

Copyright © 1982, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

DELIGHT-MIMO: AN INTERACTIVE, OPTIMIZATION-BASED
MULTIVARIABLE CONTROL SYSTEM DESIGN PACKAGE

by

D.Q. Mayne, W.T. Nye, E. Polak, P. Siegel, T. Wu

Memorandum No. UCB/ERL M82/53

1 July 1982

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

DELIGHT-MIMO: AN INTERACTIVE, OPTIMIZATION-BASED
MULTIVARIABLE CONTROL SYSTEM DESIGN PACKAGE. *

by

D. Q. Mayne**, W. T. Nye***, E. Polak***, A. Sangiovanni Vincentelli***,
P. Siegel***, A.L. Tits***, and T. Wu***.

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.

* This research was supported by the National Science Foundation under grant ECS-79-13148 and the U. K. Science Research Council.

** Department of Electrical Engineering, Imperial College, London, SW7 2BT, U.K.

**** Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca. 94720.

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. In response to engineering needs, a new generation of semi-infinite optimization algorithms has been developed [G1, G2, M2, P1, P2, T2]. In the context of multivariable control system design, these algorithms enable the 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. [L1, S1, S2, 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 system for multidisciplinary optimization-based engineering design. It incorporates a high level language, RATTLE, whose main function is to simplify the programming of algorithms, graphics and interaction. Since RATTLE executes somewhat slower than FORTRAN, for maximum efficiency, the DELIGHT system must be used as a "hybrid", with the optimization algorithms, graphics, interaction and data manipulation procedures written in RATTLE, and the simulation, linear and quadratic programming and numerical linear algebra subroutines written in FORTRAN. Since these FORTRAN programs are normally developed independently, it is clear that this hybridization is absolutely essential to the success of a design system such as DELIGHT.

The DELIGHT system enables the designer to monitor, interrupt, diagnose, modify and restart a computation as it progresses while its powerful graphics make it easy to form highly informative color displays. Consequently, the designer can efficiently guide an optimization run by adjusting algorithm or problem parameters at appropriate times during execution. The flexibility of the DELIGHT system, coupled with the interactive system definition and design aids provided by an applications package, reduces substantially not only the optimization computing time, but also the overall time needed to carry out a design.

The first version of the DELIGHT-MIMO system for multivariable control system design was constructed by incorporating a number of subroutines from the Imperial College Multivariable Design System (MDS) [S4] to the DELIGHT system. These subroutines enable the designer to define system components and interconnections, to evaluate time and frequency responses, and to produce an initial design by "classical" techniques. Optimization is carried out by a RATTLE implementation of the Polak-Wardi algorithm [P2] which permits constraints both on time responses and on singular values over a frequency range. At the present time, DELIGHT-MIMO is being substantially revised by the introduction of a graphical system component interconnection capability, the replacement of the MDS subroutines by numerically more robust subroutines from the SLICE [D2] library, and by the addition of a number of utility programs. These utilities implement various modern design techniques. The designer will have the option of using these utilities either independently from the optimization algorithms provided, or as a means for obtaining an initial design for a semi-infinite optimization program.

Thus, although the DELIGHT-MIMO system is primarily intended for optimization-based design, it can be used as a universal design tool and hence should be of broad usefulness in control system design.

2. CONTROL SYSTEM DESIGN VIA SEMI-INFINITE OPTIMIZATION.

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 $f(x)$, to be minimized, while the "hard" requirements are expressed as simple inequalities of the form

$$g^j(x) \leq 0, \text{ for } j = 1, 2, \dots, l, \quad (1)$$

or as semi-infinite inequalities of the form

$$\phi^k(x, w_k) \leq 0 \text{ for all } w_k \in W_k, k = 1, 2, \dots, m, \quad (2)$$

where $f: R^n \rightarrow R$ and the $g^j: R^n \rightarrow R$ are continuously differentiable; the $\phi^k: R^n \rightarrow R \rightarrow R$ are at least locally Lipschitz; the $W_k = [w_k', w_k'']$ are compact intervals, and x is the design vector to be introduced in the second stage. The second stage consists of defining a system configuration and a controller structure containing x as a vector of n design parameters to be adjusted by an 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 [A2] or multivariable root loci [M1], which usually results in a control system that fails to satisfy a

number of the "hard" specifications and is in no sense optimal. 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.

