Development of a New Generation of Interactive CACSD Environments

Grantham K. H. Pang, Mathukumalli Vidyasagar, and Andrew J. Heunis

Abstract: This paper describes the development of a new interactive computer-aided control system design package for the design of linear multivariable control systems. The package incorporates the Stable Factorization design technique together with other recent design methodologies, such as the Systematic Design Approach. To provide a high level of interactive design environment for the user, knowledge-based expert systems development tools are provided within the package. In addition, the user is provided with the facilities to write user-defined functions and procedures.

Introduction

The earlier version of the Stable Factorization design package (SFPACK) was developed at the University of Waterloo. It is essentially an extension of the MATLAB program. The Stable Factorization design technique is implemented using a set of high-level executive commands that sit on top of the basic set of MATLAB commands as well as the EISPACK and LINPACK routines. Whereas the basic set of commands provided in MATLAB allows one to perform various operations on matrices, the command set in SFPACK consists of additional commands that can be performed on packed matrix representations of systems.

However, the previous version of SFPACK does not provide facilities such as user-defined functions in MATRIXx or M-files in PC-MATLAB; therefore, its programming environment is limited. Also, the addition of any new commands (or primitives) and new syntax to the command language involve many changes in the stack of MATLAB. Therefore, a new version of SFPACK is developed that is not based on extending the original MATLAB program. In this paper, the development of the new version of SFPACK is described. The current status of computer-aided control system design (CACSD) tools will be discussed briefly. Then, the proposed environment for computer-aided design of control systems is presented.

Current Status of CACSD Tools

Presently, there are a number of CACSD programs and packages available from industry and universities [1]. Readers who are interested in a summary of these packages should refer to [1]-[3]. The objective of this section is to summarize two common features and shortcomings of existing CACSD tools. These two shortcomings are that (1) very few packages permit the user to augment the capabilities of the package by writing routines, and (2) almost all packages are really analysis tools purporting to be design tools.

It is clear that, in view of the wide variety of control systems in existence, ranging from aircraft and spacecraft to chemical reactors, no single methodology or computer package is well-suited to all applications. Moreover, within each discipline, over the years practitioners have developed various insights and specialized techniques that they would be loath to discard. Thus any CACSD product claiming to be universally applicable without providing users with the ability to add specialized instruction sets naturally will meet market resistance.

Another shortcoming of existing CACSD tools is that almost all of them are analysis packages that claim to be design packages. By this is meant the following: Almost all existing packages allow one to analyze a given system with the controller specified by the designer in place, by computing and displaying various items of data such as frequency and time responses and so forth. This, in turn, allows the designer to assess the extent to which the current design meets the specifications. However, should the current controller design prove to be unsatisfactory, the existing CACSD packages do not provide any guidance as to how the design should be modified to meet the specifications.

Overview of SFPACK

The SFPACK design package handles linear time-invariant control system analysis and design for both single- and multi-input and multi-output systems. The command language of the package is identical to the MATLAB program, and, hence, the versatile matrix operation facility fundamental to MATLAB as well as all the basic commands in MATLAB are available. Additional commands are included to provide a richer design environment for the user. The 150 built-in commands of the package can be broadly divided into five groups. The first group includes basic matrix functions for performing various matrix decompositions, such as eigenvalue and singular-value decomposition, schur triangular decomposition, etc. Functions for obtaining the real schur form of a matrix are also provided as well as functions for obtaining a solution to the continuous and discrete Riccati, Sylvester, and Lyapunov equations. The function for obtaining the generalized eigenvalues and eigenvectors using the QZ algorithm is also included.

The second group includes basic commands for implementing the Stable Factorization design technique. There are functions for performing left-coprime, right-coprime, and doubly coprime system factorizations. Commands for forming a stable left-inverse, stable right-inverse, balanced realization of a system are also available. Within this group, there are also facilities for finding the poles of a system, serial and parallel connections of systems, obtaining time responses to both step and impulse input, etc.

The third group of commands are for use in developing knowledge-based systems within the control environment. They include commands for asserting and removing facts, rules, and frames. Question statements can also be stored as well as the goals in the inference process. A command 'run' is used to start the inference mechanism. There is also a group of utilities for displaying and storing the knowledge base, etc.

