER, R., C. GREGORY, JR. AND S. SHAH (1984a): "MATRIX$_X$: A Data Analysis, System Identification, Control Design and Simulation Package," Integrated Systems, Inc., Palo Alto, California.

WALKER, R., S.C. SHAH AND N.K. GUPTA (1984b): "Computer-Aided Engineering (CAE) for System Analysis," *Proc. of the IEEE* **72**, No. 12, 1732–1745.

WIESLANDER, J. (1980a): "Idpac Commands – User's Guide," CODEN: LUTFD2/TFRT-3157, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

WIESLANDER, J. (1980b): "Modpac Commands – User's Guide," CODEN: LUTFD2/TFRT-3158, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

WIESLANDER, J. (1980c): "Synpac Commands – User's Guide," CODEN: LUTFD2/TFRT-3159, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.