OPTIMIZATION OF LINEAR SYSTEMS

Thesis for the Degree of Ph. D

MICHIGAN STATE UNIVERSITY

Vijay K. Jain

1964

This is to certify that the

thesis entitled

OPTIMIZATION OF LINEAR SYSTEMS

presented by

Vijay K. Jain

has been accepted towards fulfillment
of the requirements for

Ph.D degree in Electrical Engineering

Harry G. Hedges
Major professor

Date August 10, 1964

O-169

ABSTRACT

OPTIMIZATION OF LINEAR SYSTEMS

by Vijay K. Jain

In this thesis, certain problems in the optimization of linear systems are considered. Techniques are given to carry out minimization (or maximization) of arbitrary object functions with respect to the system parameters and control functions. Using the theory of functions of matrices, explicit formulas for some important object functions have been derived and the method of steepest descent is used to minimize the object function. The cases of terminal state error, and integral-quadratic-error to a step or ramp input-function are solved specifically. Extension to include time weighting is also considered. A practical procedure for stabilizing an initially unstable linear system, of arbitrary order, with respect to q variables is given.

A new approach to time optimal control has been developed. It has been shown that switching the parameters of the system and the control functions a better time optimality can be obtained. Time optimal control is discussed also for the case when certain mass and other parameters of the system are time varying with their variation depending on the controls. For a special case, the optimal controls are of the generalized "bang-bang" type. For the fixed time problem with integral- quadratic- error and control, a theoretical investigation is given and an explicit solution is obtained for the case when the transition time grows out of bounds.

OPTIMIZATION OF LINEAR SYSTEMS

BY

Vijay K. Jain

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering

1964

ACKNOWLEDGEMENTS

The author wishes to thank Dr. H. E. Koenig for his guidance and help, Dr. H. G. Hedges for his constant encouragement, and Dr. J. S. Frame for his inspiration and suggestions.

He also wishes to thank the U.S. Agency for International Development and the Birla Institute of Technology and Science, (earlier Birla College of Engineering), Pilani, for their joint sponsorship which made this work possible.

CONTENTS

LIST OF APPENDICES

1. INTRODUCTION

The concepts of optimum control and performance are basic to the economic studies, control system design, guidance and other physical systems. The increasing complexity of these systems created the impelling need for improved analytical description of the system[3-4] and its performance measure.[6-10] For example, consider the system $\mathcal{A}$ in Fig. 1 actuated by the controller $\mathcal{U}$ continuously (or discretely) in time. The state of the system at time t can be characterized by a real n-dimensional vector x(t). The objective of the controller may be to transfer the state of

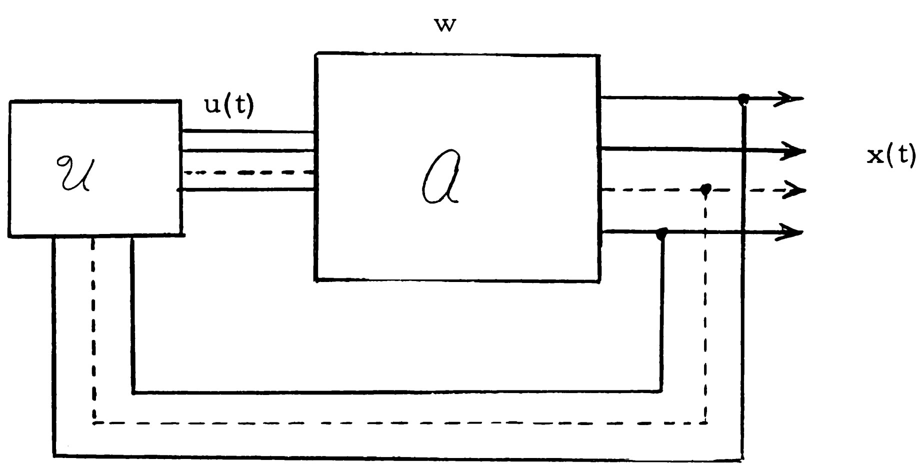


Figure 1

the system x^o at time t_o to some state x^1 at time t_1 in such a manner that the object function

$$J = \int_{t_o}^{t_1} f_o(x(t), u(t))dt$$

is a minimum, where the system state x(t) is related to the control by the first order vector differential equation

$$\dot{x}(t) = f(x(t), w, u(t))$$

The object function may be a measure of the cost of control, a measure of the error relative to a desired performance, or some other physical measure. For physical realization, it is required to impose certain restrictions on the control function u(t), $t_o \leq t \leq t_1$, namely that it shall belong to a certain class of functions (e.g. piece-wise continuous functions) and it shall have values in some bounded region $\overline{U}$ of a suitable r-dimensional space.

In the past, it has been a practice to realize an acceptable design of a dynamic system on the analog computer by adjustment of parameters. With the development of state-space models and the modern computing facilities, a new level of sophistication in design is possible. Not only is it possible to meet the requirements of the design, but in many cases an optimum dynamic performance may be obtained as part of a routine design procedure. Although extensive work has been done in the way of highly theoretical investigations, recently by Pontryagin,[1, 2] Bellman[11] and others and earlier through the subject of Calculus of Variations,[12] comparatively little has been achieved in the way of applications, perhaps because of the awe with which these theories have been viewed.

A general solution encompassing all optimization problems cannot be expected, not only because of the large variety of problems, but also because of the extreme difficulty encountered in solving non-linear algebraic equations, particularly when the order of equations is large. However, judicious choice of the theoretical tools and appropriate use of the computer can considerably soften these intrinsic difficulties. This thesis is an attempt in the direction of bringing these highly theoretical results closer to practice and also bring some new concepts in time optimality. For many of the results, computer programs have been written and used for obtaining concrete results. These programs are presented as a supplement to this thesis.

In section 2, the theory of parameter optimization for arbitrary object functions is developed. Optimization of terminal state object functions are considered specifically and illustrated by an example taken from the area of multifeedback control systems. Procedures for determining the parameters for best simulation of a specified response of a system to a given input function are developed and explicit formulas for calculating the object function in a closed form are given for the cases of step and ramp inputs. Procedures for minimizing the integral square error, including time weighting, over infinite transition time are also considered and applied to an example of an electro-mechanical system. Section 2 concludes with a computer procedure for stabilizing a system of arbitrary order with q variable parameters.

An object function of the highest importance is 'time'. Rather recently, it has been established that for a linear system, driven by a control with a limited magnitude, the bang-bang control yields a better time optimality[22] than a linear controller. It is believed that a step further has been taken in this direction in Section 3 of this thesis. If certain system parameters represented by the q-dimensional vector $\underline{w}$ can be changed within a certain set of values $\overline{W}$, then by changing these values at suitable instants, in conjunction with the manipulation of controls, a better solution to the optimal time problem results. A second order mechanical system has been used to illustrate this point.

The fixed time problems, where the final state of the process is arbitrary, is of interest in certain economic studies. A theoretical procedure is given although an explicit solution for the optimal control may, in general, be difficult to obtain except in the case when the transition time grows out of hand.

Appendix A states some standard theorems and results on functions of matrices and systems of differential equations that are necessary to the development of Section 2. Formulas for calculating a function of a real matrix in the real domain are derived in terms of modified constituent matrices and the modified eigenvalues. Also, it has been shown that the modified eigenvalues of a linear state model for an arbitrary parameter vector $\underline{w}$, can be evaluated from the initial eigenvalues by numerical integration of certain differential equations.

In Appendix B, a proof of Pontryagin's maximum principle, based on the theory of calculus of variations, is given.

2. PARAMETER OPTIMIZATION OF LINEAR SYSTEMS

2.1 Preliminary Considerations

The systems used as a basis for development are deterministic and governed by linear, ordinary differential equations. Techniques for developing the mathematical model describing the dynamic performance of linear physical systems have already been established. If the components of a system can be characterized by linear differential equations then using the topological properties of the linear graph of the system, the mathematical model of the system consisting of a set of linear differential equations and possibly a set of algebraic equations can be obtained.[3,4] For linear systems, these equations, called the State Model of the system, are of the form

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx + Du \tag{2}$$

where x is the n-dimensional state-vector in the n-space $\underline{X}$, A is a real nxn matrix, B is a real nxr matrix, and u is an r-dimensional vector with range in a subset $\overline{U}$ of the r-space. In some cases, A and B may take on values in families of matrices A(w) and B(w) where the q-dimensional vector $\underline{w}$ of parameters may assume values in a specified subset $\overline{W}$ of the q-space.

Equation (2) indicates that the "output" vector $\underline{y}$ of dimension $\underline{p}$ which assumes its values in the p-space is obtained as a linear transformation of the state vector $\underline{x}$ and the vector $\underline{u}$. In particular, any performance measure or object function J, is uniquely determined by the trajectory x(t), $t_o \leq t \leq t_1$, and the control function u(t), $t_o \leq t \leq t_1$, i.e.,

$$J = \int_{t_o}^{t_1} g(y(t),\ x(t),\ u(t))dt = \int_{t_o}^{t_1} f_o(x(t), u(t))dt \qquad (3)$$

For the purposes of optimization studies, it is convenient to transform the above integral relation as follows. Introduce a new variable x_o defined as

$$x_o = \int_{t_o}^{t} f_o(x, u)dt,\ x_o(t_1) = J \qquad (4)$$

then upon differentiation of (4)

$$\dot{x}_o = f_o(x, u) \qquad (5)$$

where f_o is a scalar function which is positive definite. If f_o is defined and continuous in $\underline{\overline{X}}$ x $\overline{U}$ it is said to be in class $C(\underline{\overline{X}} \times \overline{U})$. If in addition, it has continuous partial derivatives* with respect to x in $\underline{\overline{X}}$ then it will be said to be in classes $C(\underline{\overline{X}} \times U)$ and $D(\underline{\overline{X}})$.

2.2 General Theory

For the linear system (1), let the control signal u(t) be specified on the time interval $[0, t_1]$. Let the parameter vector $\underline{w} = (a, b)$, assume values in a closed and bounded subset $\overline{W} = \overline{W}_a x \overline{W}_b$ of the q-dimensional vector space, where $\underline{a}$ and $\underline{b}$ are q_a and q_b dimensional

---

* This statement would be used to mean that all entries of the row vector $(\partial f_o/\partial x)$ have this property. In general, $\partial f/\partial x$ has a certain property would be used to mean that all entries of the Jacobian matrix $[\partial f/\partial x]$ have that property.

vectors whose components are entries of the matrices A and B. The problem is to find the parameter vector $\underline{w}$ which minimizes the performance measure $\underline{J}$ in (4) when f_o is in $C(\overline{\underline{X}} \times \overline{U})$ and $D(\overline{\underline{X}})$.

The minimization procedure considered here requires the evaluation of the gradient $\partial J/\partial w$. To establish this gradient let (1) be rewritten as

$$\begin{aligned} \dot{x} &= A(a)x + B(b)u \\ &= f(x, u, a, b) \\ &= f(x, w) \end{aligned} \qquad (6)$$

Since $\underline{f}$ is linear in $\underline{w}$, as stated in the problem, the partials $\frac{\partial f}{\partial w}$ exist and are continuous for $\underline{w}$ in $\overline{W}$, and the solution $x(t, w)$ to (6), has[13] continuous partials $\partial x(t, w)/\partial w$. Therefore, the partials $\partial f_o(x, u)/\partial w$ are continuous for $\underline{w}$ in $\overline{W}$ and it follows from (5) that the solution x_o has continuous partials $\partial x_o/\partial w$. In particular, for $t = t_1$, we have the following Lemma.

Lemma 1. For the linear system (6), and the performance measure defined by (3), the performance measure has continuous partial derivatives with respect to $\underline{w}$, i.e. $\partial J/\partial w$ exist and are continuous.

Lemma 2. If in addition to the hypothesis of Lemma 1, the integral in (3) converges, uniformly in $\underline{w}$, $\underline{w} \in \overline{W}$, then the performance measure $J = x_o(\infty)$, has continuous partial derivatives with respect to $\underline{w}$.

Let $\lambda = (\alpha_1 + j\beta_1, \alpha_1 - j\beta_1, \ldots, \lambda_{2m+1}, \ldots, \lambda_n)$, where

$$\alpha_i, \beta_i, \; i = 1, \ldots, m$$

$$\lambda_k, \; k = 2m + 1, \ldots, n$$

are all real, be the vector of eigenvalues of the matrix A and let

$$\lambda^* = (\alpha_1, \beta_1, \ldots, \lambda_{2m+1}, \ldots, \lambda_n)$$

be the modified vector of eigenvalues, having real entries corresponding to the real and imaginary parts of the complex eigenvalues in the first 2m entries and the real eigenvalues in the following n-2m entries.

It is shown in the Appendix A that $\underline{J}$ is a function of the modified vector of eigenvalues $\lambda^* = \lambda^*(a)$ and the parameter vector $\underline{w}$. Therefore, the gradient of $\underline{J}$ is given by

$$\frac{\partial J}{\partial w} = \left(\left(\frac{\partial J}{\partial \lambda^*}\right)\left[\frac{\partial \lambda^*}{\partial a}\right] + \left(\frac{\partial J}{\partial a}\right), \left(\frac{\partial J}{\partial b}\right)\right) \tag{7}$$

The method of steepest descent[14-18] can now be used to minimize $\underline{J}$ with respect to $\underline{w}$. Let $w \in \overline{W} = \overline{W}'_a \times \overline{W}_b$ where $\overline{W}_b$ is the allowable set of parameter vectors $\underline{b}$ and $\overline{W}'_a$ is the subset of allowable values of the parameter vector $\underline{a}$ such that the matrix A(a) has distinct eigenvalues. From the theory of the method of steepest descent the variation of $\underline{w}$ along the path of steepest descent is given by

$$\frac{dw}{d\sigma} = - C\left(\frac{\partial J}{\partial w}\right)' \tag{8}$$

where σ is an independent parameter and C is a suitable weighting matrix. One simple and obvious choice for C is the unit matrix. A local minimum for $\underline{J}$ is obtained as a solution to (8) as $\sigma \to \infty$.

An alternative to a direct solution of (8) is a one-dimensional search. Discretizing (8)

$$w(k+1) = w(k) - C\left(\frac{\partial J}{\partial w}\right)'\Bigg|_{w(k)} \Delta\sigma \tag{9}$$

For a fixed value of the vector $\left(\frac{\partial J}{\partial w}\right)'\Big|_{w(k)}$ the scalar $\Delta\sigma$ is increased from 0 until J takes on a minimum value where J is a function of w(k + 1) and λ *. A new gradient is now found and the procedure is continued until the ratio J(k+1)/J(k) is suitably near unity. For each successive value of the parameter w(k + 1), λ * is evaluated by quadrature integration of Eq. (A19).

2.3.1 Terminal State Optimization

A control system is often designed on the basis of a terminal state criterion, i.e.,

$$J = f_o[(x - x^d), u]\Big|_{t = t_1} \tag{10}$$

where x^d is a desired or specified response to the system. Typical examples of the function f_o are:

(i) $$J = x'(t_1)\, x\,(t_1) \tag{11}$$

representing the euclidean norm of the deviation from the equilibrium state $\underline{x} = 0$.