Typical examples of constraints of the form (1) are those expressing bounds on gains and those restricting the eigenvalues of the closed loop system to a specified region in the s -plane (the eigenvalues of the closed loop system can be assumed to remain distinct during the design process). Thus, if $e_i(x)$, $i = 1, 2, \dots, L$, are the eigenvalues of the closed loop system, we may require that they all lie in a parabolic region in the s -plane defined by

$$\operatorname{re}[e_i(x)] + a(\operatorname{im}[e_i(x)])^2 - b \leq 0, \text{ for } i = 1, \dots, L, \quad (3)$$

with $a > 0$, $b \geq 0$.

Typical examples of constraints of the form (2) are those confining the step responses $y^i(x, t)$, $i = 1, 2, \dots, K$ of the closed loop system to specified envelopes:

$$b_l^i(t) \leq y^i(x, t) \leq b_u^i(t) \text{ for } t \in [0, T], i=1, \dots, K, \quad (4)$$

where the $b_l^i(t)$ and $b_u^i(t)$ are piecewise continuous functions. Another important example of constraints of the form (2) are bounds on the smallest singular value $s(x, w)$ of the return difference matrix $F(x, w)$, imposed to reduce sensitivity to parameter variations. These have the form

$$b(w) - s(x, w) \leq 0 \text{ for all } w \in [w', w'']. \quad (5)$$

where $b(w)$ is a piecewise continuous function.

A possible role for the cost function could be to express the desire to keep small the energy being pumped into the system, during a specified maneuver. It would then take the form of an integral:

$$f(x) = \int_0^T u(x,t)dt \quad (6)$$

Hopefully, these examples will suffice to give the reader an idea of the versatility of semi-infinite inequalities in expressing complex system performance requirements.

To give the reader a feel for the complexity of the algorithms needed to solve the resulting semi-infinite optimization problems, we now describe the Polak-Wardi algorithm [P2]. For the sake of simplifying notation, we will assume that we have exactly one constraint in each category, i.e. that we wish to solve a problem of the form:

$$\begin{aligned} \min\{f(x) \mid g(x) \leq 0; y(x,t) - b(t) \leq 0 \text{ for all } t \in [0,T]; \\ a(w) - s(x,w) \leq 0 \text{ for all } w \in [w',w'']\}. \end{aligned} \quad (7)$$

We assume that the cost $f(x)$, the constraint $g(x)$, the response $y(x,t)$ and the $m \times m$ return difference matrix $F(x,w)$ are all continuously differentiable in $x \in R^n$. Let

$$\begin{aligned} M(x) = \max\{0, g(x); y(x,t) - b(t), t \in [0,T]; \\ a(w) - s(x,w), w \in [w',w'']\} \end{aligned} \quad (8)$$

and for any $\epsilon \geq 0$, let

$$d_{\epsilon}(x) \triangleq \left. \begin{array}{l} 1 \text{ if } g(x) \geq M(x) - \epsilon \\ \infty \text{ otherwise} \end{array} \right\} \quad (9)$$

$$T_{\epsilon}(x) \triangleq \{t \in [0, T] \mid y(x, t) - b(t) \geq M(x) - \epsilon\} \quad (10)$$

$$W_{\epsilon}(x) \triangleq \{w \in [w', w''] \mid a(w) - s(x, w) \geq M(x) - \epsilon\} \quad (11)$$

$$G_{\epsilon}(x, w) \triangleq \text{co}\{v \in \mathbb{R}^n \mid$$

$$v^i = \langle U_{\epsilon}(x, w)z, [dF(x, w)/dx^i]U_{\epsilon}(x, w)z \rangle, \quad i = 1, \dots, n;$$

$$\|z\| = 1\}$$

(12)

where $U_{\epsilon}(x, w)$ is an orthonormal matrix of singular vectors corresponding to the singular values $s_j(x, w)$, $j = 1, 2, \dots, m$, of the $m \times m$ matrix $F(x, w)$, which satisfy

$$a(w) - s_j(x, w) \geq M(x) - \epsilon, \quad j = 1, 2, \dots, m. \quad (13)$$

Next, for any $\epsilon > 0$, we define

the phase-I search ϵ -direction

$$h_{M, \epsilon}(x) \triangleq \text{argmin}\{\|h\| \mid h \in \text{co}\{d_{\epsilon}(x)\nabla g(x); G_{\epsilon}(x, w), w \in W_{\epsilon}(x); \nabla y(x, t), t \in T_{\epsilon}(x)\}\} \quad (14a)$$

