Symbolics in control design: prospects and research issues

Christensen, Anders

Published in:
Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design

Link to article, DOI:
10.1109/CACSD.1994.288942

Publication date:
1994

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

Symbolics in Control Design: Prospects and Research Issues

Anders Christensen, Assoc. Prof, MScEE, PhD
Institute of Automatic Control Systems,
Technical University of Denmark, DK - 2800 Lyngby, Denmark.
E-mail address ac@sl.dth.dk, FAX +45 42 88 12 95.

Abstract

The symbolic processor is targeted as a novel basic service in computer aided control system design. Basic symbolic tools are examplified. A design process model is formulated for control design, with subsets manipulator, tools, target and goals. It is argued, that symbolic processing will give substantial contributions to future design environments, as it provides flexibility of representation not possible with traditional numerics. Based on the design process, views on research issues in the incorporation of symbolic processing into traditional numerical design environments are given.

Keywords: Symbolic control design, Design Process models, Design State.

1. Introduction

In the control design community, the need for computational support is widespread, as many design projects are of great complexity. Methodology for design of multivariable controllers is continuously updated from mathematics, and models of design is made an inherent need because of the huge number of powerful tools solving interrelated or identical problems.

The history of Computer Aided Control Design dates back to the FORTRAN numerical matrix computation subroutine libraries LINPACK / EISPACK, which together with commercial simulators like SIMNON, ACSL, CSSL etc. have developed into the MATLAB / MATRIX-X numerical matrix computation environment, which today provides a widely accepted standard platform in the control community for design and simulation. Tools for control design and simulation are programmed directly in matrix language and organized into toolboxes. Since 1985, CACSD has been an accepted and active community in the control design field, offering research in control design and implementation of computer tools. In commercial control design products, the recent years have brought no substantial progress since the graphics-based simulator of MATRIX-X. In the research community however, prototype design systems have emerged with interesting new developments. After MATLAB became available, the emphasis has been concentrated on object-oriented methods emphasized in OMOLA (Andersson, 1989), symbolic modelling in DYMOLA (Elmquist, 1992), database systems and definition and support of iterative control design (ANDECS, Grübel et al., 1993), all mainly with support from numerical and graphical basic services. In the theory department, the main problems are support tools for iterative controller design, and abstractions of tools and data needed for this. The "CACSD Framework Reference Model" proposed in (Barker et al. 1993) is under discussion for referencing software structures, as well as useable abstraction on design projects are reported from several sources, f.ex. the inspiring design-support database considerations in (Taylor & Grübel, 1993).

During the recent years, basic services for handling symbolic computations have matured to become stand-alone products (MACSYMA, MAPLE V, MATHEMATICA, etc.) capable of handling the complexity demanded by control designers. Control designers in nonlinear systems have developed implementations of exact linearization-based controllers in practice, which are of a complexity which is not computable by hand (Hahn et al., 1993). In (Christensen, 93), a flexible modelling tool has been developed and shown to connect nonlinear dynamical equations directly to both implementation and analysis.

This paper provides a formulation of aspects of control design which can gain substanitially from the incorporation of symbolic services. In section 2, some basic mechanisms are described, emphasizing things not implementable in numerics. It is argued, that a structural analysis of the control problem shows promising prospects of using symbolics in control design. In section 3, a short structural analysis is made of control design comprising a MacFarlane-like (MacFarlane at al., 1989) design environment, methods, designed "target" data, and a design state. Section 4 provides an analysis of prospects and issues of research, and section five some views on the incorporation of symbolic processing in control design products.

0-7803-1800-5/94/$3.00

2. Why Symbolics ?

When implementing design in symbolics, some basic problems arrive, which propose new questions to be solved. When moving from a numeric to a symbolic environment, data structure and typing can be defined as manipulable variables, and the typing problem thus raises questions of abstraction in new flexible concepts, "a model", "a plant", "a signal", etc., instead of well-defined matrices and vectors. This problem makes semantics of control design an interesting issue. Further, programs, actions and functions can be (are) treated as data items. Hereby, design routines can be flexibly implemented, and can be manipulated, applied, analyzed as operators, etc. This in general means that design systems can be organized with only one implementation of the data, in contrast to the several ones used today for analysis, simulation etc.