(ii) $$J = (x(t_1) - x^d)'\, G(x(t_1) - x^d) \tag{12}$$

which represents a linearly transformed (or weighted) euclidean norm of the deviation from a desired terminal state.

For the important case when the input is a unit-step function of time, the system state is given as

$$x(t) = \sum_{i=1}^{n} Z_i \left[e^{\lambda_i t}\left(x_o + \frac{B}{\lambda_i}\right) - \frac{B}{\lambda_i}\right]$$

and the object function is

$$J \leq \sum_{i=1}^{n} \sum_{k=1}^{n} [e^{(\lambda_i+\lambda_k)t}(x_o' + \frac{B'}{\lambda_i}) Z_i'GZ_k(x_o + \frac{B}{\lambda_k})]$$

$$- [e^{\lambda_k t}(x^{d'} + \frac{B'}{\lambda_i})Z_i'GZ_k(x_o + \frac{B}{\lambda_k})]$$

$$- [e^{\lambda_i t}(x_o' + \frac{B}{\lambda_i})Z_i'GZ_k(x^{d'} + \frac{B}{\lambda_k})] + [(x^d + \frac{B}{\lambda_i})Z_i'GZ_k(x^d + \frac{B}{\lambda_k})] \tag{13}$$

These and the general performance measure

$$J = f_o[(x - x^d), (\dot{x} - \dot{x}^d), u]$$

can be treated according to the general theory of this section.

Example 1

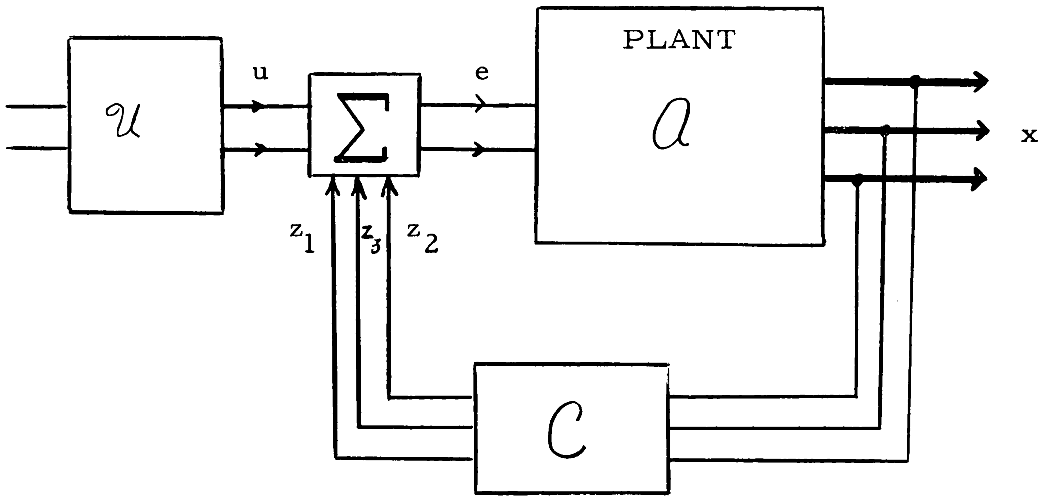


Fig. 2

Consider the system shown in Fig. 2 where the plant characterized by the linear model

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3.75 & -7.25 & -5. \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} z_1 \\ e_2 \\ e_3 \end{bmatrix} ;$$

$$e_2 = z_2 + u_1 \qquad \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \text{(E.1.1)}$$
$$e_3 = z_3 + u_2$$

has a feedback circuit described by

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -c_1 \\ -c_2 & -3 & -1 \\ 0 & 0 & c_1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \text{(E.1.2)}$$

where $\underline{z}_1$, z_2 and $\underline{z}_3$, represent the feedback signals and the parameters $\underline{c}_1$ and $\underline{c}_2$ are constrained to the region

$$|c_1| \leq 0.1 \qquad |c_2| \leq 1.0 \qquad \text{(E.1.3)}$$

and the inputs are specified as

$$u_2 = 2.0, \quad u_3 = 1.0 \qquad \text{(E.1.4)}$$

It is desired to find the value of the parameter vector (c_1, c_2) which will minimize the scalar

$$J = (x(t_1) - x_{ss})' \, G(x(t_1) - x_{ss})$$

where $x_{ss} = x(\infty)$, $t_1 = 5$ and the weighting matrix is

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The state equations of the system are

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3.75 & -7.25 & -5. \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ u_1 \\ u_2 \end{bmatrix}$$

$$+ \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & -c_1 \\ -c_2 & -3 & -1 \\ 0 & 0 & c_1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & (1+c_1) \\ (-3.75 - c_2) & -10.25 & -6. \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

or

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & a_1 \\ a_2 & -10.25 & -6. \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \text{(E.1.5)}$$

where

$$a_1 = 1 + c_1$$

$$a_2 = -3.75 - c_2$$

The matrix $\left[\frac{\partial \lambda^*}{\partial a}\right]$ required in (7) is

$$\begin{bmatrix} \dfrac{-(10.25\lambda_1 - a_2)}{3\lambda_1^2 + 12\lambda_1 + 10.25a_1} & \dfrac{a_1}{3\lambda_1^2 + 12\lambda_1 + 10.25a_1} \\ \dfrac{-(10.25\lambda_2 - a_2)}{3\lambda_2^2 + 12\lambda_2 + 10.25a_1} & \dfrac{a_1}{3\lambda_2^2 + 12\lambda_2 + 10.25a_1} \\ \dfrac{-(10.25\lambda_3 - a_2)}{3\lambda_3^2 + 12\lambda_3 + 10.25a_1} & \dfrac{a_1}{3\lambda_3^2 + 12\lambda_3 + 10.25a_1} \end{bmatrix}$$

and the detailed form of differential equations as in Eq. (A16) are

$$\frac{d\lambda_i}{da_1} = - \frac{(10.25\lambda_i - a_2)}{3\lambda_i^2 + 12\lambda_i + 10.25a_1}$$

$$\frac{d\lambda_i}{da_2} = \frac{a_1}{3\lambda_i^2 + 12\lambda_i + 10.25a_1}$$

The machine solution along with the initial parameter and object function are

	Parameter Vector		Object Function
	a_1	a_2	
Initial	1.0	-3.75	0.3183
Final	0.9	-3.836	0.1338

giving the parameter vector for minimum object function.

2.3.2 Trajectory Optimization

A performance measure of considerable physical significance and practical importance is

$$J = \int_{t_o}^{t_1} f_o[(x(t) - x^d(t)), \; u(t)]dt \tag{14}$$

where $x^d(t)$ is a specified or desired trajectory. As an example, consider a position servo described by the second order differential equation

$$\ddot{\theta} + a_1\dot{\theta} + a_2\theta = u \tag{15}$$

setting $\theta = x_1$, $\dot{\theta} = x_2$ in (15),

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix} + \begin{bmatrix} 0 \\ u \end{bmatrix}$$

It is required to find $(a_1, a_2) \in \overline{W}$ such that for any specified $\beta(t)$ the integral-square-error performance measure

$$J = \int_{t_o}^{t_1} [\theta(t) - \beta(t)]^2 \; dt$$

$$= \int_{t_o}^{t_1} [x_1(t) - \beta(t)]^2 \; dt$$

is minimized.

Explicit formulas for this performance index and for the general case of an n-th order system for a step input response are derived below. Using equation (A13) of the Appendix A

$$x(t) = \sum_{i=1}^{n} Z_i[e^{\lambda_i t}(x_o + \frac{1}{\lambda_i} B) - \frac{1}{\lambda_i} B] \quad (17)$$

If the response characteristic is taken to be of the form

$$x^d(t) = \sum_{k=1}^{m} C_k \quad e^{a_k t} \quad (18)$$

where the vectors C_k and the scalars a_k, k = 1, ... m are given, then the integral-quadratic-error performance measure is calculated as

$$J = \int_o^{t_1} [x(t) - x^d(t)]\, G[x(t) - x^d(t)]dt$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \Bigg\{ \left[\frac{e^{(\lambda_i + \lambda_j)t_1} - 1}{(\lambda_i + \lambda_j)} \cdot (x_o' + \frac{1}{\lambda_i} B')Z_i'GZ_j(x_o + \frac{1}{\lambda_j} B) \right]$$

$$- \frac{e^{\lambda_i t_1} - 1}{\lambda_i \lambda_j} \left[(x_o' + \frac{1}{\lambda_i} B')Z_i'GZ_jB \right] - \left[B'Z_i'(x_o + \frac{1}{\lambda_j} B) \right] \cdot$$

$$\frac{e^{\lambda_j t_1} - 1}{\lambda_i \lambda_j} + \frac{t_1}{\lambda_i \lambda_j} \quad B'Z_i'GZ_jB \Bigg\}$$

$$- \sum_{i=1}^{n} \sum_{k=1}^{m} \left\{ \left[\frac{e^{(\lambda_i + a_k)t_1} - 1}{(\lambda_i + a_k)} (x_o' + \frac{1}{\lambda_i} B') Z_i' G C_k \right] - \frac{e^{a_k t_1} - 1}{a_k \lambda_i} B' Z_i' G C_k \right\}$$

$$- \sum_{k=1}^{m} \sum_{i=1}^{n} \left\{ \left[\frac{e^{(a_k + \lambda_i)t_1} - 1}{(a_k + \lambda_i)} C'_k G Z_i (x_o + \frac{1}{\lambda_i} B) \right] - \frac{e^{a_k t_1} - 1}{a_k \lambda_i} C_k' G Z_i B \right\}$$

$$+ \sum_{k=1}^{m} \sum_{j=1}^{m} \frac{e^{(a_k + a_j)t_1} - 1}{(a_k + a_j)} C_k' G C_j \qquad (19)$$

If the desired response in (18) has a slightly different form

$$x^d(t) = \sum_{k=1}^{m} C_k e^{a_k t} + x_{ss} \qquad (20)$$

where x_{ss} is the steady state value of (17) and $x_o = 0$ then (19) reduces to the especially simple and useful form

$$J = \sum_{i}^{n} \sum_{j}^{n} \left\{ \frac{e^{(\lambda_i + \lambda_j)t_1} - 1}{\lambda_i \lambda_j (\lambda_i + \lambda_j)} B' Z_i' G Z_j B \right\}.$$

$$- \sum_{i=1}^{n} \sum_{k=1}^{m} \frac{e^{(\lambda_i + a_k)t_1} - 1}{\lambda_i(\lambda_i + a_k)} \left\{ B' Z_i' G C_k + C_k' G Z_i B \right\}$$

$$+ \sum_{k=1}^{m} \sum_{j=1}^{m} \frac{e^{(a_k + a_j)t_1} - 1}{(a_k + a_j)} C_k' G C_j \tag{21}$$

In the case of a velocity servo, where the criterion command-signal is a ramp and the state equations are of the form

$$\dot{x} = A x + B t \tag{22}$$

the response is given as

$$x(t) = e^{At}[x_o - \sum_{i=1}^{n} Z_i B e^{-\lambda_i t} \frac{(t\lambda_i + 1)}{\lambda_i^2} \Bigg|_0^t]$$

$$= \sum_{i=1}^{n} Z_i [e^{\lambda_i t}(x_o + \frac{1}{\lambda_i^2} B) - (\frac{t}{\lambda_i} + \frac{1}{\lambda_i^2})B] \tag{23}$$

If the desired response is taken as

$$x^d(t) = \sum_{t=1}^{m} (C_i + t D_i) e^{a_i t} \tag{24}$$

where the vectors C_i, D_i and the scalars a_i, i=1, . . . m are given, then the integral- quadratic- error performance index for the velocity

servo can be calculated explicitly from the formulas derived below by substituting (23) and (24).

$$J = \int_o^{t_1} \sum_{i=1}^{n} \sum_{k=1}^{n} \left[e^{\lambda_i t}(x_o' + \frac{1}{\lambda_i^2} B') - (\frac{t}{\lambda_i} + \frac{1}{\lambda_i^2})B' \right] Z_i$$

$$\cdot \; GZ_k \left[e^{\lambda_k t} (x_o + \frac{1}{\lambda_k^2} B) - (\frac{t}{\lambda_k} + \frac{1}{\lambda_k^2})B \right] dt$$

$$- \int_o^{t_1} \sum_{i=1}^{n} \sum_{k=1}^{m} \left[e^{(\lambda_i + a_k)t} (x_o' + \frac{1}{\lambda_i^2} B') - (\frac{te^{a_k t}}{\lambda_i} + \frac{e^{a_k t}}{\lambda_i^2})B' \right] Z_i$$

$$\cdot \; G(C_k + t\, D_k) dt$$

$$- \int_o^{t_1} \sum_{i=1}^{m} \sum_{k=1}^{n} (C_i + t\, D_i)' GZ_k \quad e^{(\lambda_k + a_i)t} (x_o + \frac{1}{\lambda_k^2} B)$$

$$- (\frac{t\, e^{a_i t}}{\lambda_k} + \frac{e^{a_i t}}{\lambda_k^2}) B \quad dt$$

$$+ \int_o^{t_1} \sum_{i=1}^{m} \sum_{k=1}^{m} (C_i + t\, D_i)' G(C_k + t\, D_k)\, e^{(a_i + a_k)t} dt \quad (25)$$

Upon integrating by parts, each of the indicated integrals reduces to an algebraic expression in the constituent matrices, eigenvalues, initial state of the system, the vector B, and the constants of the specified response. This expression is explicitly

$$J = J_1 + J_2 + J_3 + J_4 \tag{26}$$

where

$$J_1 = \sum_i^n \sum_k^n [Z_i(x_o + \frac{1}{\lambda_i^2} B)]'G \quad Z_k \left\{ (x_o + \frac{1}{\lambda_i^2} B) \frac{e^{(\lambda_i + \lambda_k)t_1} - 1}{(\lambda_i + \lambda_k)} \right.$$

$$\left. -B \left(\frac{e^{\lambda_i t_i}(\lambda_i t_i - 1) + 1}{\lambda_i^2 \lambda_k} + \frac{e^{\lambda_i t_i} - 1}{\lambda_i \lambda_k^2} \right) \right\}$$

$$- \sum_i^n \sum_k^n B'Z_i'G \, Z_k \left\{ (x_o + \frac{1}{\lambda_k^2} B) \left[\frac{e^{\lambda_k t_i}(\lambda_k t_i - 1) + 1}{\lambda_k^2 \lambda_i} + \frac{e^{\lambda_k t_i} - 1}{\lambda_k \lambda_i^2} \right] \right.$$

$$\left. -B \left(\frac{t_1^3}{3\lambda_i \lambda_k} + \frac{(\lambda_i + \lambda_k) t_1^2}{2\lambda_i^2 \lambda_k^2} + \frac{t_1}{\lambda_i^2 \lambda_k^2} \right) \right\} \tag{27}$$

$$J_2 = \sum_i^n \sum_k^m [Z_i(x_o + \frac{B}{\lambda_i^2})]' \, G \left\{ C_k \frac{e^{(\lambda_i + a_k)t_1} - 1}{(\lambda_i + a_k)} \right.$$

$$\left. + D_k \frac{e^{(\lambda_i + a_k)t_1}[(\lambda_i + a_k)t_1 - 1] + 1}{(\lambda_i + a_k)^2} \right\}$$

$$- \sum_i^n \sum_k^m B'Z_i'G \left\{ C_k \left[\frac{e^{a_k t_1}(a_k t_1 - 1) + 1}{a_k^2 \lambda_i} + \frac{e^{\lambda_i t_1} - 1}{a_k \lambda_i^2} \right] \right.$$

$$\left. + D_k \left[\frac{e^{a_k t_1}(a_k^2 t_1^2 - 2a_k t_1 + 2)}{\lambda_i a_k^3} + \frac{e^{a_k t_1}(a_k t_1 - 1) + 1}{\lambda_i^2 a_k^2} \right] \right\}$$