and the phase-II ϵ -search direction

$$h_{f, \epsilon}(x) \triangleq \text{argmin}\{\|h\| \mid h \in \text{co}\{\nabla f(x); d_{\epsilon}(x)\nabla g(x); G_{\epsilon}(x, w), w \in W_{\epsilon}(x); \nabla y(x, t), t \in T_{\epsilon}(x)\}\} \quad (14b)$$

and the combined phase-I phase-II ϵ -search direction

$$h_{\epsilon}(x) \triangleq [\exp(-M(x))h_{f, \epsilon}(x) + [1 - \exp(-M(x))]h_{M, \epsilon}(x)]. \quad (14c)$$

With $\epsilon_0, \delta > 0$ and $\nu \in (0, 1)$ given, let

$$E \triangleq \{0, \epsilon_0, \forall \epsilon_0, \forall^2 \epsilon_0, \dots\} \quad (15a)$$

and let

$$\epsilon(x) \triangleq \max\{\epsilon \in E \mid \max\{(\exp(-M(x)) \|h_{f,\epsilon}(x)\|)^2, \\ ([1-M(x)\exp(-M(x))] \|h_{M,\epsilon}(x)\|)^2\} \geq \epsilon\} \quad (15b)$$

We now have all the elements needed to state the Polak-Wardi algorithm [P2].

ALGORITHM

Parameters: $\alpha, \beta, \nu \in (0,1), \epsilon_0, \delta > 0$.

Data: x_0 .

Step 0: Set $i = 0$.

Step 1: Compute $\epsilon(x_i)$ and set $h_i = h_{\epsilon(x_i)}(x_i)$.

Step 2:

If $M(x) > 0$, compute the largest step size $s_i = \beta^{k_i}$,

$k_i \in \{0,1,2,\dots\}$, such that

$$M(x_i - s_i h_i) - M(x_i) < s_i \alpha \|h_i\|^2. \quad (16a)$$

Else, compute the largest step size $s = \beta^{k_i}$,

$k_i \in \{0,1,2,\dots\}$, such that

$$f(x_i - s_i h) - f(x_i) \leq s_i \alpha \|h_i\|^2 \quad (16b)$$

and

$$M(x_i - s_i h_i) \leq 0. \quad (16c)$$

Step 3: Set $x_{i+1} = x_i + s_i h_i$, set $i = i + 1$ and go to

step 1. □

The computation of the argmins in (14a), (14b) is performed by means of a proximity algorithm, such as the one described in Chapter 5 of [P3], which makes use of the fact that given a vector v , a point of tangency $z(v)$ in $G_\epsilon(x)$ defined by

$$z(v) = \operatorname{argmin}\{\langle v, y \rangle \mid y \in G_\epsilon(x)\} \quad (17)$$

can be computed by means of a single singular value decomposition. The truncation of the computation carried out according to the theory in Appendix A of [P3].

3. CONTROL SYSTEM DESIGN AIDS.

Currently, the control system design aids include (i) the LINPACK [D1] linear algebra and Harwell [H1] linear and quadratic programming subroutines (these are part of the DELIGHT system), (ii) an overall control system assembly and graphical display program, and (iii) subroutines from the Multivariable Design System (MDS) [S4] which was produced by the Computer Aided Design Group in the Department of Computing and Control at Imperial College. At present these design aids are being updated and extended by additions from the SLICE library [D2] developed at the Kingston Polytechnic, and other subroutines as they become available. 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.

At present system components can be entered by means of the create command, either interactively or from a file, in state-space or transfer function matrix form. The system coefficients can be entered either as numbers or as design variables. It is possible to convert a state-space description into a transfer function matrix and vice-versa. Overall system assembly is accomplished by defining a reverse polish list in which the variables are system component names and the three allowable operations are * (series connection), + (parallel connection) and < (feedback connection). A graphical command converts the polish list into a block diagram for verification. To display a component description the check

command is used. To edit a component description, one uses the modify command. To change a component description from one of the allowed types into another the convert command is used. The descriptions can be saved in a file by means of the save command.