It has been found (Christensen, 1992,1993), that basic facilities of the symbolic processors have potential of enhancing flexibility and power of control design:
Late evaluation enables the designer to program abstract control loops in a top / down manner. The control loop can be expressed by an unevaluated expression tree. Take f.ex. a transfer function, implemented by the following MATHEMATICA program (Wolfram, 1991):

```
cloop := D H / (1+D H);
D := 1;
tf := 1 / (1+a1 s+a2 s^2);
a1 :=0.1; a2 :=0.2;
```

Using the := operator, the left hand side variables are stored unevaluated as the right-hand side expressions.

Terms for design, analysis and validation can be implemented in the abstracted control for working on any specific control loop, as long as the abstraction is valid. This is done by *substitution* of the abstracted terms, which is also a standard facility of symbolic processors; in the above example we can insert the plant described by *tf* as follows:

```
cloop /. {H->tf}
```

A root locus equation can be found by varying any parameter entering a polynomial in a 1.st order form:

$$H(a,s) = H_1(s) + a\,H_2(s) = 0 \quad \Rightarrow 1 + a\,\frac{H_2(s)}{H_1(s)} = 0 \qquad (1)$$

Take a root locus equation for the characteristic polynomial for the system. This can be obtained by defining a Root Locus Equation generator:

```
RlocEq[polynom_,parm_] := 1 +
parm(Coefficient[polynom,parm,1]/Coefficient[polyno
m,parm,0])
```

This construction separates the polynomial into *polynomial* = $H_1(s)$ + *name* $H_2(s)$. The root locus equation is now computed by the following sequence, executed in a *dynamic scope*: A root locus equation for the closed loop system above is now found by calling

```
In[5]:= Block[{a2="a2"},
        RlocEq[Denominator[cloop],a2]]   (2)
```

$$Out[5]:= 1 + a2\,\frac{s}{2 + 0.2\,s^2} == 0$$

In thjis example, the dynamic scope produces the characteristic polynomial ***Denominator[cloop]*** parametrized in *a2* (substituted to *"a2"*), and the *RlocEq* routine then reorganizes the polynomial into a root locus equation for the parameter *a2*. From a symbolically defined model, a root locus equation can be obtained also "through" linearization, such that dynamic behavior can be traced to nonlinear model entries (Christensen, 1993), (Blanke & Christensen, 1993).
A special nicety is the ability to perform conversions enabling use of the same models in complex plane analysis as well as algorithmic form.

3. Structures in Control Design

As argued above, a main reason for applying symbolics to control design by computer is the ability to perform structure computations. The present subsection discusses this issue and formulates some of the structures relevant for consideration. A control system design environment, based on the well known approach by (MacFarlane et al., 1989), is shown in Figure 1 (Christensen, 1992). This consists of a real world system holding the actual physical plant and control equipment, a customer representing the goals and objectives of a design session, and a designer. The designer utilizes the tools his "engine" provides, and modifies and refines implemented standard strategies for design to suit his present needs.
The internals of the control system design process can be described as an iteration in the basic activities of modelling, controller design, validation and iteration. These skills consist of a high-level generic task and a set of low-level 'standardized' tools, organized in toolboxes. The generic tasks work from a problem formulation and in cooperation produce a control system. Figure 2 illustrates the situation in form of 3 systems: a manipulator representing the tasks, a goal

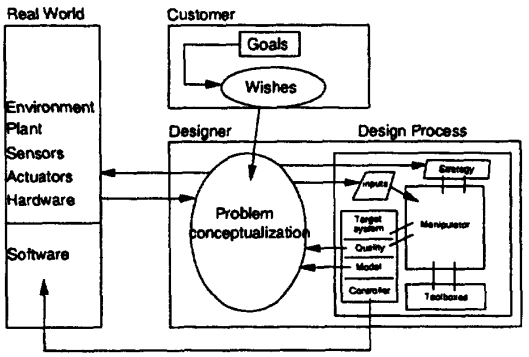

Figure 1: The control design environment

system holding the problem formulation and solution strategy and a target system holding the results.