(28)

$$J_3 = \left\{ \sum_{i=1}^{m} \sum_{k=1}^{n} C_i' \frac{e^{(\alpha_i + \lambda_k)t_1} - 1}{(\alpha_i + \lambda_k)} \right.$$

$$\left. - \sum_{i=1}^{m} \sum_{k=1}^{n} D_i' \frac{e^{(\alpha_i + \lambda_k)t_1}((\alpha_i + \lambda_k)t_1 - 1) + 1}{(\alpha_i + \lambda_k)^2} \right\} GZ_k(x_o + \frac{B}{\lambda_k^2})$$

$$- \left\{ \sum_{i=1}^{m} \sum_{k=1}^{n} C_i' \left(\frac{e^{\alpha_i t_1}(\alpha_i t_1 - 1) + 1}{\alpha_i^2 \lambda_k} + \frac{e^{\lambda_k t} - 1}{\alpha_i \lambda_k^2} \right) \right.$$

$$+ \sum_{i=1}^{m} \sum_{k=1}^{n} D_i' \left(\frac{e^{\alpha_i t_1}(\alpha_i^2 t_1^2 - 2\alpha_i t_1 + 2) - 2}{\lambda_k \alpha_i^3} \right.$$

$$\left. \left. + \frac{e^{\alpha_i t_1}(\alpha_i t_1 - 1) + 1}{\lambda_k^2 \alpha_i^2} \right) \right\} GZ_k \qquad (29)$$

$$J_4 = \sum_{i=1}^{m} \sum_{k=1}^{m} C_i' G C_k \frac{e^{(\alpha_i + \alpha_k)t_1} - 1}{\alpha_i + \alpha_k}$$

$$+ \sum_{i=1}^{m} \sum_{k=1}^{m} (D_i' G C_k + C_i' G D_k) \frac{e^{(\alpha_i + \alpha_k)t_1}((\alpha_i + \alpha_k)t_1 - 1) + 1}{(\alpha_i + \alpha_k)^2}$$

$$+ \sum_{i=1}^{m} \sum_{k=1}^{m} D_i' G D_k \, e^{(\alpha_i + \alpha_k)t_1} \frac{((\alpha_i + \alpha_k)^2 t_1^2 - 2(\alpha_i + \alpha_k)t_1 + 2}{(\alpha_i + \alpha_k)^3} \qquad (30)$$

Example 2 Let the equation (36) correspond to the second order velocity servo shown in the block diagram of Fig. 3.

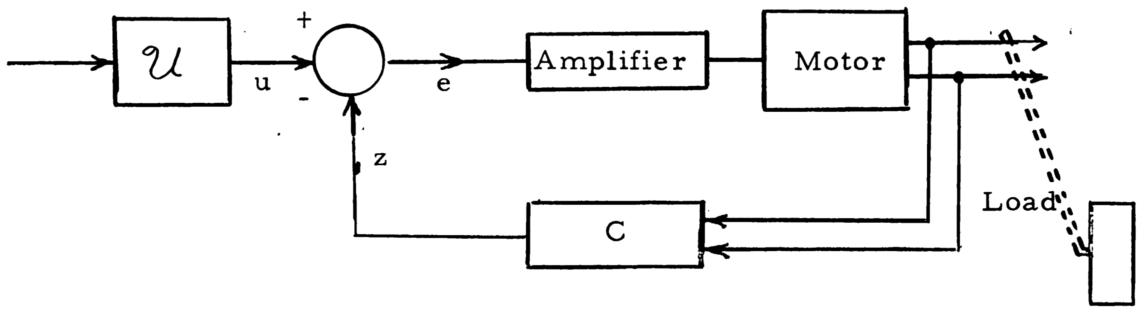


Fig. 3

Let the plant characteristic be taken as

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1.5 & -3.0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} e; \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{(E2.1)}$$

and the desired response to u(t) = t be taken as

$$x_1^d(t) = t$$
$$x_2^d(t) = \text{arbitrary} \quad \text{(E2.2)}$$

The transition time is taken to be $t_1 = 2.0$ and the feedback equation is taken to be

$$z = [-c_1 \quad -c_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{(E2.3)}$$

where $\underline{c}_1$ and $\underline{c}_2$ are constrained to the set $\overline{W}_a$.

$$0.8 \leq c_1 \leq 1.7$$
$$2.0 \leq c_2 \leq 3.0 \qquad \text{(E2.4)}$$

It is required to find the value of the parameters $\underline{c}_1$ and $\underline{c}_2$ subject to the above constraints which minimize

$$J = \int_0^{t_1} (x - x^d)' \; G(x - x^d)dt$$

where the weighting matrix G is

$$G = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad \text{(E2.5)}$$

The state equations for the system as obtained from (E2.1) and

$$e = u - c' x \qquad \text{(E2.6)}$$

are

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} t \qquad \text{(E2.7)}$$

where

$$a_1 = 1.5 + c_1$$
$$a_2 = 3.0 + c_2 \qquad \text{(E2.8)}$$

The characteristic equation of the system matrix in (E2.7) is

$$\lambda^2 + a_2\lambda + a_1 = 0 \qquad \text{(E2.9)}$$

To apply the theory of section 3.1, evaluate the matrix $[\frac{\partial \lambda^*}{\partial a}]$ of equation (7) as

$$[\frac{\partial \lambda^*}{\partial a}] = \begin{bmatrix} -\frac{1}{2\lambda_1 + a_2} & -\frac{\lambda_1}{2\lambda_1 + a_2} \\ -\frac{1}{2\lambda_2 + a_2} & -\frac{\lambda_2}{2\lambda_1 + a_2} \end{bmatrix}$$

The eigenvalues for any parameter vector can be evaluated from the initial eigenvalues by integrating the differential equations in (A16) which in this case are

$$\frac{d\lambda_i}{da_1} = -\frac{1}{2\lambda_i + a_2}$$

$$\frac{d\lambda_i}{da_2} = -\frac{\lambda_i}{2\lambda_i + a_2}$$

The results of computer solution of this optimization problem are

	Parameter	Vector	Performance Measure
	a_1	a_2	
Initial	1.5	3.0	7.623
Final	0.8	2.183	5.634

giving the best parameter vector for optimum performance measure.

2.3.3 Infinite Transition Time:

When the transition time is increased without bound $t_1 \to \infty$ in (3) and the performance measure is of the form

$$J = \int_0^{\infty} f_o(x, u)\, dt = \lim_{t_1 \to \infty} \int_0^{t_1} f_o(x, u)\, dt \tag{31}$$

where it is assumed that the limit on the right exists for all $\underline{w}$ in $\overline{W}$. From Lemma 2, it follows that J has continuous partial derivatives $\partial J / \partial W$ and the General Theory of this section applies directly. The following particular case is of considerable importance in conventional control system design and therefore is considered in some detail.

If $\underline{x}_{ss}$ represents the steady state of the system, i.e., $\underline{x}_{ss} = x(\infty)$ then

$$J = \int_{0}^{\infty} (x(t) - x_{ss})' \, G(x(t) - x_{ss}) dt \qquad (32)$$

where x(t) is the solution to the system

$$\dot{x} = A(a) + B(b) \; u(t) \qquad (33)$$

with u(t) a unit step function of time. For $\underline{J}$ to have meaning A(a) will be assumed to have negative real parts throughout $\overline{W}$. The computer algorithm developed here requires the inessential assumption that $\underline{w}$ is in $\overline{W}'$ i.e., the eigenvalues of A(a) are distinct. Let these eigenvalues be designated as $\lambda_1, \overline{\lambda_1}, \ldots \lambda_m, \overline{\lambda}_m, \lambda_{2m+1} \ldots \lambda_n$ where $\lambda_i = \alpha_i + j\beta_i$, $i = 1, \ldots m$ are complex and λ_i, $i = 2m+1, \ldots n$ are real eigenvalues.

Let the respective constituent matrices be designated as

$$Z_1, \overline{Z}_1, \ldots Z_m, \overline{Z}_m, Z_{2m+1}, \ldots Z_n$$

where

$$Z_i = A_i + jB_i, \; i = 1, \ldots m$$

are complex constituent matrices and

$$Z_i, \; i = 2m+1, \; \ldots \; n$$

are real.

It can be shown that the object function in (32) reduces to the closed form

$$J = -u'B' \; [M + N + R] \; Bu \tag{34}$$

where the matrices M, N, R are real and given by

$$M = \sum_{1 \leq i} \; \sum_{, \; k \leq m} \frac{2}{(\alpha_i^2 + \beta_i^2)(\alpha_k^2 + \beta_k^2)} \left\{ \frac{1}{[(\alpha_i + \alpha_k)^2 + (\beta_i - \beta_k)^2]} \cdot \right.$$

$$[(\alpha_i + \alpha_k)(P_i'GP_k + Q_i'GQ_k) + (\beta_i - \beta_k)(P_i'GQ_k - Q_i'GP_k)]$$

$$+ \frac{1}{[(\alpha_i + \alpha_k)^2 + (\beta_i + \beta_k)^2]}$$

$$\left. \cdot \; [(\alpha_i + \alpha_k)(P_i'GP_k - Q_i'GQ_k) - (\beta_i - \beta_k)(P_i'GQ_k + Q_i'GP_k)] \right\} \tag{35}$$

$$N = \sum_{1 \leq i \leq m} \; \sum_{2m+1 \leq k \leq n} \frac{1}{(\alpha_i^2 + \beta_i^2) \, \lambda_k \, [(\lambda_k + \alpha_i)^2 + \beta_i^2]} \cdot$$

$$\left\{ [(\alpha_i + \lambda_k) P_i'GZ_k - \beta_i Q_i'GZ_k] \right.$$

$$\left. + \; 2 \, [(\alpha_i + \lambda_k) Z_k'GP_i - \beta_i Z_k'GQ_i] \right\} \tag{36}$$

$$R = \sum_{2m+1 \leq i,\ k \leq n} \sum \frac{Z_i' \, G \, Z_k}{\lambda_i \lambda_k (\lambda_i + \lambda_k)} \tag{37}$$

with

$$P_i = \alpha_i A_i + \beta_i B_i \tag{38}$$

$$Q_i = \beta_i A_i - \alpha_i B_i \tag{39}$$

<u>Example 3</u>

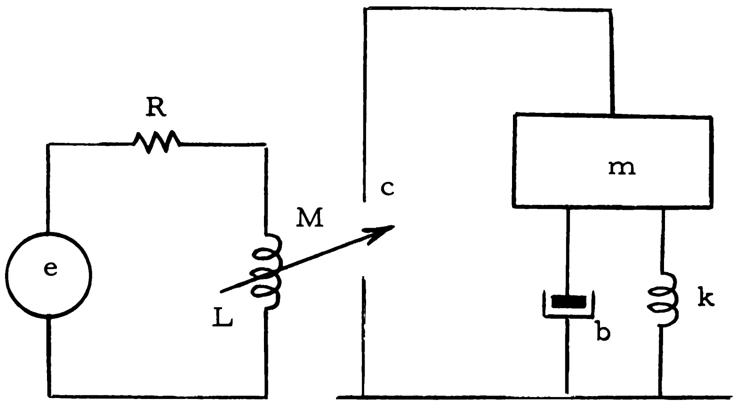


Fig. 4

The dynamics of the electromechanical system shown in Fig. 4 is described by the differential equations

$$e = R\,i + L\frac{di}{dt} + M\,\dot{\delta}$$

$$m\,\ddot{\delta} + b\,\dot{\delta} + k\,\delta = c\,i \tag{E3.1}$$

It is desired to find the parameters <u>R</u> and <u>k</u> such that the object function

$$J = \int_0^\infty (\delta(t) - \delta_{ss})^2\, dt \tag{E3.2}$$

is minimized, where δ_{ss} is the steady-state value of the displacement for a unit step voltage.

Setting $x_1 = \delta$, $x_2 = \dot{\delta}$, $x_3 = i$, the system is described by the state-model

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\frac{k}{m} & -\frac{b}{m} & c \\ 0 & -\frac{M}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{e}{L} \end{bmatrix}$$

Let $-k/m = a_1$ and $-R/L = a_2$ have bound $1 \leq a_1 \leq 6$ and $0.9 \leq a_2 \leq 1.1$. Substituting the fixed parameters into E3.3 gives

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -a_1 & -7 & 1 \\ 0 & -3.75 & -a_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

To minimize the integral square error using the theory of this Section, evaluate the matrix of equation (7) as

$$\begin{bmatrix} -\dfrac{\lambda_1 + a_2}{3\lambda_1^2 + 2(a_2 + 7)\lambda_1 + (7a_2 + 3.75 + a_1)} & -\dfrac{\lambda_1^2 + a_1}{3\lambda_1^2 + 2(a_2 + 7)\lambda_1 + (7a_2 + 3.75 + a_1)} \\ -\dfrac{\lambda_2 + a_2}{3\lambda_2^2 + 2(a_2 + 7)\lambda_2 + (7a_2 + 3.75 + a_1)} & -\dfrac{\lambda_2^2 + a_1}{3\lambda_2^2 + 2(a_2 + 7)\lambda_2 + (7a_2 + 3.75 + a_1)} \\ -\dfrac{\lambda_3 + a_2}{3\lambda_3^2 + 2(a_2 + 7)\lambda_3 + (7a_2 + 3.75 + a_1)} & -\dfrac{\lambda_3^2 + a_1}{3\lambda_3^2 + 2(a_2 + 7)\lambda_3 + (7a_2 + 3.75 + a_1)} \end{bmatrix}$$

The eigenvalues for any given parameter vector are evaluated from the initial eigenvalues by quadrature integration of the follow ng differential equations obtained from (A16).

$$\frac{d\lambda_i}{da_1} = - \frac{\lambda_i + a_2}{3\lambda_i^2 + 2(a_2 + 7)\lambda_i + (7a_2 + 3.75 + a_1)}$$
$$\frac{d\lambda_i}{da_2} = - \frac{\lambda_i^2 + a_1}{3\lambda_i^2 + 2(a_2 + 7)\lambda_i + (7a_2 + 3.75 + a_1)} \qquad \text{(E3.5)}$$

The results of machine solution are given below

	Parameter Vector		Performance Measure
	a_1	a_2	
Initial	2	1.0	0.87689
Final	6	1.1	0.17309

giving the parameters for the minimum performance measure.

2.4 Time Weighted Optimization

A more general measure of performance is

$$J = \int_{t_o}^{t_1} g(t) f_o(x, u) dt \qquad (40)$$

where g(t) is a time weighting function. It can be shown that if g(t) is continuous and f_o is in $C(\underline{\bar{X}} \times \bar{U})$, i.e., if it is defined and is continuous on a suitable domain $\underline{\bar{X}} \times \bar{U}$, then J is continuously differentiable with respect to w in this domain. In that case (7) is applicable and the procedure can be carried out as indicated. For efficient use of (7) it is necessary that explicit relations be available to evaluate J. If the weighting function g(t) is of the form

$$g(t) = \sum_k (C_o^k + \ldots + C_\ell^k t^\ell) \tag{41}$$

then explicit formulas for J can be obtained.