The fourth group contains the utilities commands for loading and saving matrices, keeping a diary of a session, deleting vari-

An early version of this paper was presented at the 1988 American Control Conference. The authors are with the Department of Electrical Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.
ables, plotting graphs, etc. The command 'system' allows the user to execute Unix commands as if outside SFPACK. Hence, the user can edit an external file or copy files from within the package. The last group of commands belongs to elementary mathematical functions, including finding square root, sine, cosine, tangent, logarithm, etc.

Language of the CACSD Environment

In this section, the environment of the SFPACK program for computer-aided design of control systems is described. It is an interactive environment for CACSD that can be used to design controllers for linear multivariable systems. The language supports design methodologies based on the Stable Factorization approach, but it is not restricted to this method.

The essential features of this language are the following:

- It is an interactive terminal language that can execute commands as soon as they have been entered, after which control is returned to the user. That is, the environment is command driven.
- The language supports an extended data structure: the packed matrix. A matrix can be input interactively as follows:

\[
\begin{pmatrix}
1 & 2 & 1 & 0 \\
3 & 4 & 2 & 1 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{pmatrix}
\]

In SFPACK, the preceding data model is extended to the packed matrix form, which is a natural form for representing a state-space and transfer-function models can be performed with the commands 'tfm2ss' and 'ss2tfm.' For example, let us consider the transfer function

\[
\frac{s - 1}{s - 2}
\]

We first form the row vector 'denom,' which holds the coefficients of the common denominator of the transfer function. The second matrix, 'num,' holds all the numerator coefficients laid out in the same positions as they appear in the matrix form, adding padding zeros for terms with zero coefficients. For example,

\[
\begin{pmatrix}
-6 & s & 2 \\
(s + 1)(s + 2)
\end{pmatrix}
\]

In addition to the preceding description of the language, the following features make the package a significant departure from existing CACSD tools.

Design Methodology

The package is a genuine design tool as well as an analysis tool for control system designers, and the design methods it has implemented are summarized next.

Stable Factorization Design Approach

The design environment incorporates the Stable Factorization design technique developed by Vidyasagar [5]. The Stable Factorization Approach is of relatively recent origin and is based on the idea of representing an unstable system as a 'ratio' of two stable systems. Although the basic idea was first proposed in a paper published in 1972 [6], the applications to control system synthesis (as opposed to analysis, which was the focus of [6]) date from about 1980. Despite its recent origin, the method quickly has gained wide acceptance, owing to its potential for solving many practical design problems in a systematic manner. Space limitations do not permit a detailed discussion of the method, and the reader is referred to [5]. However, for the purposes of the present discussion, it suffices to state that a broad class of controller design problems can be reduced to the following canonical optimization problem.

Given matrices $F, G, H, A, B, C$ whose elements are all stable rational functions, together with constants $\alpha_i$, find a stable rational matrix $R$ of appropriate dimensions that minimizes the cost functional

\[ J = F - GRH \]
subject to the constraints
\[ |A - BRC| \leq \alpha_\epsilon \nu_1 \]

The class of problems that can be formulated as a canonical problem of the preceding type includes all problems involving performance specifications on tracking, disturbance rejection, robustness, and nonsaturation.

**Systematic Design Approach** The Systematic Design Approach developed by Pang and MacFarlane [7] belongs to the class of frequency-response design techniques. The design approach has been represented in three hierarchically organized design levels. The first level of design, using the Simple Design Technique, results in a simple proportional-plus-integral controller. At the second level, a more complicated controller, which allows for gain and phase manipulation in an intermediate frequency region, may be obtained. This new technique, called Reverse Frame Alignment, is a development of the Characteristic Locus Design Method and is considered to give a more systematic design procedure. At the top level, an observer-based controller is used, which provides a means of accessing the information of the states of the system. These three techniques give a powerful, systematic, and intuitively appealing approach to the design of linear multivariable systems.

**Other Design Methods** The program allows users to add procedures and macros that implement their preferred design methods. In this way, they will not be restricted to the Stable Factorization and Systematic Design Approaches and can make the most of their past experience and expertise by implementing other methods of their choice.