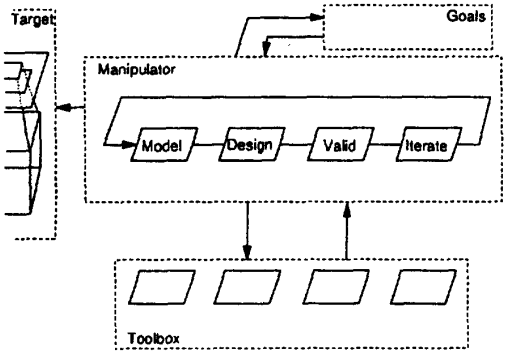

Figure 2: The design process consists of a manipulator system with a generic task level and a tool level and a target system holding the output.

Target System
The target system can be seen as a "variable structure declaration" for the variables generated and used during design. It can be modelled in five levels, as visualized in Figure 3.

Control structure (The closed-loop system level):
The control structure is given by the "arrows" in a conventional block diagram, describing the relations between the main subsystems of plant, environment and controller. The elements of the control structure are links and models as input/output descriptions, or merely identifiers.

Model structure (The model description level):
The model description level is the specification of internal model signals and parameters, and can be represented in several forms, choosing nonlinearity types, operator domains etc.

Physical structure (The implementation level):
The models of controller and plant are behavioral descriptions of a "physical" implementation and can thus be specified for the controller into algorithms of implementation, or for the plant and environment as physical components. The structure of these physical entities can be described in a physical "implementation" level.

Objective & Constraint structure: The validation of a control design utilizes indicators, which are designed as "measuring devices" on the closed-loop system. They are directly related to the produced control system result and are formulated from entities in the three levels below. Having the objectives described and modelled as indicators, constraints and preferences can be represented in a Constraint level referring to the objectives.

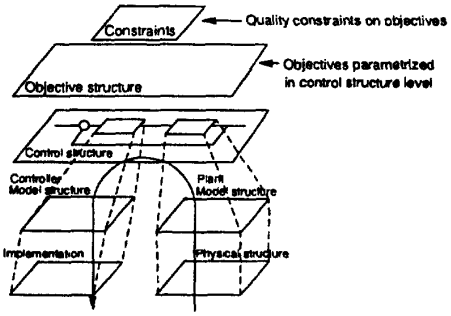

Figure 3: Illustration of the target system. The design process has a procedural interpreta-tion, producing data as indicated by the arrow.

Design State
The design state can be written, specifying from system description of the target to *structure* of the variables, *parameters* to insert into the structure, and *values* for replacing the parameters in numerical computation. All the concepts are chosen and designed by *methods* and their inputs. The state is shown in Figure 4, comprising methods, structure and parameters.

4. Key Issues for Incorporating Symbolics in CACSD

4.1: Model Representation
The typical representations of models in control design can be formulated in short by (3).

$$\textit{entities} > \textit{laws} > \textit{equations} > \textit{operators} > \textit{algorithms} \qquad (3)$$

Due to the numerical nature of the existing standard

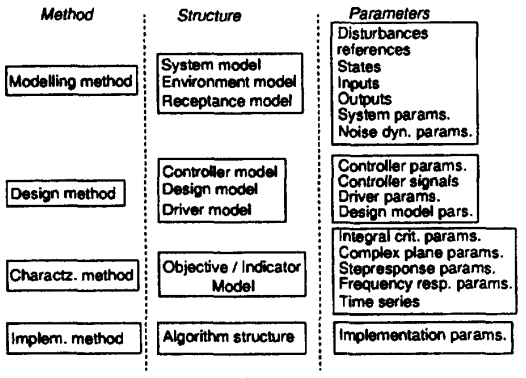

Figure 4: The design state, comprising methods chosen, structure, and parameters. Numerical values are inserted for parameters (Not shown).