Some of the weighting functions commonly used are

$$\text{i)}\quad g(t) = \begin{cases} 1 \text{ for } 0 \leq t \leq t_1 \\ 0 \text{ for } \quad t \geq t_1 \end{cases} \tag{42}$$

$$\text{ii)}\quad g(t) = t \tag{43}$$

$$\text{iii)}\quad g(t) = e^{\alpha t} \tag{44}$$

If the input is a step function, then the time weighted integral-quadratic error for these three cases can be shown to be

i) For g(t) given in (42)

$$J_I = \sum_{i=1}^{n} \sum_{k=1}^{n} Z_i{}'GZ_k \frac{e^{(\lambda_i + \lambda_k)t_1}}{(\lambda_i + \lambda_k)\lambda_i\lambda_k} - \frac{1}{(\lambda_i + \lambda_k)\lambda_i\lambda_k} \tag{45}$$

The evaluation is done in real numbers if all eigenvalues are real. If some of the eigenvalues are complex, the calculations involved are complex. However, formulas in the real domain can be obtained as was done in (34).

ii) For g(t) given in (43)

$$J_{II} = \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{Z_i{}'GZ_k}{\lambda_i\lambda_k} \frac{1}{(\lambda_i + \lambda_k)^2} \tag{46}$$

where it has been assumed that all eigenvalues of A have a negative real part.

iii) and for g(t) given in (44)

$$J_{III} = \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{Z_i' G Z_k}{\lambda_i \lambda_k (\alpha + \lambda_i + \lambda_k)} \tag{47}$$

where again it has been assumed that all eigenvalues of A have a negative real part and

$$(\alpha + \max \text{Re}\ \{\lambda_i\}) < 0$$

i.e.

$$a < - \max \text{Re}\ \{\lambda_i\}$$

2.5 Stabilization of a Linear System

If the system is initially unstable, many of the error criterions used as performance index fail to apply. For example, the integral square error over the infinite time interval, used in Section 2.3, has meaning only if the system is stable. Moreover, it is necessary that the system resulting at the end of optimization, with respect to a certain performance index, be stable. Thus, a computer procedure for stabilizing the system initially is desirable.

It is well known that the equilibrium state, x = 0, of the linear homogeneous system

$$\dot{x} = Ax \tag{48}$$

is asymptotically (globally) stable if and only if all eigenvalues of the matrix $\underline{A}$ have negative real parts, i.e.,

$$\text{Re}\ \lambda_i < 0 , \quad i = 1, \ldots, n \tag{49}$$

Here the stability is in the Liapunov sense[19] as given by the following definitions.

Definition 1

The equilibrium state of the differential equation

$$\dot{x} = f(x, t) \tag{50}$$

is said to be stable if for every $\epsilon > 0$ there exists a number $\delta > 0$ such that

$$|x^o| < \delta \tag{51}$$

implies

$$|x(t, x^o, t_o)| < \epsilon \text{ for all } t \geq t_o \tag{52}$$

Definition 2

The equilibrium state of (16) is said to be asymptotically stable if

i) it is stable (Definition 1)

ii) and there exists a $\delta_o > 0$ such that for any x^o, such that $|x^o| < \delta_o$,

$$\lim_{t \to \infty} x(t, x^o, t_o) = 0$$

Definition 3

If the number δ_o in Definition 2 is non-finite, then the equilibrium state is said to be globally asymptotically stable.

The problem of stabilization can be stated more precisely as follows. Find the $\underline{q}_a$ dimensional parameter vector $\underline{a}$ from the closed and bounded set $\overline{W}_a$ which stabilizes the matrix A(a). The parameter optimization procedure given above can be used to stabilize an unstable system by using the criterion function

$$J = \sum_{i=1}^{n} e^{\text{Re}\,\lambda_i} \tag{53}$$

This function has an isolated absolute minimum at

$$\text{Re}\,\lambda_i = -\infty, \; i = 1, \dots n$$

For this function (7) reduces to

$$\frac{\partial J}{\partial a_j} = 2 \sum_{i=1}^{m} e^{\alpha_i} \frac{\partial \alpha_i}{\partial a_j} + \sum_{i=2m+1}^{n} e^{\lambda_i} \frac{\partial \lambda_i}{\partial a_j} \tag{54}$$

where, from (A18)

$$\frac{\partial \alpha_i}{\partial a_j} = - \frac{1}{\left(\frac{\partial p_1}{\partial \alpha_i}\right)^2 + \left(\frac{\partial p_2}{\partial \alpha_i}\right)^2} \cdot \left[\frac{\partial p_2}{\partial \beta_i} - \frac{\partial p_1}{\partial \beta_i}\right] \begin{bmatrix} \frac{\partial p_1}{\partial a_j} \\ \\ \frac{\partial p_2}{\partial a_j} \end{bmatrix} \tag{55}$$

$$\frac{\partial \lambda_i}{\partial a_j} = - \frac{\frac{\partial \delta(\lambda_i, a_j)}{\partial a_j}}{\frac{\partial \delta(\lambda_i, a_j)}{\partial \lambda_i}} \tag{56}$$

Following the method of steepest descent, the solution to the set of equations

$$\frac{da}{d\sigma} = -k\left(\frac{\partial J}{\partial a}\right)' \tag{57}$$

for increasing σ leads to a parameter vector $\underline{a}$ for which the system is stable.

<u>Example 4</u>

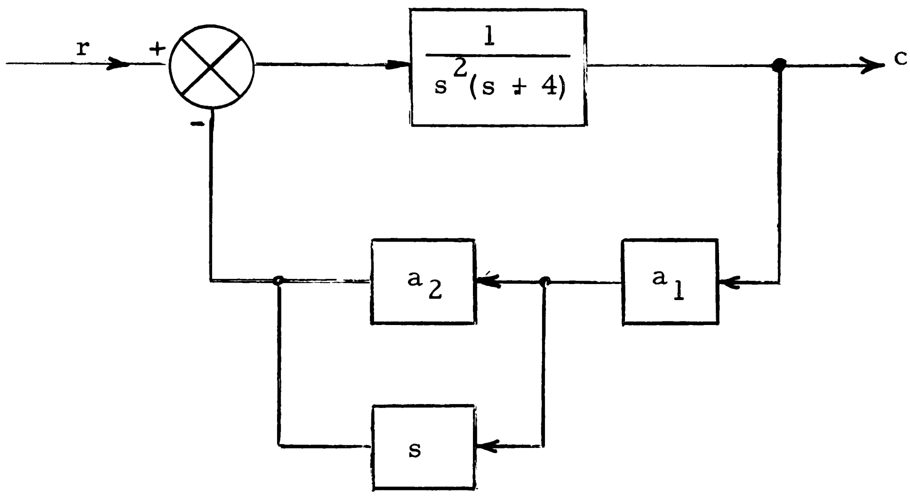


Fig. 5

Consider the system represented by the block diagram in Fig. 5, or the state model

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & a_1 \\ -a_2 & -1. & -4. \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{E1.1}$$

with the initial parameter values $a_1 = 1$ and $a_2 = -6$ and let the set $\overline{W}_a$ in which these parameters are to be restricted be

$$0.5 \leq a_1 \leq 1.1$$
$$-2.0 \leq a_2 \leq 7.0 \qquad \text{(E1. 2)}$$

The characteristic polynomial of the matrix is

$$\delta(\lambda) = \lambda^3 + 4\lambda^2 + a_1\lambda + a_1 a_2 \qquad \text{(E1. 3)}$$

the eigenvalues for the initial parameter vector are

$$\lambda_1 = 1.0$$
$$\lambda_2 = -2.0$$
$$\lambda_3 = -3.0$$

and the system is clearly unstable. For this simple problem, it is obvious that $\underline{a}_2$ being negative leads to a non-Hurwitz characteristic polynomial.

To stabilize this system using the theory in this Section, evaluate the matrix $\frac{\partial\lambda^*}{\partial a}$ of equation (11) as

$$\frac{\partial\lambda^*}{\partial a} = \begin{bmatrix} -\dfrac{(\lambda_1 + a_2)}{3\lambda_1^2 + 8\lambda_1 + a_1} & -\dfrac{a_1}{3\lambda_1^2 + 8\lambda_1 + a_1} \\ \\ -\dfrac{(\lambda_2 + a_2)}{3\lambda_2^2 + 8\lambda_2 + a_1} & -\dfrac{a_1}{3\lambda_2^2 + 8\lambda_2 + a_1} \\ \\ -\dfrac{(\lambda_3 + a_2)}{3\lambda_3^2 + 8\lambda_3 + a_1} & -\dfrac{a_1}{3\lambda_3^2 + 8\lambda_3 + a_1} \end{bmatrix}$$

The eigenvalues for any given parameter vector are evaluated from the initial eigenvalues by quadrature integration of (A16) which in this case are specifically

$$\frac{d\lambda_i}{da_1} = -\frac{(\lambda_i + a_2)}{3\lambda_i^2 + 8\lambda_i + a_1}$$

$$\frac{d\lambda_i}{da_2} = -\frac{a_1}{3\lambda_i^2 + 8\lambda_i + a_1}$$

The computer solution based on steepest descent gives

$$a_1 = 0.5$$

$$a_2 = -0.0289$$

for which

$$\lambda_1 = -.0445$$

$$\lambda_2 = -.0835$$

$$\lambda_3 = -3.87$$

3. TIME OPTIMAL CONTROL PROBLEMS

3.1 "Bang-Bang" Parameter Optimization

The problem of time optimization for linear systems, i.e., reaching the prescribed final state $x(t_1) = 0$ from a given initial state $x(t_o) = x^o$, through the manipulation of the control function u has been the subject of numerous investigations.[1-2, 21-27, 30]

In the following development, the possibility of manipulating some of the system parameters, which can possibly be changed a finite number of times (possibly only at certain discrete levels), in conjunction with control manipulation is considered. It is shown that if the parameter region is a closed parallelepiped the optimum transition time results when the parameters assume values only on the boundary of $\overline{W}$. Furthermore, it turns out that the parameter vector $\underline{w}$ is a constant except at a finite number of instants of time where it undergoes a jump. This result is based on the Maximum Principle of Pontryagin. This basic theorem is given along with a proof based on the calculus of variations in Appendix B.

The time optimal problem as considered here can be stated as follows. Let w(t) and u(t) represent vectors of dimensions $\underline{q}$ and $\underline{r}$ respectively from the class of piecewise continuous functions, which assume values in the compact sets $\overline{W}$ and $\overline{U}$. From this class of functions, select $\underline{w}(t)$ and $\underline{u}(t)$ such that

$$J = \int_o^{t_1} f_o \, dt = t_1, \quad x(t_1) = 0 \tag{58}$$

takes on a minimum value for a given initial state $x(0) = x_o$ and for the terminal state $x(t_1)$, obtained as a solution to

$$\dot{x} = A(w(t))\; x + B\; u(t) \tag{59}$$

Following a procedure similar to that used by Pontryagin and others, let (58) and (59) be written as

$$\dot{x} = 1$$

$$\dot{x} = f(x,\; v)$$

where v is the product vector (w, u) which takes on values in the product set $\overline{W} \times \overline{U}$ in an q + r space. The Hamiltonian function appearing in theorem B1 is

$$\underline{H} = \Psi_o\, f_o + \Psi' f = \Psi_o + \Psi' (A(w)x + Bu)$$

By the maximal principle, the optimum w and u satisfy the functional equality

$$\Psi'(A(w) + Bu) \quad = \quad \sup_{\substack{w \in \overline{W} \\ u \in \overline{U}}} \Psi'(A(w)x + Bu)$$

or

$$\Psi' A(w)x \quad = \quad \sup_{w \in \overline{W}} \Psi' A(w)x \tag{60}$$

and

$$\Psi'\, B\, u \quad = \quad \sup_{u \in \overline{U}} \Psi'\, Bu \tag{61}$$

Consider the case when the entries in the vector w are identified with distinct entries in A. Extensions to the case where entries in w correspond to distinct entries in B are straightforward. Let $\overline{W} \times \overline{U}$ be a polyhedron and assume that the optimal w(t) and u(t) exist. Then

<u>Theorem:</u> Equations (60) and (61) uniquely* determine the optimal w(t) and u(t) for each non-trivial solution of the adjoint system

$$\dot{x} = - A'(w(t))x \tag{62}$$

if for every vertex of $\overline{W}$, A(w) satisfies the condition of complete controllability and there are no critical** points of $\underline{x}$. In addition, w(t) and u(t) are piecewise constant and assume values at the vertices of the polyhedrons $\overline{W}$ and $\overline{U}$.

<u>Proof:</u> The function

$$\Psi' A(w)x + \Psi' Bu \tag{63}$$

is linear in (w, u) and is either a constant or attains its maximum on the boundary of $\overline{W} \times \overline{U}$. Thus, it assumes its maximum at a vertex of $\overline{W} \times \overline{U}$ or one of its entire faces. We wish to show that the latter happens only for a finite number of values of time.

---

* Unique to the extent that at the points of jumps u(t) is arbitrary between u^- and u^+.

** A point where $\left.\begin{array}{l}\Psi_i = 0 \\ \text{or } \dot{x}_j = 0\end{array}\right\}$

and $\left.\begin{array}{l}\dot{\Psi}_i = 0 \\ \text{or } x_j = 0\end{array}\right\}$

Assume the contrary; i.e., let there exist an infinite set of such times t, $t_o \leq t \leq t$. Since there are only a finite number of faces of $\overline{W} \times \overline{U}$, the function (63) assumes its maximum for an infinite set $\underline{M}$ of times on a certain face L. Two cases now arise.