The control system definition utility is currently undergoing revision. In the new version, control system components will be entered in any one of the following forms: (i) state-space, with the elements of the matrices described as quotients of multivariable polynomials in the design variables, (ii) matrix transfer function, with the coefficients of powers of s described as quotients of multivariable polynomials in the design variables, and (iii) polynomial matrix fractions, with the coefficients of powers of s described as quotients of multivariable polynomials in the design variables.

Once the various components have been defined, their interconnection will be specified by means of the link command which will enable the user to specify the interconnection equation either graphically or by means of a table which can be stored on disk. Graphical or tabular editing of a system interconnection will be carried out by means of the relink command.

The initial design aids will include commands for computing the greatest common divisor of two polynomials, for permuting rows or columns of a matrix, for computing the roots of a polynomial, for computing the Pade approximant to a time delay, for determining whether a state-space model is controllable and/or observable, for computing the transfer function matrix resulting from closing a single loop, for computing the inverse D.C. matrix of a transfer function matrix, for

computing and plotting Nyquist diagrams for a scalar or matrix transfer function, for plotting Gershgorin and Ostrowski circles on a Nyquist diagram, for plotting root locus diagrams, for computing the Smith normal form of a polynomial matrix or the Smith Macmillan form of a rational matrix, for computing multivariable root loci, for computing a minimal-state space realization, and for LQR and LQG design.

The simulation aides will include polynomial, sinusoidal and exponential input response evaluation, matrix frequency response evaluation and singular value evaluation.

4. THE DELIGHT SYSTEM.

The DELIGHT system was developed by W. T. Nye, E. Polak, A. Sangiovanni Vincentelli and A. L. Tits [N1] as part of a broad project dedicated to optimization-based computer-aided design. It was conceived so as to facilitate optimization-based engineering design as well as optimization algorithm development. For the unsophisticated designer it provides simple command and algorithm execution. For the advanced designer it provides features which make it easy 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. In addition, it has powerful, highly flexible graphics which can be used both to display the usual (time domain, frequency domain, s-plane, etc) plots for a control system as well as the more arcane information which facilitates optimization algorithm adjustment. The optimization algorithm expert is given a high level language which permits him to write compact computer programs that bear a close resemblance to the mathematical description of the

algorithms being implemented. Consequently, most of the usual coding errors are eliminated and the programming time is shortened tremendously. Finally, since the DELIGHT system is intended to be used as a basis for many discipline specific applications packages, allowances have been made to facilitate the addition of new built-in FORTRAN functions, simulation subroutines, utilities, or other FORTRAN application dependent features.

The high level, structured, interactive programming language in the DELIGHT system is called RATTLE (an acronym for RATfor Terminal Language Environment) and was evolved from the structured language RATFOR. The structured 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':

$$\text{lp } z = \text{argmin} \{ c^*x \mid x \geq 0, x \leq d, A^*x \leq b \}$$

where the array z is assigned the minimizing value of x, and the matrix A and vectors b,c,d have previously been defined. This macro creates all the necessary work arrays and inputs for the call to a built-in Harwell Library linear programming FORTRAN routine. Thus, macros not only enhance readability, but 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 matop macro. The following table gives a sample of operations available to the user through macros:

| Operation | Macro Syntax |
|--------------------|-------------------------|
| add matrices | matop A = B + C |
| multiply matrices | matop A = B * C |
| eigenvalues | matop lambda = eigen(A) |
| inverse of matrix | matop Ainv = inv(A) |
| solve linear eqns. | lineq A*x = b |
| inner product | <<x,y>> |

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 allow the command "window name" to be substituted for the specification of the particular set of world coordinates [N2], and corresponding viewport coordinates which are associated with the window.