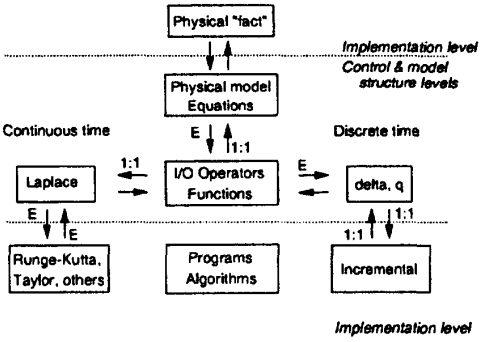

Figure 5: Model domains used in control design. Transformations are 1:1 (no error); or E (errors made)

design tools, this production sequence above is done by hand, as it involves the manipulation of symbols. Next, a model-generating tool is implemented, which is capable of producing numerics in a prestructured form. Conversions are provided between numerical representations between its linear instances, but it necessitates the premises of the model to be *values*. The user must again provide the structure information by himself. Thus, original premises are not, as in the symbolics example above, reproducible from the resulting model representations.

Basic modelling category theory has been provided by Leitch (Leitch, 1992). A modelling tool representing the link between entities (called phenomena), laws and equations is derived in the Hybrid Phenomena Theory (Woods,1993) and implemented in Common LISP. Object-orientation has been used in OMOLA (Andersson,89) in the same context with reuseability as the main goal. A 10-year old modelling language DYMOLA has reappeared (Elmqvist, 1992). This is a symbolic model formulation tool capable of producing model equations and simulator code programs for standard commercially available simulators. The DYMOLA tool provides solutions and inspiration to development of symbolic tools for control designs. Specifying equations and data, it has been shown possible, by using a standard symbolic processor, to implement in some 300 lines of code a model generator capable of linearization and conversion between equations, algorithms and operators (Figure 5), thus forming a sound common basis for analysis and simulation (Christensen, 1993). The model structure level and the implementation level of the target system has been shown directly connectable in terms of a flexible adaptive control implementation (Torp et al., 1993).

4.2: Iteration and accounting support

In terms of validation (or analysis), the interesting feature of the symbolic processor is the ability to represent a system with its design model, such that all indicators are traceable backwards to their physical coefficients. Take the design diagram of Figure 6 and denote the indicator (or design method "driver") J. Then, given real plant behavior P and controller structure C, we are seeking a set of values to C designating an optimal solution to J:

$$C_v = arg\{minJ(P,C_s)\} = DJ(P,C_s)$$

where a design method D is abbreviated to include a design driver J, and indices v and s denote values and structure, respectively. As we do not have the "real" model of P, some designed model is used:

$$\hat{C}_v = arg\{\min J(\hat{P},\hat{C}_s)\} = DJ(\hat{P},\hat{C}_s)$$

The "real" obtained quality of the design is then

$$J(P,\hat{C}) = J(P,DJ(\hat{P},\hat{C}_s))$$

Now, replacing the "real" model by a "master" for the design models, designated Ptilde, we can rewrite the solution as

$$C_v = J\{\tilde{P},D_iJ_i(\hat{P},\hat{C})\}$$

into which we can insert several structurally different designs.

The validation problem can be programmed very flexibly by adopting this scheme to a *general validation*

problem (Christensen, 1992), also expressing the dependency on design driver parameters θ:

$$V = J(P,D,J_i[\theta](\hat{P},\hat{C}))$$

From this abstract type of formulation, performance indices, sensitivity functions, root locus derivations and robustness measures can be implemented symbolically as shown in the root locus example above. The "physical" parameters of the the plant can be used in the validation, and if it is possible to formulate the design routine as an operator, it is possible to evaluate all "premises" to the design problem in the closed-loop result. For several known design synthesis methods this scheme is indeed possible, but some effort is needed to describe the applicability of design methods based on numerical optimization, as it may not be possible to arrive at a symbolical solution.

