DELIGHT. MIMO: An Interactive, Optimization-Based Multivariable Control System Design Package

E. Polak, P. Siegel, T. Wuu and W. T. Nye
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720

D. Q. Mayne
Department of Electrical Engineering, Imperial College, London, SW7 2BT, U.K.

Abstract

This paper describes an interactive, optimization-based multivariable control system design package which is currently under development at the University of California, Berkeley. The package will combine a number of subroutines from the Imperial College Multivariable Design System and the Kingston Polytechnic SLICE library with DELIGHT, the University of California, Berkeley, general purpose, interactive, optimization-based CAD system.

1. Introduction

Optimization, either heuristic or algorithmic, is an integral part of engineering design. Heuristics are most frequently used in selecting a system configuration, while algorithmic optimization is used to determine parameter values which satisfy design specifications or optimize a performance function. A new generation of semi-infinite optimization algorithms, see e.g., [G1, P1, P2] enable the multivariable control system designer to satisfy complex specifications, some involving constraints on time responses, others involving constraints on closed loop system eigenvalues or on frequency dependent singular values of various system matrices, see e.g., [D3, T1, Z1]. For best results, these new algorithms must be implemented in an interactive computing environment.

The DELIGHT system [N1] was conceived as an interactive computing environment for multidisciplinary, optimization-based engineering design. It incorporates a high level language (RATTLE) which simplifies the programming of algorithms; standard FORTRAN numerical analysis programs; a modular RATTLE optimization library; as well as highly flexible color graphics and interaction capabilities. DELIGHT.MIMO is a multivariable control system design package which was constructed by incorporating into the DELIGHT system routines for control system definition, response evaluation and graphical display. It permits the designer to use interactively both modern multivariable system design tools as well as semi-infinite optimization algorithms.

Section 2 of this paper describes the "basic" DELIGHT system, section 3 presents the control system design specific enhancements, while section 4 illustrates the use of DELIGHT.MIMO in the optimization of a multivariable control system design, by means of an example.

2. The DELIGHT System

The DELIGHT system [N1] is an interactive computing environment which was developed by W. T. Nye, E. Polak, A. Sangiovanni Vincentelli and A. L. Tits as part of a broad project dedicated to optimization-based computer-aided design. It provides the user with a number of features which are present in the UNIX C-shell [U1] as well as with a number of others that facilitate interactive optimization-based design and optimization algorithm development. It can be extended into an application specific design package by the addition of system definition and simulation programs. DELIGHT.MIMO is such a package.

The major features of DELIGHT are (i) simple command and algorithm execution, (ii) a capability to rescale or modify either the design problem being solved or the optimization algorithm being used, without recompilation and reloading of the programs or reinitialization of the algorithm, (iii) powerful, terminal independent color graphics which can be used to display information stored in arrays, in a number of ways, in colors and windows designated by the user, (iv) a high level language which permits the writing of compact computer programs closely resembling the mathematical description of the algorithms being implemented, eliminating most of the usual coding errors and shortening programming time, (v) provisions which facilitate the addition of new built-in FORTRAN functions, simulation subroutines, utilities, or other FORTRAN application dependent features.

* This research was supported by the National Science Foundation under grant ECS-79-13148 and the U. K. Science and Engineering Research Council.
constructs include "while", "repeat-until", "if-then-else", etc. It allows matrix and graphical commands, eliminates the need for dimension statements, common block declarations and time consuming load/linkages.

It is possible in RATTLE to create new language constructs or new commands from existing ones by means of macros and defines. Macros are used to call highly complex FORTRAN procedures by means of very simple commands. For example, to solve a linear program one uses the following RATTLE code, which uses the macro 'lp':

\[
lp z = \arg \min \{ c^* x \mid x > 0, \\
\quad x < d, A x = b \}
\]

where the array \( z \) is assigned the minimizing value of \( x \), and the matrix \( A \) and vectors \( b, c, d \) have been defined previously. Macros not only enhance readability, they also relieve the programmer of the need to create work arrays and to master the other requirements of the library routines. Both binary and unary matrix computations are carried out by means of the \( \text{matop} \) macro. The table on this page gives a sample of operations available to the user through macros.

An important use of defines is in the creation of simple commands for invoking complex, terminal independent graphics procedures that are written in RATTLE. For example, there exists a define which allows the command "window name" to be substituted for the specification of the particular set of world coordinates, and viewport coordinates which are associated with the window [N2].

The RATTLE language permits incremental program development [W1], so that one can execute by just typing it in, a single statement, a procedure, or a section of an algorithm, without having to write and load/link a whole program. The following is a complete RATTLE statement, implementing Newton's method, which would execute when the closing brace and carriage return are typed in:

\[
\text{while } (f(x)) > \text{eps} \{ x = x - f(x)/\text{derf}(x) \\
\quad \quad \text{print } x \}
\]

An important RATTLE feature, both from the designer's and algorithm developer's point of view, is that execution of a program can be interrupted by the user or by the program and later resumed after modifying variable values or even recompiling an entire subprocedure. While execution is suspended, the values of both global and local variables can be displayed and modified by appropriate commands.