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:

```

while ( f(x) > eps ) {
  x = x - f(x)/derf(x)
  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 inter-

rupted by the user or by the program and later resumed after modifying variable values or even re-compiling an entire sub-procedure. 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]:

```
procedure Armijo {
  repeat {
    evaluate h = dir(X[Iter])
    lambda = step (X[Iter], h)
    update X[Iter+1] = X[Iter] + lambda * h
    Iter = Iter + 1
    output
  }
  forever
}
```

where "evaluate" and "update" are defines for calling the appropriate subprocedures, "output" is a procedure that produces a display, while X[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.

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.

- [A1] Armijo, L., "minimization of functions having continuous partial derivatives", Pacific J. Math., vol. 16, pp.1-3, 1966.
- [A2] Athans, M., "The role and use of stochastic linear-quadratic-gaussian problem in control system design", IEEE Trans. vol. AC-16, no. 6, 1971.
- [D1] Dongarra, J.J., et.al., LINPACK Users' Guide, SIAM, Philadelphia, Pa., 1979.
- [D2] Denham, M. J., "SLICE: a subroutine library for control system design", Publication ??/82, School of Electronic Engineering and Computer Science, Kingston Polytechnic, Kingston upon Thames KT1 2EE.
- [G1] Gonzaga, C., Polak, E., and Trahan, R., "An Improved Algorithm for Optimization Problems with Functional Inequality Constraints", IEEE Trans., Vol. AC-25, No. 1, 1980.
- [G2] Gonzaga, C., Polak, E., "On Constraint Dropping Schemes and Optimality Functions for a Class of Outer Approximations Algorithms", J. SIAM Control and Optimization Vol. 17, 1979.
- [H1] Harwell Subroutine Library, Harwell, England.
- [L1] Laub, A. J., "An inequality and some computations related to the robust stability of linear dynamic systems", IEEE Trans. on Automatic Control Vol. AC-24, No.2, pp 318-320, 1979.
- [M1] MacFarlane, A. G. J. and Postlethwaite, I., "Generalized Nyquist Stability Criterion and Multivariable Root Loci", Int. J. Control, Vol. 25(1) 1977.
- [M2] Mayne, D. Q., Polak, E., "A Quadratically Convergent Algorithm for Solving Infinite-Dimensional Inequalities", Memo No. UCB/ERL M80 /11, University of California, Berkeley, 1980.
- [N1] Nye, W., Polak, E., Sangiovanni-Vincentelli, A., and Tits, A., "DELIGHT: an Optimization-Based Computer-Aided-Design System" Proc. IEEE Int. Symp. on Circuits and Systems, Chicago, Ill, April 24-27, 1981.
- [N2] Newman, W.M., and R.F. Sproul, Principles of Interactive Computer Graphics, 2'nd Edition, Mcgraw-Hill, N.Y., 1979.
- [P1] Polak, E. and Mayne, D. Q., "An Algorithm for Optimization Problems with Functional Inequality Constraints", IEEE Trans., Vol. AC-21, No. 2, 1976.
- [P2] Polak, E., and Wardi, Y. Y., "A nondifferentiable optimization algorithm for the design of control systems subject to singular

value inequalities over a frequency range", Automatica, Vol. 18, NO. 3, pp. 267-283, 1982.

- [P3] Polak, E., Computational Methods in Optimization: A Unified Approach, Academic Press, N.Y., 1971.
- [S1] Safonov, M. G., "Choice of quadratic cost and noise matrices and the feedback properties of multiloop LQG regulators", Proc. Asilomar Conf. on Circuits, Systems, and Computers, Pacific Grove, California, 1979.
- [S2] Safonov, M. G., Laub, A. J., and Hartman, G. L., "Feedback properties of multivariable systems: the role and use of the return difference matrix", IEEE Trans. on Control Vol. AC-26, 1981.
- [S3] Sandel, N. R., "Robust stability of systems with applications to singular value perturbations", Automatica Vol. 15, 1979.
- [S4] Shearer, B. R. and Field, A. D., "Multivariable Design System (MDS): An interactive package for the design of multivariable control systems", Publication No. 75/29, Department of Computing and Control, Imperial College, London, SW7 2BZ, 1975.
- [S4] Stein, G., "Generalized quadratic weights for asymptotic regulator properties", IEEE Trans., Vol. AC-24, 1979.
- [T1] Taiwo, O., "Design of a Multivariable Controller for a High Order Turbofan Engine Model by Zakian's Method of Inequalities", IEEE Trans. Vol. AC-23, No. 5, 1978.
- [T2] Trahan, R., and Polak, E., "A Derivative Free Algorithm for a Class of Infinitely Constrained Optimization Problems", IEEE Trans. Vol. AC-25, No. 1, 1979.
- [W1] J. Wilander, "An interactive programming system for Pascal", BIT
- [Z1] Zakian, V., "New Formulation for the Method of Inequalities", Proc. IEE, 126(6), 1979.