**Design Analysis**

Frequency-response methods of analysis are provided in the package. For example, the poles and zeros of a system can be obtained. Indicators such as the generalized Nyquist diagrams and Bode plots of principal gains (derived from singular values) and characteristic gains (derived from eigenvalues) can be displayed. Indicators such as the generalized Nyquist diagrams and Bode plots of principal gains (derived from singular values) and characteristic gains (derived from eigenvalues) can be displayed. Indicators for the assessment of stability, performance, and robustness of a linear multivariable system are also included. In particular, the systematic use of a complete set of indicators, called primary indicators, is basic to the Systematic Design Approach. The primary indicators can be displayed with the following user-defined function. The command "unpack" partitions a packed matrix into the four state-space matrices.

To obtain the primary indicators of the system "state_space_sys," the user can define the function and run it as follows:

```matlab
< > define 'primary'
< > primary(state_space_sys)
```

The primary indicators of the system are shown in Fig. 1.

**Knowledge-Based Expert Systems Facilities**

The incorporation of expert system techniques in a design package can provide a more supportive and high-level design environment for the user. Knowledge of various design techniques and control system design in general can be incorporated within the package. Hence, for nonexpert users, there will be more "dynamic" help on the use of the design package, and users can be guided through a design process. Information that is relevant at each stage of a design will be given to the user. For the expert users, designs may be carried out more productively and efficiently.

The preceding facility is developed by extending the parser of the package to handle the additional rule-based syntax required. Knowledge is represented in rules, facts, and objects. A user can also develop a personal knowledge base following a very simple syntax, such as plain English, when writing the knowledge base. The packed matrix data structure in SFPACK has been extended further for the above-mentioned purpose. The following example illustrates the way knowledge is represented.

```matlab
< > state_space_sys = "process system model"
< > fact "process system model" is "open-loop stability unknown"
< > fact "system poles" is "open-loop stability unknown"
< > fact "system poles" is "obtained"
< > fact "the aim is "test stability"
```

The variable "state_space_sys" is now associated with the identifier "process system model." The second statement shown is an assertion of a new fact to the knowledge base. The symbolic value of the identifier is now "open-loop stability unknown." The following rule is used to illustrate the rule syntax of the knowledge-based development facility. It can be one of the results in a built-in knowledge base.

![Fig. 1. Primary system indicators.](image-url)
the design package. The development of FORTRAN subroutines from LINPACK and the parser are the same as those in MATLAB; hence, the new package has the same matrices are entered and manipulated in a natural and straightforward fashion using a simple command language. Another advantage for developing a user interface similar to that of MATLAB is that a new user of SFPACK will have no problem using the package, because the MATLAB syntax is very popular and widely known. As noted in [9], most new developments of CACSD packages that started after 1981 are based on MATLAB in some way or another.

The new SFPACK parser is produced using the Unix tool YACC, which generates a parser when given the grammar rules of a language. The parser that is generated consists of a finite-state machine with a stack. YACC has been an invaluable tool during the development of SFPACK, because it allowed rapid prototyping in the software design environment. It has served as an aid in analysis and design of the software. For example, when developing the expert system development facility of SFPACK, the requirements and specifications were defined first. A model or prototype of the system was then constructed using YACC. The behavior of the prototype was then evaluated against its desired behavior. If the prototype does not execute properly, the grammar rule specification to YACC will be modified and the iterative process continues until the prototype has captured all the critical aspects of the envisioned system. Thus, SFPACK has gone through an evolutionary development and the software package is developed incrementally.

The new command language of SFPACK is a MATLAB-like language. It provides all the basic commands available in MATLAB as well as additional commands for carrying out the Stable Factorization and other design techniques such as the Systematic Design Approach. It allows the user to augment the abilities of the package by writing one's own routines, which can be carried out in two ways: user-defined functions and procedures. Knowledge bases on the use of various design techniques or control theory can also be loaded into the control environment of SFPACK. These knowledge sources are composed of production rules and facts for a particular domain area, and they are included in the control environment to provide guidance to the users of the package. The user can also write his/her own knowledge bases following a particular syntax of production rules. Thus, the user can store experience or knowledge of the package and control engineering in general, and then transfer this valuable expertise to others. Good graphics facilities are also included.