The nice relationship between the mathematical description of an algorithm and its implementation in RATTLE is illustrated by the following code implementing the Armijo gradient method [A1]:

```rattle
procedure Armijo {
\text{repeat }
\quad \text{evaluate } h = \text{grad } (X[\text{Iter}]) \\
\quad \text{lambda} = \text{step } (X[\text{Iter}], h) \\
\quad \text{update } X[\text{Iter} + 1] = X[\text{Iter}] \
\quad \quad - \text{lambda } h \\
\text{Iter} = \text{Iter} + 1 \\
\text{output }
\}
\text{forever}
```

where "evaluate" and "update" are defines for calling the appropriate sub-procedures, step is a function which computes a step size by the Armijo rule, "output" is a procedure that produces a display, while \( X[\text{Iter}] \) is a vector in a sequence whose last \( k \) iterates (typically 20) are stored.

The DELIGHT system contains an ever growing library of RATTLE routines implementing algorithms for solving unconstrained and constrained, both ordinary and semi-infinite optimization problems. This library is organized to exploit the natural modularity of modern optimization algorithms which, in the simplest case, can be assembled from search-direction, step-size and update subalgorithms. In turn, search-direction subalgorithms can be constructed from subprocedures which compute the gradients to be used for search direction construction and from linear or quadratic programs. Similarly, step-size subalgorithms can be built up from constrained and unconstrained step-size blocks. The user may interactively explore algorithm component and output options and select those that suit his needs. Substitutions from the options list may be made at any time, including when execution has been interrupted in the middle of an optimization run (by depressing the break key).

The use of the RATTLE optimization library has been highly mechanized. Thus, the user defines his problem by inserting appropriate lines in a setup file, in a cost file, in a constraint file and in an initial data file. The setup file contains information on the nature and number of constraints to be used and the dimension of the design vector. The cost and constraint files contain code for evaluating the cost and constraint functions and their gradients, which may involve calls to FORTRAN simulation subroutines. Once the problem files have been created, the user may select an algorithm from the optimization library and link it to his problem by a command of the form `solve probname using algoname`.

### 3. Control System Design Aids.

As already mentioned in the preceding section, the "basic" DELIGHT system includes the LINPACK [D1] linear algebra and Harwell [H1] linear and quadratic programming subroutines which can be used for all relevant MIMO design computations by means of high level macros. DELIGHT MIMO was formed by (i) adding to the DELIGHT system an overall control system assembly and graphical display program, (ii) interfacing a number of FORTRAN subroutines for MIMO design from the Multivariable Design System (MDS) [S2], via the built-in function mechanism provided by DELIGHT, and (iii) adding a number of RATTLE routines for interaction and graphical display of results. The control system specific design aids can be grouped into two categories: (a) aids for data entry and manipulation and (b) aids for producing an initial design and evaluating system responses.
Multivariable control systems may be entered in either state-space or transfer function form. State-space descriptions are limited to 40 states, 10 inputs and 10 outputs. Transfer function matrices can be of maximum dimension $10 \times 10$, where each entry of the transfer function matrix may contain rational functions of maximum order 20. Systems may contain up to 20 blocks; each block may contain parameters to be optimized.

To enter a control system, one must first enter the individual blocks, either from a file or interactively, using the `entblock` command. One may verify if a block has been entered correctly by means of the `check` command. Editing facilities are available for modifying existing blocks. Parameters to be optimized are specified by means of the command `entparam`. Initial values for these parameters are entered via the `entblock` command. The `convert` command can be used to convert a state-space description into transfer function form and vice versa. Descriptions can be saved in files by means of the `save` command.

Once the individual blocks have been entered, their interconnection is specified in terms of a reverse polish list. Three operations may be used to connect blocks: * cascade connection; + feedforward connection; < feedback connection. This method of entry allows systems of arbitrary complexity to be specified. See Fig. 1 for an example. The `bd` command may be used to display graphically the block diagram of the system (computed from the reverse polish list) in order to verify that the control system was correctly entered.

The control system definition utility is currently undergoing revision. The Polish list method of system interconnection definition is being replaced by a menu-driven graphical input facility. The designer will be able to enter the block diagram of the system by "drawing" it on the screen with the aid of a graphics tablet and stylus. It will be possible to edit block diagrams graphically. Input waveforms will be entered either by tracing the waveform on the tablet or by selecting from the menu.

Since optimization requires derivatives of responses with respect to design parameters, we are constructing a symbolic differentiator, as described in [B1]. This requires that control system components containing design parameters be entered only in state-space form, with the elements of the matrices described as quotients of multivariable polynomials in the design variables. The remaining blocks can be entered in state-space form, as matrix transfer functions, or as polynomial matrix fractions. The time-domain responses which the symbolic differentiator can handle include responses to polynomial, sinusoidal and exponential inputs. Frequency-domain responses handled by the symbolic differentiator include distinct singular values of transfer function matrices.