If L is parallel to $\overline{U}$, then from the results of Pontryagin[2] a contradition is obtained if the system is completely controllable. If L is parallel to $\overline{W}$, let $v_1 = (w_1, u)$ and $v_2 = (w_2, u)$ be vectors to the endpoints of some edge Γ. Then for $t \in \underline{M}$, we have

$$\Psi' A(w_1)x + \Psi'Bu = \Psi'A(w_2)\, x + \Psi'Bu$$

$$\Psi'A(w_1)x - \Psi'A(w_2)x = 0$$

$$\Psi' A(w_1 - w_2)x = 0$$

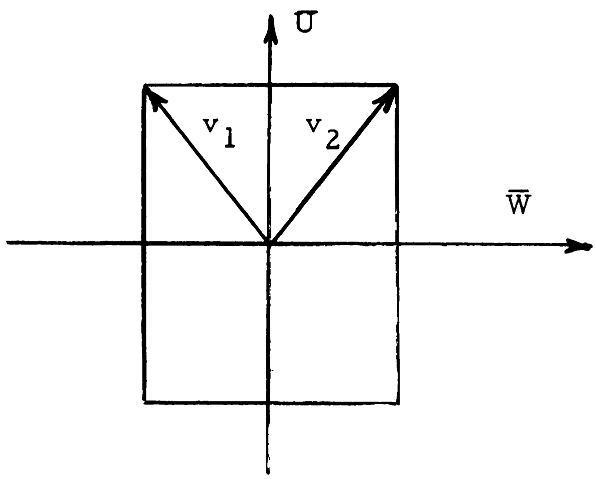


Fig. 6

But since $w_1 - w_2$ is an edge of $\overline{W}$, it is a vector whose components are all zero except one and this one forms some entry i, j of the matrix A. Thus

$$[\quad \Psi' \quad] \begin{bmatrix} 0 & 0 & \overset{j}{|} & . & 0 \\ 0 & 0 & | & & \\ . & & | & & \\ i \; - & - & a_{ij} & - & - \\ . & & | & & \\ . & & | & & \\ 0 & 0 & . \; | & . & 0 \end{bmatrix} \begin{bmatrix} \\ x \\ \\ \end{bmatrix} = 0$$

and it follows that $\Psi_i(t) \;\; a_{ij} x_j(t) = 0$ (64)

The set $\underline{M}$ must have at least one limit point τ since the interval (t_o, t_1) is finite. Therefore, there exists a monotone sequence of points $\tau_1, \ldots, \tau_k, \ldots$ converging to τ. Since $x(t)$ and $\Psi(t)$ are continuous functions

$$\lim_{k \to \infty} \Psi_i(\tau_k) \, x_j(\tau_k) = \Psi_i(\tau) \, x_j(\tau) = 0 \tag{65}$$

Furthermore, between every two roots of (64) the derivative, the left hand side of (64) has a zero. Let these points be denoted $\tau_1^*, \ldots, \tau_k^*, \ldots$. The limit of this sequence also converges to τ.

Since the derivative of (64) vanishes at each point τ_k^*, we have

$$\dot{\Psi}_i(\tau_k^*) x_j(\tau_k^*) + \Psi_i(\tau_k^*) \dot{x}_j(\tau_k^*) = 0 \tag{66}$$

where from the system and adjoint equations

$$\Psi_i(\tau_k^*) = - \Psi' (\tau_k^*) \, A^i(w(\tau_k^*))$$

$$\dot{x}_j(\tau_k^*) = A_j(w(\tau_k^*)) \, x\, (\tau_k^*)$$

The subscripts and superscripts denote the row and column respectively of the indicated matrices. Substituting these relations in (66) and using the one-sided continuity of w(t), it follows that

$$\lim_{k \to \infty} \{\Psi'(\tau_k^*)A^i(w(\tau_k^*))x_j(\tau_k^*) + \Psi'(\tau_k^*)[A_j(\tau_k^*)x(\tau_k^*)+B_ju(\tau_k^*)]\}$$

$$= \Psi'(\tau)A^i(w(\tau))x_j(\tau) + \Psi_i'(\tau)[A_j(\tau)x(\tau) + B_ju(\tau)]$$

$$= \dot{\Psi}_i(\tau)\, x_j(\tau) + \Psi_i(\tau)\dot{x}_j(\tau) = 0 \tag{67}$$

But this result is a contradiction to the hypothesis and the proof is complete.

When the boundaries of $\overline{W}$ and $\overline{U}$ are independent, that is when

$$\begin{aligned} &\alpha_k \leq w_k \leq \beta_k, \; k = 1, \ldots, q \\ &\alpha_k^* \leq u_k \leq \beta_k^*, \; k = 1, r \end{aligned} \tag{68}$$

it follows from (60) and (61) that the parameters and control are

$$w_k = \frac{\beta_k + \alpha_k}{2} + \frac{\beta_k - \alpha_k}{2} \text{ sign } \Psi_i x_j \quad k = 1, \ldots r \tag{69}$$

$$u_k = \frac{\beta_k^* + \alpha_k^*}{2} + \frac{\beta_k^* - \alpha_k^*}{2} \text{ sign } \Psi' b_k \tag{70}$$

where w_k is the i,j entry in A and b_k is the kth column of the matrix B.

Example 5

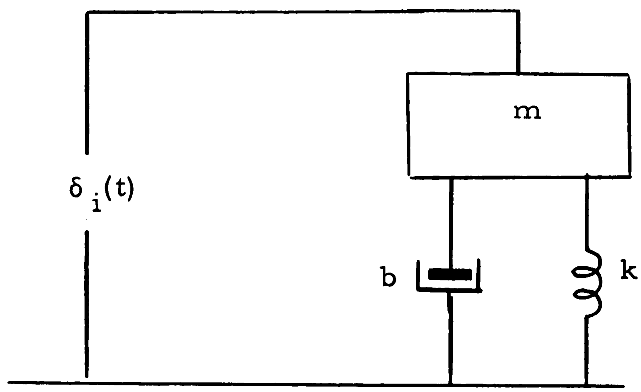


Fig. 7

Consider the system shown in Fig. 7. The state model is

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & a \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$-1.8 \leq a \leq 0$$

$$-1 \leq u(t) \leq +1$$

The Hamiltonian function is

$$H = -1 + [\Psi_1 \quad \Psi_2]\left\{\begin{bmatrix} 0 & 1 \\ -1 & a \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u\right\}$$

$$= -1 + \Psi_1 x_2 + \Psi_2(-x_1 + ax_2 + u)$$

From which it follows using (49) and (50)

$$u = \text{sign}\, \Psi_2$$

$$a = \begin{cases} -1.8 & \text{if } \Psi_2 x_2 < 0 \\ 0 & \text{if } \Psi_2 x_2 > 0 \end{cases}$$

Time optimal trajectories for fixed $\underline{a} = -1.8$ and $\underline{a} = 0$ are in Figs. 8 and 9 respectively. Numbers marked along the trajectories give the time in seconds to reach the origin. Trajectories optimized with respect to control and bang-bang parameters are shown in Fig. 10. A comparative picture of the trajectories for $\underline{a} = -1.8$, $\underline{a} = 0$ and for bang-bang parameters for a specific initial state x^o is shown in Fig. 11. The results are shown below.

Initially:

$$x^o = \begin{bmatrix} x_1^o \\ x_2^o \end{bmatrix} = \begin{bmatrix} 5.4 \\ 10.5 \end{bmatrix}$$

$$\Psi^o = \begin{bmatrix} \Psi_1^o \\ \Psi_2^o \end{bmatrix} = \begin{bmatrix} -.126 \\ -.142 \end{bmatrix}$$

$$a = -1.8$$

$$u = -1$$

At time $t = 1.095$ parameter switches to $a = 0$

At time $t = 2.085$ control switches to $u = 1$

parameter switches to $a = -1.8$

Final state time $t_1 = 3.785$ sec.

Comparative timings are

	Transition time
a fixed at -1.8	4.37
a fixed at 0	20.9
Bang-Bang parameters	3.785 sec.

3.2 Time-Varying Mass Parameters

It is not unusual that some of the system parameters undergo change with an integral relation to the controls exercised. For example, the mass parameters of a space vehicle decrease with continuous fuel consumption.

Thus, it is required to find a vector function u(t) from the class of piecewise continuous functions which take their values in the compact set $\bar{U}$ of r-dimensional vectors such that

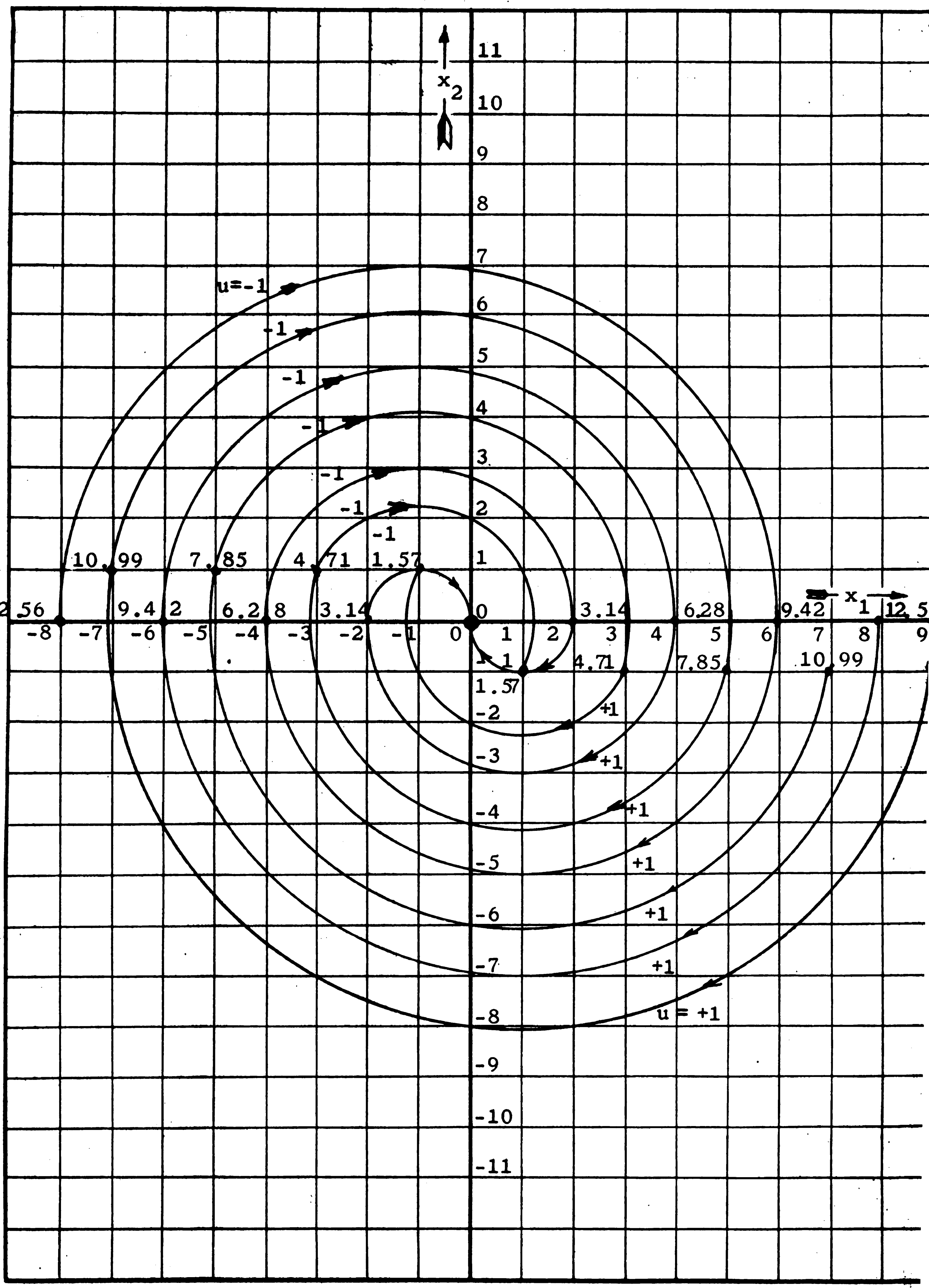
$$J = \int_o^{t_1} f_o dt = t_1, \quad x(t_1) = 0$$

takes on a minimum value subject to the constraint.

$$\dot{x} = A(w(t))\, x + B(w(t))\, u, \qquad x(t_o) = x^o \tag{71}$$

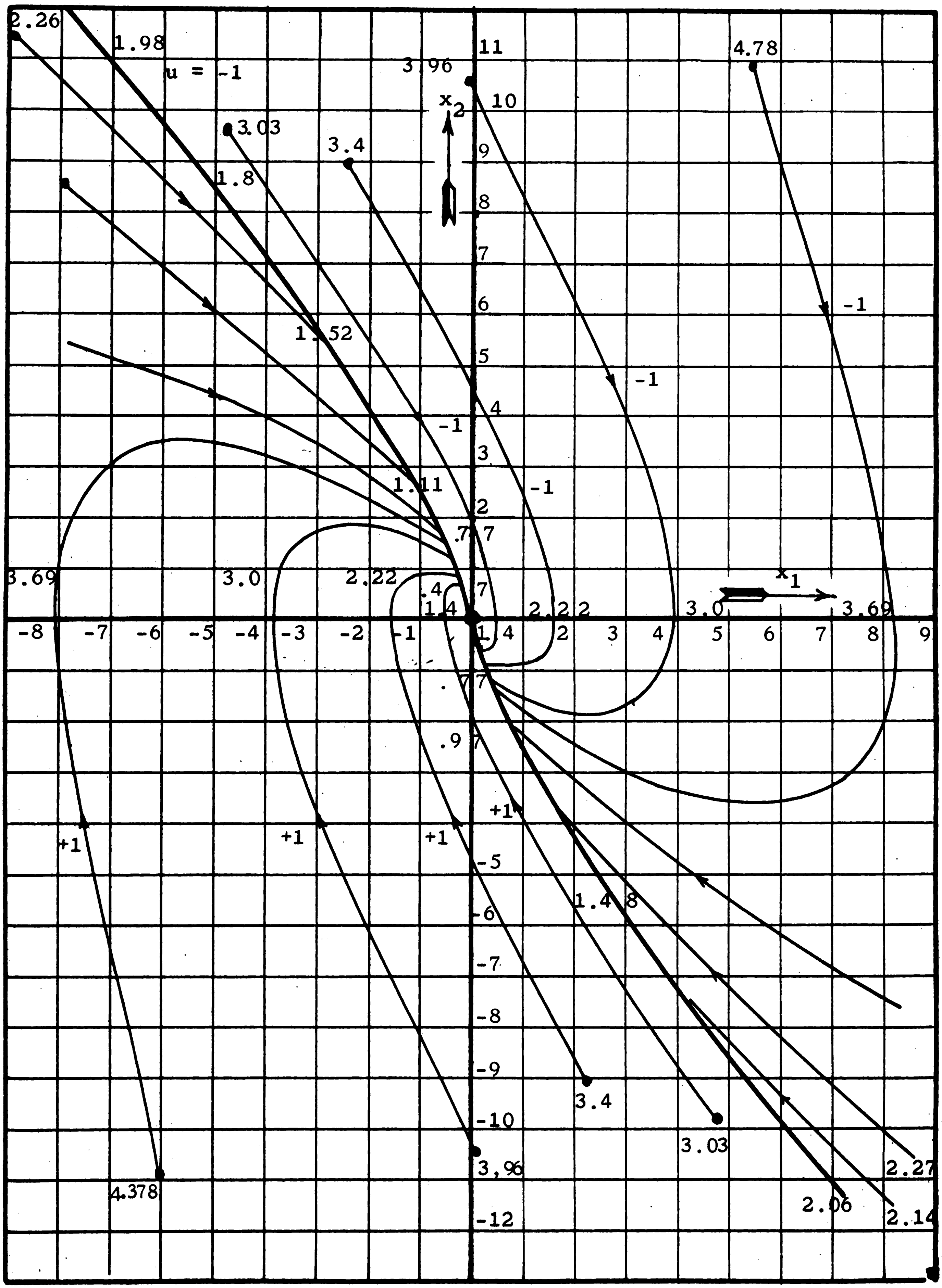
where the components of the vector w are given by