Portability and Availability
The package now runs on Unix, either BSD 4.3 on a VAX or SUN OS 4.0 on a Sun workstation. The plots can be obtained in either regis or tektronix mode on a vt240 terminal. On a Sun workstation or other terminals, the plots can be obtained in the tektronix mode.

Our desire is to make the package widely circulated and used in the academic and industrial communities. A nominal charge will be made to cover the cost of the cartridge tape, documentation, and additional administrative cost. The interested reader should contact Dr. G. Pang for more information (e-mail address: pang@watale.waterloo.edu or unet!watmath!watale!pang).

Conclusions
Our objective is to develop an interactive environment for computer-aided control system design (CACSD) that can be used to design controllers for linear multivariable systems. The language of the environment can support design techniques based on the Stable Factorization (SF) and Systematic Design Approaches, but will not be restricted to these methods. Users will also be able to write their own procedures, and this will enhance the applicability of the software. The essential features intended for the language to possess are as follows:

1. It should be an interactive terminal language, able to execute commands as soon as they have been entered, after which control is returned to the user.
2. It must allow one to write procedures and run them immediately, without the need to load and link the entire design program.
3. It should include a fixed set of primitives rich enough to implement a wide range of established design methods, especially including the Stable Factorization and Systematic Design Approaches.

We believe that the CACSD environment developed has several advantages over existing design tools. Current design programs are, to a large extent, either purely analysis programs enabling a designer to evaluate the suitability of a controller that has been obtained by means unspecified or, at the very best, are limited to tuning the parameters or variables within an a priori fixed controller structure. These design environments give users very little help in deciding on the controller structure itself, and designers are thus forced to adopt a cumbersome trial-and-error procedure in which the structure of the controller is obtained largely by guesswork. The Stable Factorization design method, on the other hand, is based on a parameterization of all possible stabilizing controllers. Of course, design based on the SF approach is still iterative and interactive, but the interactions between the designer and the program that implements the SF approach take place at a very high level. For example, once a controller that meets all of the given design requirements has been generated, the user might decide to tighten up some of the spec-
ifications, or trade some specifications for others, and then run the design procedure again. Finally, the ability of users to enhance the basic set of design methods with specialized procedures written in the language of the environment will permit them to take advantage of their past experience and expertise by supplementing the SF approach with design methods with which they are familiar, and to incorporate recent advances in design methodology.

Acknowledgments

The work reported herein was supported by the Information Technology Research Centre, a Centre of Excellence set up by the Ontario Government in Canada.

References

[10] Grantham K. H. Pang is an Assistant Professor at the Department of Electrical and Computer Engineering, University of Waterloo. He received the Ph.D. degree from the University of Cambridge in 1986 for research in multivariable control system design. His research interests include expert systems for control system design, intelligent control, control theory, and computer-aided design. Dr. Pang has been a Scholar and Fellow of The Croucher Foundation (United Kingdom/Hong Kong).

Mathukumalli Vidyasagar received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Wisconsin in 1965, 1967, and 1969, respectively. After receiving the Ph.D. degree, he taught at Marquette University, Milwaukee, Wisconsin, and for 10 years at Concordia University, Montreal, Canada. In 1980, he joined the University of Waterloo, Canada, as a Professor of Electrical Engineering. Since mid-1989, he has been the Director of the Center for Artificial Intelligence and Robotics, Bangalore, India. He has written over 100 technical papers in various journals, and he is the author or coauthor of five books. His current research interests are in robotics and neural nets. Dr. Vidyasagar was awarded the E.W.R. Steacie Memorial Fellowship by the Natural Sciences and Engineering Research Council of Canada for the period 1981-1983, the Frederick Emmons Terman Award by the American Society of Engineering Education in 1984, and the Prize Paper Award of the Japanese Society of Instrumentation and Control Engineers in 1989.

Andrew J. Heunis received the B.Sc. in electrical engineering from the University of the Witwatersrand, Johannesburg, and the M.Sc. and Ph.D. degrees from Imperial College, London. Since 1985, he has been with the Department of Electrical Engineering at the University of Waterloo, Ontario.