The MDS system [SZ] and the SLICE library [D2] contain most of the commonly used subroutines for modern control system design. A number of these have already been incorporated into DELIGHT.MIMO. Additional ones will be incorporated as necessary.

\[
\begin{align*}
Q(x,s) &= \frac{1}{3(s+2)} \begin{bmatrix} 3s^2 + 9s + 8 \\ -2(s+1) \\ -s^2 \end{bmatrix} \\
P(s) &= \frac{1}{(s+2)(s+3)} \begin{bmatrix} 3s + 8 & 2s^2 + 6s + 2 \\ s^2 + 6s + 2 & 3s^2 + 7s + 8 \end{bmatrix} \\
f(x) &= -(x_2/x_1 + x_4/x_3)
\end{align*}
\]
Control system design by means of semi-infinite optimization is a four-stage process. The first stage consists of sorting the system performance requirements into “soft” and “hard” categories. The “soft” requirements are modeled as a composite cost $J(x)$, to be minimized, while the “hard” requirements are expressed as simple inequalities of the form

$$g^j(x) \leq 0, \quad j = 1, 2, \ldots, 1, \quad (1)$$

or as semi-infinite inequalities of the form

$$f^k(x, w_k) \leq 0 \quad \text{for all}$$

$$w_k \in [w_k', w_k''], \quad k = 1, 2, \ldots, m, \quad (2)$$

where $x$ is the design vector to be adjusted by optimization algorithm. The third stage consists of computing an initial value for the design vector $x$ by means of some “classical” method, such as LQR [S1], or multivariable root loci [M1], which usually results in a control system that fails to satisfy a number of the “hard” specifications. The last stage involves the use of a semi-infinite optimization algorithm to adjust the parameter $x$ so that all the “hard” constraints are satisfied and the cost minimized, or at least reduced.

For example, consider the system in Fig. 2, where the $z_i$ are the design parameters. The structure and initial values for the controller are designed by the method in [G2]. The design objective is to maximize the bandwidth of the closed loop system subject to constraints on the closed loop step response, as in Fig. 3, and constraints on the input amplitude of the plant, which are expressed as bounds on the singular values of the matrix $Q$, as shown in Fig. 4. The routine opt init helps the designer to transcribe interactively these requirements into functions and subroutines conforming to the format in (1) and (2). The result is a DELIGHT format problem called optdesign.

Next an algorithm is selected and coupled to the problem by using the solve command. For example, solve optdesign using Apolwar [P1] nondifferentiable optimization algorithm from the optimization library. This algorithm consists of a phase I phase II search direction finding subprocedure, an Armijo type step size subprocedure and an update subprocedure. The algorithm contains a number of parameters which need to be adjusted for efficient coupling of the problem and algorithm. This adjustment is carried out interactively, with the aid of graphics, as follows. First, the search direction procedure is executed, by means of the command $dir$. At this point the algorithm is suspended, allowing the user to change the problem scaling parameters (push factors). To decide whether this is necessary, the user types in Gangles to display the gradient clock, see Fig. 5, which show the angles between the search direction and the
gradients of the cost and active constraints. In Fig. 5 the gradient clock indicates that the gradients of inequality constraints 3 and 4 are almost perpendicular to the search direction $h$ due to poor initial scaling of the design problem and, possibly, inadequate precision in the search direction calculation. The user attempts to remedy this by changing the values of the precision parameter. To verify that the problem has been eliminated, the user repeats the dir and Gangles commands, to obtain Fig. 6. Since this is satisfactory, the user decides to examine the step-size parameters. For this purpose, the user types in the command Garnijo to obtain Fig. 7 which shows all the information that governs step size computation and active constraint selection. Since the algorithm is not spending an excessive amount of time in the step-size loop, the user decides to carry out five iterations of the algorithm: he types in reset carriage return run 5. When these are completed, the user examines the results by typing in response green 0 0 1, svplt green 0 1, response sky 0 5 2, svplt sky 0 2, to display the appropriate step and singular value responses in the colors indicated. These commands produce the display shown in Fig. 8, in which the colors were converted to dash patterns. This figure shows in solid curves the responses corresponding to the initial design, while the dashed curves correspond to the design parameters at the end of the 5th iteration. Note that after the
5th iteration the design satisfies all of the constraints (indicated in the figure by solid lines). The box marked paller gives the color in which the results of each iteration were plotted.

5. Conclusion.

The DELIGHT.MIMO control system design package is intended both as a practical design tool and as a test bed for concepts to be used in interactive control system design. It is hoped that it will prove to be highly obsolescence-resistant and that it will eventually evolve into a comprehensive facility.

6. References