Fig. 8



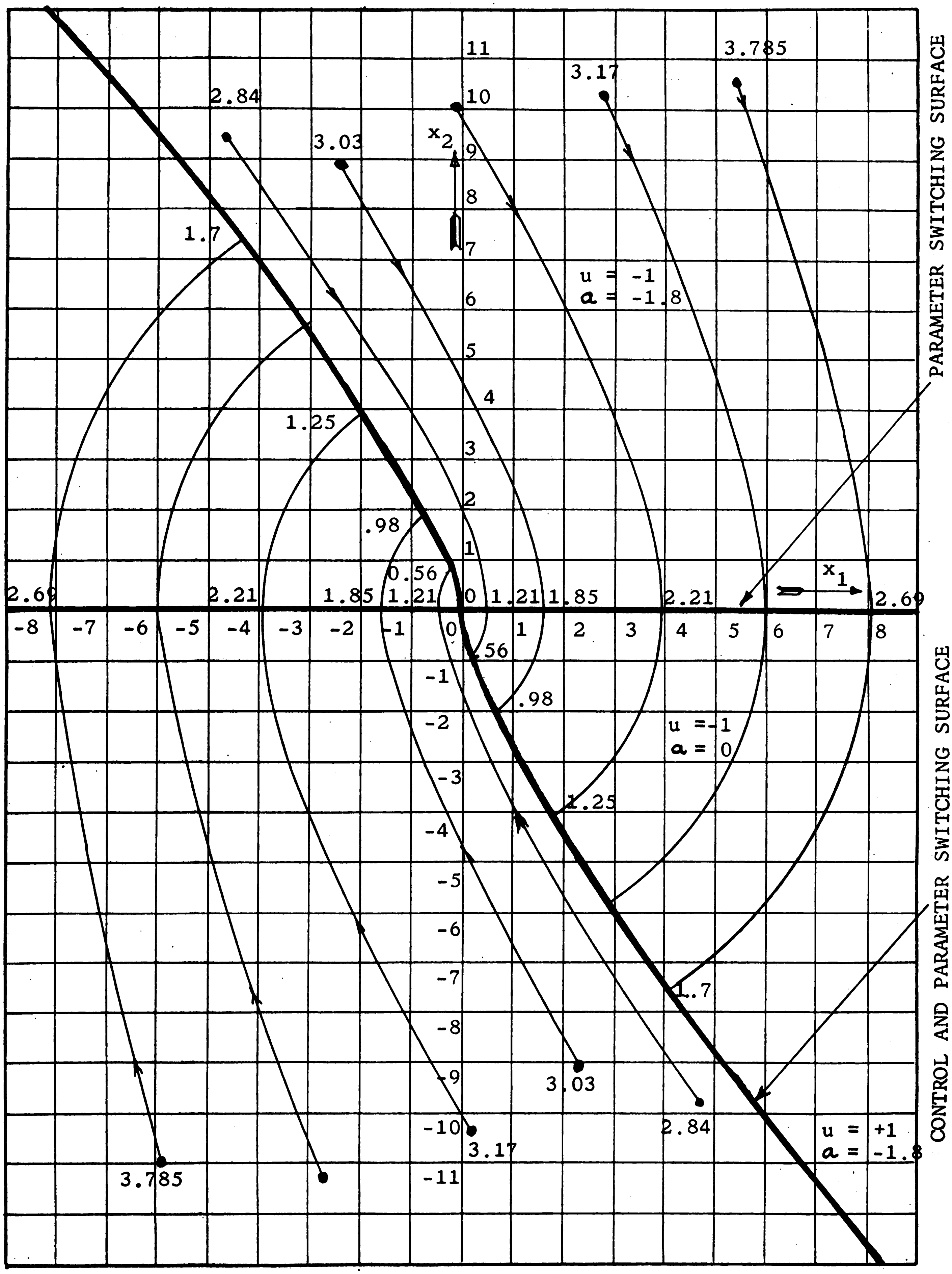
TRAJECTORIES FOR PARAMETER VALUE $a = 0$

Fig. 9



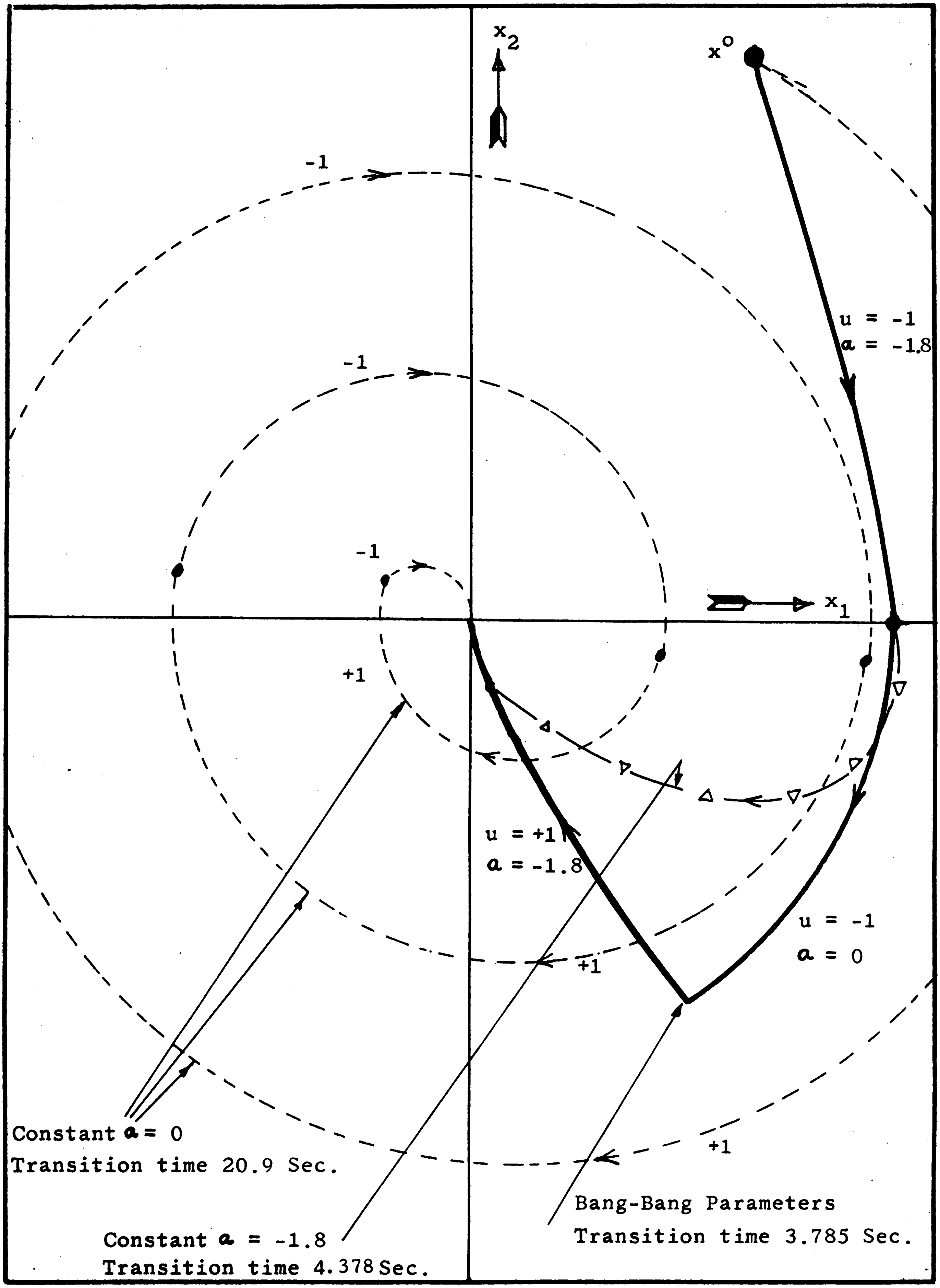
TRAJECTORIES FOR PARAMETER VALUE $a = -1.8$

Fig. 10



TRAJECTORIES FOR BANG-BANG PARAMETERS

Fig. 11



COMPARISON OF TRANSITION TIME

$$w_k(t) = 1 - \int_o^t g_k(u(t))dt \quad k = 1, \ldots q \tag{72}$$

$$\left.\begin{aligned} &w_k(0) = 1 \\ &w_k(t_1) = \theta_k, \quad 0 < \theta_k < 1 \end{aligned}\right\} k = 1, \ldots, q \tag{73}$$

Here g_k, $k = 1, \ldots, q$ are positive definite functions of u.

We differentiate (72) and consider the augmented system

$$\begin{aligned} \dot{x} &= A(w(t))\, x + B(w(t))\, u \\ \dot{w} &= \qquad\qquad - g(u) \end{aligned} \tag{74}$$

Denoting the augmented vector (x, w) as y, (74) can be written as

$$\dot{y} = h(y, u) \tag{75}$$

The equivalent problem is to transfer the state of the autonomous system (75), from $y^o = (x^o, \underline{1})$ at time t_o to $y^1 = (x^1, \theta)$. The Hamiltonian function $H(\bar{\bar{\Psi}}, y, u)$ is

$$H = -\Psi_o + \bar{\bar{\Psi}}' h = -\Psi_o + \bar{\bar{\Psi}}' \begin{bmatrix} A(w)x + B(w)u \\ g(u) \end{bmatrix} \tag{76}$$

Treating this as a function of u, the optimum control according to the maximum principle, is given by

$$\bar{\bar{\Psi}}' \begin{bmatrix} B(w)u \\ g(u) \end{bmatrix} = \sup_{u^* \in u} \bar{\bar{\Psi}}' \begin{bmatrix} B(w)u^* \\ g(u^*) \end{bmatrix} \tag{77}$$

where $\bar{\bar{\Psi}} = (\Psi, \bar{\Psi})$. is a non-zero solution to the adjoint system

$$\dot{\Psi} = - A'(w(t))\Psi$$

$$\dot{\bar{\Psi}} = -\left[\frac{\partial A(w)x(t)}{\partial \quad w}\right]' \Psi - \left[\frac{\partial B(w)u(t)}{\partial \quad w}\right] \Psi \tag{78}$$

and Ψ and $\bar{\Psi}$ are n and q dimensional vectors respectively.

<u>Theorem:</u> If $g_k \equiv |u_k|$ and $|u_k| \leq 1$, $k = 1, \ldots, r$ for the problem above the piecewise continuous controls u_k assume only the values -1, 0, +1.

<u>Proof :</u> The equation (77) takes on the form

$$\Psi' B(w)u + \bar{\Psi}' \begin{bmatrix} |u_1| \\ \cdot \\ \cdot \\ |u_r| \end{bmatrix} = \max_{\substack{|u_k^*| \leq 1 \\ k = 1, \ldots, r}} \left\{ \Psi' B(w)u^* + \bar{\Psi}' \begin{bmatrix} |u_1^*| \\ \\ \\ |u_r^*| \end{bmatrix} \right\}$$

and the expression

$$\sum_{k=1}^{r} \sum_{i=1}^{n} \Psi_i b_{ik} u_k - \sum_{k=1}^{n} \Psi_{n+k} \; |u_k|$$

must assume a maximum with respect to u_k, $k = 1, \ldots r$. But this function assumes a maximum when for each k, $k = 1, \ldots r$, the function

$$\sum_{i=1} \Psi_i b_{ik} u_k - \Psi_{n+k} \; |u_k| \tag{79}$$

assumes its maximum. Letting

$$\sum_{i=1}^{n} \Psi_i b_{ik} = \sigma_k \tag{80}$$

it can be shown that (79) assumes its maximum when

Case 1. $\Psi_{n+k} \leq 0$: then $u_k = \text{sign}\, \sigma_k$

Case 2. $\Psi_{n+k} > 0$: then $u_k = 0$ for $\Psi_{n+k} > |\sigma_k|$

$\text{sign}\, \sigma_k$ for $\Psi_{n+k} < |\sigma_k|$

Although the above theorem indicates that the optimum control takes on its values either at the boundaries +1 or -1, or is 0, the problem of finding the switching instants remains to be solved. And in this sense this problem has been carried to the same state as the problem of finding the switching time for the usual optimum time problem. For the latter problem, some investigators[22,24] have developed methods of obtaining these switching instants through the solution of certain transcendental equations. The same procedures can possibly be extended to the solution of the switching problem defined above. However, the possible number of values for each control is now three instead of two.

4. THE MAXIMUM PRINCIPLE IN FIXED TIME PROBLEMS

4.1 Finite Transition Time

For automatic control systems, the final state of the system is usually specified to be a specific value or within a region of specified values. For certain economic and possibly other systems, it is only the object function, over a fixed time duration, which is of interest and the final state is arbitrary. To formulate the problem of minimizing the object function for this case, consider that (1) has fixed parameters and the matrix A has distinct eigenvalues with negative real parts. It is required to find control vector u(t), from the class of measureable functions assuming values in a compact set $\bar{U}$ which minimizes (3) for the initial state $\underline{x}_o$, given that $\underline{f}_o$ in (3) is in classes $C(\underline{\bar{X}} \times \bar{U})$ and $D(\underline{\bar{X}})$.

Since u assumes its value in a compact set, the object function J, in general, is not differentiable with respect to x_o and Bellman's[11] equation

$$\frac{\partial J}{\partial t_1} = \underset{u}{\text{Min}} \left[f_o(x^o, u) + \frac{\partial J}{\partial x^o} f(x^o, u)\right] \tag{81}$$

is not applicable. However, Pontryagin's maximum principle is still applicable and states as necessary conditions[2].

<u>Theorem P1</u> Consider a system with fixed initial state x^o and free final state x^1. An admissible control u(t), $t_o \leq t \leq t_1$, and the corresponding trajectory are optimal only if there exists a non-zero continuous vector function $\Psi(t)$ corresponding to the functions u(t) and x(t) such that

(1) for all t, $t_o \leq t \leq t_1$, the Hamiltonian function $H(\underline{\Psi}(t), x(t), u)$ of the variable $u \in \overline{U}$ attains its maximum at the point $u = u(t)$

$$H = M = \sup_{u \in \overline{U}} H(\underline{\Psi}(t), x(t), u) \tag{82}$$

(2) $$\underline{\Psi}(t_1) = (-1, 0, \ldots, 0) \tag{83}$$

If there exists an optimal control and if the final state $x(t_1)$ is known, then it is possible to solve for the control u(t) explicitly. When $x(t_1)$ is not known, it is possible to proceed as follows: Consider the object function

$$J = \int_o^{t_1} f(x,u)\,dt = \int_o^{t_1} (x'Gx + u'\,Cu)\,dt \tag{84}$$

or

$$\dot{x}_o = x'GX + u'Cu, \qquad x_o(t_1) = J \tag{85}$$

where

$$|u_i| \leq 1, i = 1, \ldots, n \tag{86}$$

and G and C are nxn and rxr diagonal weighting matrices. The Hamiltonian function is

$$H(\underline{\Psi}(t), x(t), u) = \Psi_o f_o + \Psi' f$$

$$= \Psi_o (x'Gx + u'Cu) + \Psi'(Ax + Bu) \qquad (8\ \ )$$

$$\underline{\Psi} = (\Psi_o, \Psi')$$

and Ψ_o and the n-vector Ψ are solutions to the adjoint system

$$\Psi_o = 0 \tag{88}$$

$$\Psi = -2Gx - A'\Psi \tag{89}$$

From (87), u is optimal if

$$-u'\,Cu + \Psi'\,Bu = -\sum_{i=1}^{r} c_{ii}u_i^{\,2} + \sum_{i=1}^{r} \Psi' b_i u_i$$

$$= -\sum_{i=1}^{r} c_{ii}\left(u_i - \frac{\Psi' b_i}{2c_{ii}}\right)^2 + \sum_{i=1}^{r} \left(\frac{\Psi' b_i}{2c_{ii}}\right)^2 \tag{90}$$

assumes its maximum at the point u for all t. Here $\underline{b}_i$, i = 1, . . . , n is the ith column of $\underline{B}$. The expression (90) is maximum if

$$u_i = \begin{cases} \dfrac{\Psi' b_i}{2c_{ii}} & \text{when } \left|\dfrac{\Psi' b_i}{2c_{ii}}\right| \leq 1 \\[2ex] 1 & \text{when } \dfrac{\Psi' b_i}{2c_{ii}} \geq 1 \\[2ex] -1 & \text{when } \dfrac{\Psi' b_i}{2c_{ii}} \leq -1 \end{cases}$$

$$i = 1, \ldots\ r \tag{91}$$

Assuming the existence (which must be ascertained from the Physics of the problem) of a non-trivial optimal control, the following theorem establishes that the control is linear over a non-zero interval of time $t_2 \leq t \leq t_1$.