4.3: Derivation of a meta - control design theory

There is widespread activity in research concentrating on building models of the design process in itself, enabling the formulation of generic tasks capable of administrating the actions of a designer (Mostow, 1985; Brown & Chandrasekaran, 1989). These models have central concepts as

- Definitions of design projects/problems
- De/Recomposition of design projects
- Description of abstraction in design
- Emphasis on design decisions, rationales and control of iteration
- construction of design history databases and learning mechanisms

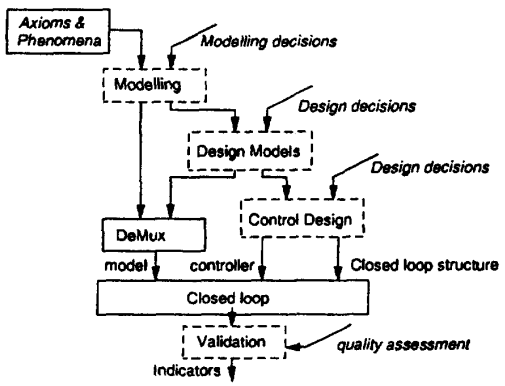

***Figure 6**: Design decisions during modelling, design and validation. Design produces a closed loop system and an indicator set.*

Assuming conventional design, the design actions can be sketched as seen in Figure 6. The diagram shows the premises P of design (Axioms, phenomena, decisions and quality asessment), the methods/tasks M involved (dashed boxes), and the result R in form of model, controller and indicators.

From P, M and R, basic logic can be used to define generic problems in design (Coyne at al., 1990). The conventional controller design process referring to the design process description above can be seen as producing output following a path emerging from the physical level of the plant upwards through the top level of the hierarchy and down again through the controller "branch", as indicated by the arrow in Figure 3. Alternatively, *Integrated Design*, defined by simultaneous design of control and plant, can be pictured in the target system by specifying indicators and quality constraints first, thereafter emerging from top to bottom down through control structure level and model structure level to the specification of implementation in both plant physics and control software. In this case the generic design tasks will have a much different content (Christensen & Lind, 1993), and only few of these models have been published yet.

Again referring to Figure 6, the formulation of abstract models of total design supports the iteration by making design decisions and tuning parameters accesible from the end result, thereby increasing the ability to criticise the solution and identifying critical input parameters to the design. As a numerical processor like MATLAB manipulates values for parameters and does not keep track of the identifiers for the input entities, this is not possible at the present state of the art. The cooperation of accounting and iterative design and the focus on indicators has been covered in an inspiring manner by the development of the ANDECS design system, (Grübel et al., 1993). ANDECS provides a numerical design environment with macro facilities for design and validation specification, and automatic design iterations by optimization of user-chosen validation indicators, and design history databases. In ANDECS, design steps are implemented with a coordinate triple incrementing coordinates by the steps a) modification of models, b) modification of design parameters and c) modification of indicators.

The "standard problem" of robust control design, see e.g. (Francis, 1987), is an example of a flexible basis for this kind of considerations. It is important to notice, though, that design methods are not only synthesis-based optimization, but also comprise trial-and-error, loop shaping etc., as well as a consisitent theory must be able to deal will control structure changes.

5. CACSD Environments with Symbolics

In the MATLAB family, all design routines are implemented in macro facilities based on a numerical matrix computation facility with a command interpreter. These tools demand the structure of the inputs to be

known implicitly. Thus, compound-type variables from complex design methodologies become difficult to comprehend. The tendency of the recent years has been development of graphical interfaces, such that the block diagram-like notation is used as basis for the control design. As argued above, representation by symbolics is much closer to theory, textbooks and the designer's way of thinking, as it manipulates structure. Incorporation of symbolics into control design environments as f.ex. MATLAB should be done by extending the numeric routines with the basic services (see CACSD framework reference model, Barker et al. 1993) of a loosely typed symbolic processor: The flexibility is retained, and strong typing can be implemented by the user himself to suit his own needs. The compound system could operate either as one product, or by a operating system level object-sharing workspace with access from otherwise independent numeric and a symbolic processing tools, as outlined in (Ravn & Szymkat, 1992). Very recently, both symbolics products MATHEMATICA and MAPLE have been provided with mex-based data and command links to MATLAB, which hopefully will make further investigations straightforward.