Theorem 1: For the control in (91), there exists a t_2, $0 \leq t_2 \leq t_1$, such that on (t_2, t_1), $u_i = \frac{\Psi' b_i}{2c_{ii}}$ and $|u_i| < 1$, $i = 1, \ldots, r$. Moreover, $u \not\equiv 0$ on (t_2, t_1).

Proof: From Theorem P1, $\Psi_o^1 = -1$, $\Psi^1 = 0$. Since $\Psi \not\equiv 0$ (for otherwise the control is trivial) there exists a τ, $0 < \tau \leq t_1$, such that on $(\tau - \epsilon, \tau)$, $\Psi \neq 0$ and on $[\tau, t_1]$, $\Psi \equiv 0$.

Denote the open sphere $S_i = \{ \Psi : |\frac{\Psi' b_i}{2c_{ii}}| < 1 \}$, and let $S = \bigcap_{i=1}^{r} S_i$. Using the continuity of Ψ at τ the inverse image of S is a non-empty open interval $(\tau - \delta, \tau)$ on which $|\frac{\Psi' b_i}{2c_{ii}}| < 1$ and

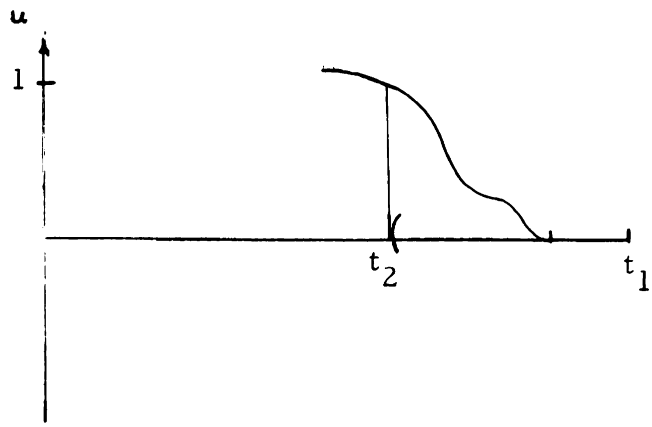


Fig. 12

$\frac{\Psi' b_i}{2c_{ii}} \neq 0$, $i = 1, \ldots, r$. Denote the inf. of $\tau - \delta$ as t_2 as in Fig. 12. Then on (t_2, t_1), we have $|u_i| < 1$, $i = 1, \ldots, n$ and $u \not\equiv 0$.

Structure of the Control

From the control equations (91), it follows that the composite system

$$\dot{z} = Dz + Cd \tag{92}$$

where $\underline{z} = (x, \Psi)$, has 2^r possible forms, i.e.,

$$D = \left[\begin{array}{c|c} A & \sum_{1 \le i_k \le r} \frac{b_{i_k} b_{i_k}'}{2c_{i_k i_k}} \\ \hline 2G & -A' \end{array}\right]$$

$$C = \begin{bmatrix} \\ b_{j_1} \\ \\ \end{bmatrix} \qquad d = \begin{bmatrix} - - - - - \\ d_{j_1} \\ - - - - - \\ \\ \end{bmatrix} \quad d_{j_k} = \text{Sign} \frac{b_{j_k}' \Psi}{2c_{j_k j_k}} \tag{94}$$

where the indices i_k, j_k are mutually exclusive and complete the indices 1 through r.

Lemma 3 Every two zeros on $[0, t_2]$ of the function $\mu_i^{\pm}(t) = \frac{b_i' \Psi(t)}{2c_{ii}} \pm 1$, $i = 1, \ldots, n$, are separated by a non-zero time interval or otherwise $\mu_i^{\pm}(t) \equiv 0$ on some non-zero sub-interval of $[0, t_2]$.

Proof: Assume τ and τ' be two zeros of the function and let $\tau < \tau'$. A unique control is determined for τ^+ and correspondingly a unique composite system. The solution Ψ is locally analytic since the composite system is linear. Consequently, for a sufficiently small $\epsilon > 0$

$$\mu_i^{\pm}(\tau+\epsilon) = \mu_i^{\pm}(\tau) + \frac{d\mu_i^{\pm}(\tau)}{dt} + \frac{\epsilon^2}{2!}\frac{d^2}{dt^2}\mu_i^{\pm}(\tau) + \ldots$$

$$= \epsilon \left[\frac{d}{dt}\mu^{\pm}(\tau) + \ldots \right]$$

Thus if $\mu_i^{\pm}(t) \not\equiv 0$ on $(t, t+\epsilon)$ then $\frac{d}{dt}\mu^{\pm}(\tau) \neq 0$ and on some interval (τ, τ'), $\mu_i^{\pm}(t)$ does not vanish.

Using Lemma 3, it follows that $\left| \frac{b_i'\Psi}{2c_{ii}} \right|$, $i = 1,..r$, can equal 1 only a finite number of times on $[0, t_2]$.

(A) Two cases must be considered

(i) If on $[0, t_2]$, $\left| \frac{b_i'\Psi}{2c_{ii}} \right| \equiv \pm 1$, $i = 1, \ldots, r$, then

$$u_i \equiv \pm 1, \; i = 1, \ldots, r, \text{ on } [0, t_2].$$

(2) If for some i, $1 \leq i \leq r$, and some τ, $0 < \tau < t_2$,

$$\left| \frac{b_i'\Psi}{2c_{ii}} \right| < 1 \text{ in the interval } (\tau, t_2)$$

we redefine the original t_2 to be inf τ as shown in Fig. 13. This procedure may be repeated until this is no longer possible. Thus for the new t_2, $|u_i| < 1$ on (t_2, t_1) except on a set of measure zero.

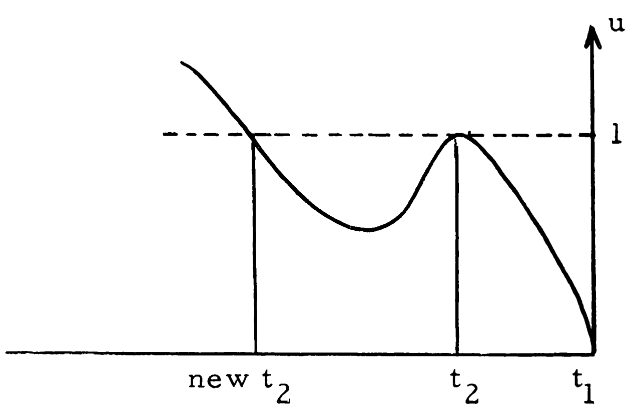


Fig. 13

Now for some i, $\left| \frac{b_i'\Psi}{2c_{ii}} \right| > 1$ on $(t_2 - \epsilon, t_2)$ for $\epsilon > 0$.

Let

$$t_3 = \inf_{[0,\ t_2]} (t_2 - \epsilon)$$

Then $t_3 = 0$ or (B) two cases arise.

(1) $\frac{b_i\Psi}{2c_{ii}} \equiv \pm 1$ for i = 1, ... r, in which case $u_i \equiv \pm 1$,

on $[0, t_3]$.

(2) If for some i, i = 1, ..., r, and some τ, $0 < \tau < t_3$,

$\left| \frac{b_i'\Psi}{2c_{ii}} \right| > 1$ in the interval (τ, t_3)

we redefine t_3 to be inf τ. This procedure is continued until it is no longer possible. Thus for the new t_3, $|u| \geq 1$ on (t_3, t_2).

Now for some $\epsilon > 0$, $\left| \frac{b_i'\Psi}{2c_{ii}} \right| < 1$ on $(t_3 - \epsilon, t_3)$. Let $t_4 =$

$$\inf_{[0,\ t_3]} (t_3 - \epsilon)$$

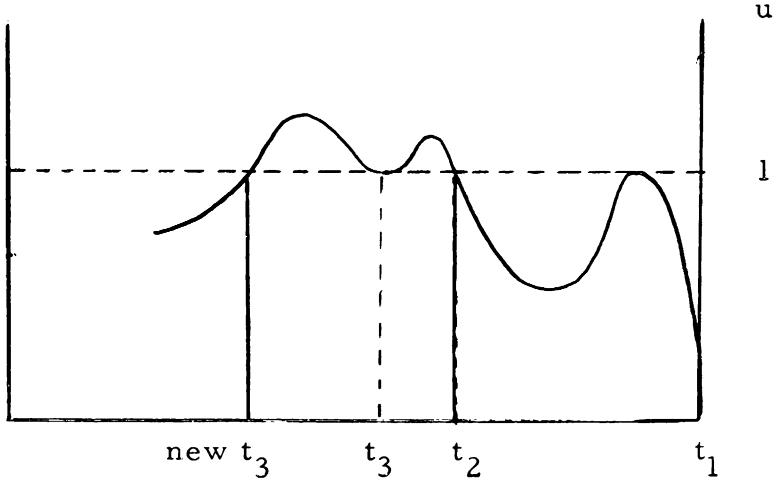


Fig. 14

Then either $t_4 = 0$ or procedure A and B may be repeated until the process terminates.

Synthesis Procedure

For simplicity, we consider a scalar control. The vector control is no different and follows directly. The composite system has either of the two forms depending on whether

$\left|\frac{b'\Psi}{2c}\right| < 1;$ or $\left|\frac{b'\Psi}{2c}\right| \geq 1.$ The corresponding composite systems are respectively

$$\frac{d}{dt}\begin{bmatrix} x \\ \Psi \end{bmatrix} = \begin{bmatrix} A & \frac{1}{2c}\, bb' \\ 2G & -A' \end{bmatrix}\begin{bmatrix} x \\ \Psi \end{bmatrix} \tag{95a}$$

$$\frac{d}{dt}\begin{bmatrix} x \\ \Psi \end{bmatrix} = \begin{bmatrix} A & 0 \\ 2G & -A' \end{bmatrix}\begin{bmatrix} x \\ \Psi \end{bmatrix} + \begin{bmatrix} b \\ 0 \end{bmatrix} \text{Sign}\,\frac{b'\Psi}{2c} \tag{95b}$$

On the interval $(t_2, t_1]$, Theorem 1 establishes the form of the composite system as (95a) and assuming, for simplicity, that the eigenvalues of the coefficient matrix D are distinct, write its constituent matrices in the partitioned form

$$Z_i = \begin{bmatrix} Z_{11}^{\ i} & Z_{12}^{\ i} \\ Z_{21}^{\ i} & Z_{22}^{\ i} \end{bmatrix} \qquad i = 1, \ldots, 2n \tag{96}$$

where Z_i, $i = 1, \ldots n$ correspond to eigenvalues λ_i, $i = 1, \ldots n$ with negative real parts and as a consequence of Lemma (A5) the matrices Z_i, $i = n+1, \ldots 2n$ correspond to the eigenvalues $\lambda_{n+i} = -\lambda_i$, $i = 1, \ldots n$ with positive real parts. From Theorem P1, it follows that $\Psi(t_1) = 0$. Letting $t_1 - t_2 = s_1$, the solution to (95a) is

$$\begin{bmatrix} x^2 \\ \\ \Psi^2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} Z_{11}^{i} e^{\lambda_i(-s_1)} + \sum_{i=n+1}^{2n} Z_{11}^{i} e^{-\lambda_i(-s_1)} \\ \\ \sum_{i=1}^{n} Z_{12}^{i} e^{\lambda_i(-s_1)} + \sum_{i=n+1}^{2n} Z_{12}^{i} e^{-\lambda_i(-s_1)} \end{bmatrix} x^1 \tag{97}$$

the superscript 2 refers to the time t_2. In order that t_2 coincide with $t = 0$, we require x^o to be within the region defined by

$$\left| \frac{1}{2c} b' \left(\sum_{i=1}^{2n} Z_{21}^{i} e^{-\lambda_i(t_1 - t_o)} \right) . \left(\sum_{i=1}^{2n} Z_{11}^{i} e^{-\lambda_i(t_1 - t_o)} \right)^{-1} x^2 \right| < 1 \tag{98}$$

If x^o does not satisfy (98) then the control over the interval $[t_3, t_2]$ is $u = \frac{1}{2} b' \Psi^2 = \pm 1$ and the solution to (95b) gives the state at t_3 as

$$x^3 = e^{A(-s_2)}(x_2 + A^{-1} b \frac{b'}{2c} \Psi_2) - A^{-1} b \frac{b'}{2c} \Psi^2 \tag{99}$$

substituting for x^2 from (97)

$$x^3 = x^3(s_2, s_1, x_1)$$

where $s_2 = t_2 - t_3$ is obtained from the solution of

$$\frac{1}{2c} b' \Psi^3 = \frac{1}{2c} b' e^{(-A')(-s_2)} \cdot$$

$$\{ \Psi^2 + \int_o^{-s_2} e^{(-A')t} 2Gx(t)\, dt \} = \pm 1$$

where

$$x(t) = e^{At}(x^2 + A^{-1}b \frac{b'}{2c} \Psi^2) - A^{-1}b \frac{b'}{2c} \Psi^2$$

then it is established that s_2 is obtained as the solution to an equation of the form

$$\frac{1}{2c} b' \Psi^3 (s_2, s_1, x_1) = \pm 1 \tag{100}$$

Continuing this procedure, the entire interval $[0, t_1]$ is covered to yield the set of equations

$$x^o = x^{m+1} = x^{m+1} (s_m, \ldots, s_1; x^1) \tag{101}$$

where the right hand side is linear in x^1, and

$$S_m = t_1 - (s_1 + \ldots + s_{m-1}) \qquad \text{i}$$

$$\frac{1}{2} b'\Psi^m = \frac{1}{2} b' \Psi^m(s_{m-1}, \ldots, s_1; x^1) = \pm 1 \qquad \text{ii}$$

$$\vdots \tag{102}$$

$$\frac{1}{2} b'\Psi^2 = \frac{1}{2} b' \Psi^2 (s_1, x^1) = \pm 1 \qquad \text{m}$$

We have thus n + m equations in n + m unknowns $s_m, \ldots, s_1; x^1$. The equation (101) is linear and has a solution by the assumption of the optimal control. The first equation in (102) is trivial and thus there are (m - 1) non-linear equations to be solved.