6. Conclusion

In investigation of symbolics for control system design, the main interest is features provided by symbolics as opposed to numeric/graphical environments like MATLAB. It is the experience of the investigation work presented here, that symbolics can provide much new insight into "old" design routines, which especially validation for physically based models can profit from. Symbolics, graphics and numerics handle different aspects of system and design definition, and are all needed in control design tools today. It is believed that symbolics will offer basic services for structuring the design and excution problems to be solved by both graphical and numerical environments.

References:

[1] BARKER, H.A.; JOBLING, C.P.; SZYMKAT, M.; RAVN, O. (1993): " A requirements analysis of future environments for computer-aided control engineering, *Preprints 12th IFAC World Congress 1993*, Sydney, Australia

[2] BLANKE, M., CHRISTENSEN, A. (1993): "Rudder-Roll Damping Autopilot Robustness to Sway-Yaw-Roll Couplings", *Proceedings 10th Ship Control Systems Symposium*, Ottawa, Canada

[3] BROWN, D.C.; B. CHANDRASEKARAN (1989): *Design Problem Solving - Knowledge Structures and Control Strategies*, Research Notes in Artificial Intelligence, Pitman Publishers, London 1989.

[4] CHRISTENSEN, A. (1992): *Models of Control Design - Modelling and Validation in Ship Control*, Ph.D. Thesis, Institute of Automatic Control Systems, Technical University of Denmark.

[5] CHRISTENSEN, A., LIND, M. (1993): "A modelling Framework for Integrated Design of AGV systems", *Proceedings 1st IFAC Symposium on Intelligent Autonomous Vehicles*, Southampton, UK.

[6] CHRISTENSEN, A. (1993): Prospects in Symbolic Proces-sing for Modelling in Control System Design, *Proceedings 12th IFAC World Congress 1993*, Sydney, Australia

[7] COYNE, R.D.; M.A. ROSENMAN; A.D. RADFORD; M. BALCHANDRAN; J.S. GERO (1990): *Knowledge-Based Design Systems*, Addison-Wesley Publishing Co.

[8] ELMQUIST, H. (1992): *DYMOLA* - Modelling software documentation, DynaSim Corp, Lund, Sweden 1992

[9] FRANCIS, B. (1987): A Course in H_∞ Control Theory, *Lecture notes in control and information sciences* vol 187, Springer-Verlag, Germany

[10] GRÜBEL, G.; H-D. JOOS; R. FEINSTERWALDER; M. OTTER (1993): The ANDECS Design Environ-ment for Control Engineering, *Preprints 12th IFAC World Congress 1993*, Sydney

[11] HAHN, H.; K.-D. LEIMBACH; X. ZHANG (1993): Nonlinear Control of a Spatial Multi-Axis Servo-Hydraulic Test Facility Test Facility, *Proceedings 12th IFAC World Congress 1993*, Sydney, Australia

[12] LEITCH, R. (1992): Artificial Intelligence in Control;some myths, some fears but plenty prospects, *Computing & Control Engineering Journal*

[13] MOSTOW, J. (1985): Towards Better Models of the Design Process, *AI Magazine*, spring 1985.

[14] MACFARLANE, A.G.J., G. GRÜBEL, J. ACKERMANN (1989): Future Design Environments for Control En-gineering, *Automatica*, vol. 25, no.2, pp 165-176, March 1989

[15] RAVN, O.; M. SZYMKAT (1992): The evolution of CACSD Tools - A Software Enginnering Perspective, *Proc. IEEE Symposium on CACSD*, Napa, California

[16] TAYLOR, J.H.; G. GRÜBEL (1993): Data-base manage-ment guidelines for computer-aided control engineering, *Proceedings 12th IFAC World Congress*, Sydney, Australia 1993

[17] TORP, S.; P.M. NØRGÅRD; A. CHRISTENSEN, O. RAVN (1993): Implementation Issues in CACSD, *Preprints 1st IEEE/IFAC Symposium on CACSD*, Arizona, USA

[18] WOLFRAM, S. (1991): *Mathematica - A System for Doing Mathematics by Computer*, 2nd Edition, Addison-Wesley 1991.

[19] WOODS, E. (1993):*The Hybrid Phenomena Theory*, PhD thesis, Techn. University of Norway, Trondheim, Norway.