4.2 Infinite Transition Time

The problem is essentially the same except that $t_1 = \infty$. Specifically, for the object function

$$J = \int_0^\infty (x'G\,x + u'\,C\,u)\,dt \tag{103}$$

it is necessary that $x^1 = x(\infty) = 0$ for convergence of the integral. This problem is of great interest from the viewpoint of applications to a large class of problems and moreover it has an explicit solution.

In order that (103) converge it is also necessary that $u(\infty) = 0$. The optimal control must satisfy (91). Thus on the interval (t_2, ∞), the controls are given as $u_i = \frac{\Psi' b_i}{2c_{ii}}$, $i = 1, \ldots r$ and the composite system of equations to be solved is of the form

$$\frac{d}{dt}\begin{bmatrix} x \\ \\ \Psi \end{bmatrix} = \begin{bmatrix} A & \sum_{i=1}^{r} \frac{b_i b_i'}{2c_{ii}} \\ & \\ 2G & -A' \end{bmatrix} \begin{bmatrix} x \\ \\ \Psi \end{bmatrix} \tag{104}$$

Assuming that the above matrix has distinct eigenvalues and using the notation for constituent matrices as in (96) and the results in (A6) and (A13') the solution to (104) can be written as

$$\begin{bmatrix} x^1 \\ \\ \\ \Psi^1 \end{bmatrix} = \lim_{t \to \infty} \left\{ \sum_{i=1}^{n} Z_i e^{\lambda_i (t - t_2)} + \sum_{i=n+1}^{2n} Z_i e^{-\lambda_i (t - t_2)} \right\} \begin{bmatrix} x^2 \\ \\ \\ \Psi^2 \end{bmatrix}$$

Now since Re $\lambda_i < 0$, $i = 1, \ldots n$, the first matrix inside the brackets vanishes in the limit and the system of equations reduces to

$$\begin{bmatrix} 0 \\ \\ \Psi^1 \end{bmatrix} = \lim_{t \to \infty} \sum_{i=n+1}^{2n} \begin{bmatrix} Z_{11}^{\ i} & Z_{12}^{\ i} \\ \\ Z_{21}^{\ i} & Z_{22}^{\ i} \end{bmatrix} \begin{bmatrix} x^2 \\ \\ \Psi^2 \end{bmatrix} e^{-\lambda_i (t - t_2)}$$

Assuming that the required inverse exists, the first set of n equations are satisfied if

$$\Psi^2 = - \left(\sum_{i=n+1}^{2n} Z_{12}^{\ i} \right)^{-1} \left(\sum_{i=n+1}^{2n} Z_{11}^{\ i} \right) x^2 = P\, x^2$$

Thus the saturation of the controls occurs on the boundaries

$$|b_i{}'\, \Psi^2| = |b_i{}'\, P\, x^2| = 1 \qquad i = 1, \ldots, r$$

The controls are thus completely synthesized for $i = 1, \ldots, r$, as

$$u_i = \begin{cases} b_i{}'\, P\, x^2 & \text{for } |b_i{}'\, P\, x^2| \leq 1 \\ \\ \text{sign } (b_i{}'\, P\, x^2) & \text{for } |b_i{}'\, P\, x^2| \geq 1 \end{cases}$$

The theoretical development in Section 4.1 is believed to be of particular use in certain economic studies, chemical and other

processes where only the object function is of interest. However, as observed at the end of Section 4.1, a set of non-linear algebraic equations is obtained which must be solved. If the exact structure of the control is known, a-priori then this set of non-linear equations could be solved for the exact control. If this is not the case, then some numerical techniques must be developed to solve the problem. However, if the transition period is sufficiently long that it can be regarded as infinite then the theory of Section 4.2 can be applied and an explicit solution obtained. This problem has been investigated by Jen-wei[33] for a scalar control. In this section, this problem has been considered for a vector control and, moreover, a method for explicit solution has been given using the concepts of functions of matrices.

5. SUMMARY

This thesis has developed techniques for the application of the recent mathematical theories on optimization and extended some of these concepts. Computer programs have been written and used on diverse problems to obtain specific answers.

In Section 2, methods have been developed which can be systematically implemented on a digital computer to optimize a performance measure or object function with respect to a given set of system parameters . Specifically, in Section 2.3.1, the case of finite-time terminal state error is considered. The general theory of parameter optimization has been applied to optimize this error and an example illustrates the theory developed. In Section 2.3.2, trajectory optimization, i.e., best fit to a desired response has been considered. Explicit formulas have been developed for the cases of step and ramp inputs which make it possible to carry out an efficient computer implementation. Extension to the case when the transition time grows out of bound, including weighting, have been considered in Sections 2.3.3 and 2.4 and applied specifically to an electromechanical system. A practical procedure of stabilizing a linear system of arbitrary order is given in Section 2.5, which uses a special object function. An example illustrates the details of the theory. In practice, the system can first be stabilized using this object function and then optimized with respect to any desired object function by merely changing the object function in the computer program.

Pontryagin's maximum principle has sometimes been regarded as a mathematical sophistication. It is shown in Sections 3 and 4 that the maximum principle can be applied to engineering problems as easily as some of the conventional techniques. The important case of time optimization has been considered in Section 3.1 and a new approach to time optimality has been suggested. It is shown that by the possible use of parameter switching in conjunction with manipulation of controls, a smaller transition time can be obtained than is possible by a relay controller only. Section 3.2 treats the case when some of the parameters of the system bear an integral relation to the controls. For a special case, the optimal control, with a bound $|u(t)| \leq 1$, is shown to be of the generalized bang-bang type, i.e., it assumes only three values -1, 0 and +1.

The fixed time problem with a free final state is considered in Section 4. Explicit results for the synthesis of the optimum control are obtained for the case when the transition time grows out of bounds.

A fundamental drawback of the general procedure in Section 2 is that the result obtained is a relative minimum and not an absolute minimum. This is an area for further investigation. Although the best-fit optimization is done with respect to a step or ramp input, as in conventional control system design, it is possible to extend this to more complex signals. If the input is an algebraic expression in exponentials and polynomials of time then the homogeneous equivalent[36] of the system may possibly be used to yield a better machine adaptation.

In this thesis, a computer program was used to generate a net of time optimal trajectories running backward in time from the origin for second order systems. It is desirable for second order systems, and a necessity for higher order systems, to calculate the switching times for a given initial state of the system. The problem of calculating the parameter and control switching times remains to be solved.

APPENDIX A

FUNCTIONS OF MATRICES; DIFFERENTIABILITY OF EIGENVALUES

Theorem A.1[4, 5, 20] If f is an analytic function in an open set containing the roots $\lambda_1, \ldots, \lambda_s$ of the minimal polynomial $\mu(\lambda)$ of A with multiplicities $m_1, \ldots, m_s$, then

$$f(A) = \sum_{j=1}^{s} \sum_{k=1}^{m_s - 1} \frac{1}{k!} (A - \lambda_j U)^k f^{(k)}(\lambda_k) E_k \tag{A1}$$

where E_k is the projection matrix obtained by replacing λ by A in the polynomials

$$\mu_k(\lambda) = \frac{\mu(\lambda) a_k(\lambda)}{(\lambda - \lambda_k)^{m_k}} \tag{A2}$$

and $a_k(\lambda)$ is the polynomial in the numerator of the partial fraction expansion

$$\frac{1}{\mu(\lambda)} = \sum_{k=1}^{s} \frac{a_k(\lambda)}{(\lambda - \lambda_k)^{m_k}} \tag{A3}$$

Lemma A.1[4, 5, 20] Let $Z_{jk} = \frac{1}{k!} (A - \lambda_j U)^k E_k$ (A4)

Then

$$f(A) = \sum_{j=1}^{s} \sum_{k=1}^{m_s - 1} f^{(k)}(\lambda_j) Z_{jk} \tag{A5}$$

i.e., the function f(A) of the matrix A is determined by the n matrices Z_{jk}, called the constituent matrices, and the scalar functions $f^{(k)}(\lambda_j)$, $k = 0, \ldots (m_s - 1)$, $j = 1, \ldots, s$.

Lemma A. 2[4,5,20] If the eigenvalues of A are distinct, then

$$f(A) = \sum_{i=1}^{s} f(\lambda_i) Z_i \tag{A6}$$

and

$$Z_i = \prod_{\substack{j=1 \\ j \neq i}}^{n} \left[\frac{1}{\lambda_i - \lambda_j} (A - \lambda_j U) \right]$$

If A is a real matrix, (i.e., its entries are from the field of real numbers) then its eigenvalues are either real or appear as complex conjugates, $\lambda_1, \ldots, \lambda_{2m}, \lambda_{2m+1}, \ldots, \lambda_n$ where $\lambda_{2i-1} = \bar{\lambda}_{2i}$, $i = 1, \ldots, m$ and λ_i real for $i = 2m+1, \ldots, n$ and the constituent matrices Z_{2i-1}, Z_{2i} associated with $\lambda_{2i-1}, \lambda_{2i}$, $i = 1, \ldots, m$ are complex conjugates. Then

$$f(A) = \sum_{i=1}^{m} [f(\lambda_{2i-1}) Z_{2i-1} + f(\lambda_{2i}) Z_{2i}] + \sum_{i=2m+1}^{n} f_i(\lambda_i) Z_i$$

where

$$f(\lambda_{2i-1}) = \bar{f}(\lambda_{2i})$$

If

$$Z_{2i-1} = A_i + jB_i$$

$$\lambda_{2i-1} = \alpha_i + j\beta_i$$

$$f(\lambda_{2i-1}) = \xi_i + j\eta_i$$

then it follows that

$$f(A) = 2\sum_{i=1}^{m} [\zeta_i A_i - \eta_i B_i] + \sum_{i=2m+1}^{n} f_i(\lambda_i) Z_i \qquad (A7)$$

where all quantities in the above expression are real.

The matrices A_i, B_i, $i = 1, \ldots m$; Z_i, $i = 2m+1, \ldots n$ are hereafter referred to as the modified constituent matrices, and α_i, β_i, $i = 1, \ldots, m$; λ_i, $i = 2m+1, \ldots, n$ as the modified eigenvalues. The vector formed by the eigenvalues will be denoted by λ and that by modified eigenvalues by λ^*.

In the following Lemma, formulas for calculating the modified constituent matrices are given without proof. They follow from direct calculations.

Lemma A.3 The modified constituent matrices are given by the following formulas:

$$L_i = \prod_{\substack{1 \leq k \leq m \\ k \neq i}} \frac{[(A - \alpha_k U)^2 + \beta_k^2 U]}{[(\alpha_i - \alpha_k)^2 + (\beta_i + \beta_k)^2][(\alpha_i - \alpha_k)^2 + (\beta_i - \beta_k)^2]}$$

$$\cdot \prod_{\substack{2m+1 \leq k \leq n \\ k \neq i}} \frac{(A - \lambda_k U)}{[(\alpha_i - \alpha_k)^2 + \beta_i^2]}$$

$$A_i = L_i \frac{1}{2\beta_i} [d(A - \alpha_i U) + c\,\beta_i U] \qquad (A8)$$

$$B_i = L_i \frac{-1}{2\beta_i} [c(A - \alpha_i U) - d\,\beta_i U] \qquad (A9)$$

$$i = 1, \ldots, m$$

where

$$c + j\,d = \prod_{\substack{1 \le k \le m \\ k \ne i}} [(\alpha_i - \alpha_k) - j(\beta_i + \beta_k)][(\alpha_i - \alpha_k) - j(\beta_i - \beta_k)]$$

$$\cdot \prod_{\substack{2m+1 \le k \le n \\ k \ne i}} (\alpha_i - \lambda_k - j\beta_i) \quad i = 1, 2, \ldots, m$$

$$Z_i = \prod_{\substack{1 \le k \le m \\ k \ne i}} \frac{(A - \alpha_k U)^2 + \beta_k{}^2 U}{(\lambda_i - \alpha_k)^2 + \beta_k{}^2} \prod_{\substack{2m+1 \le k \le n \\ k \ne i}} \frac{(A - \lambda_k U)}{(\lambda_i - \lambda_k)} \qquad \text{(A10)}$$

$$i = 2m + 1, \ldots, n$$

Similarly formulas in real domain are obtainable for multiple eigenvalue case but with additional algebraic complexity.

Theorem A.2 [13] Consider the system of differential equations

$$\dot{x} = f(w, u(t), t)$$

where u(t) is a bounded measurable function on $[t_o, t_1]$, which can take on values in a compact set $\overline{U}$. Let f be continuous in x and u and is continuously differentiable with respect to x. Then there exists a unique absolutely continuous function x(t) which satisfies the above equation.

Corollary A.2.1 If the system in Theorem A.2 is linear, then the solution to the system is defined on the interval $[t_o, t_1]$.

Corollary A.2.2 In the special case of a linear system as in (1) the solution is explicitly given as

$$x(t) = e^{A(t - t_o)}[\mathbf{x}_o + \int_{t_o}^{t} e^{-A(\tau - t_o)} B\, u(\tau)\, d\tau] \qquad \text{(A11)}$$

Furthermore, if $u(t) = u$ on $[t_o, t_1]$, then

$$x(t) = e^{A(t - t_o)}(x_o + A^{-1}Bu) - A^{-1}Bu \qquad \text{(A12)}$$

and for $u(t) = 0$, i.e., when the system is homogeneous

$$x(t) = e^{A(t - t_o)} x^o$$

Corollary A.2.3 If the input to the linear system in (1) is of the form

$$u = \sum_k p_k(t) e^{r_k t} = \sum_k (c_o{}^k + \ldots + c_\ell{}^k t^\ell) e^{r_k t}$$

then the solution is given in a closed form as

$$x(t)\; e^{A(t - t_o)}[x_o + \sum_k \sum_{i=1}^{n} Z_i \left\{ \sum_{j=0}^{\ell} c_j \frac{d^{(j)}}{d\alpha_{ki}{}^{(j)}} \left(\frac{e^{\alpha_{ki}(t - t_o)}}{\alpha_{ki}} \right) \right\} \Bigg|_{t_o}^{t}] \qquad \text{(A13)}$$

where $\alpha_{ki} = (r_k - \lambda_i)$.

Proof: Expressing $e^{-A(t - t_o)}$ in terms of its constituent matrices in equation (A11)

$$x(t) = e^{A(t - t_o)}[x_o + \int_{t_o}^{t} \sum_{t=1}^{n} Z_i e^{-\lambda_i(t - t_o)} \sum_k \sum_{j=0}^{\ell} c_j^k \tau^j e^{r_k(t-t_o)} d\tau]$$

$$= e^{A(t - t_o)}[x_o + \sum_k \sum_{i=1}^{n} Z_i \int_{t_o}^{t} c_j^k \tau^j e^{(r_k - \lambda_i)(\tau - t_o)} d\tau]$$

from which equation (A13) follows immediately.

Let

$$\overline{Wa'} = \overline{Wa} - \cup \{S(a, \epsilon): \lambda_i = \lambda_j \text{ for } i \neq j \text{ for } A(a)\} \qquad \text{(A14)}$$

and let the interior of $\overline{Wa'}$ be denoted by $\overline{Wa}^{o'}$

Theorem A.3.1 If the eigenvalue λ_i of A(a) is real for $a \in \overset{o}{\overline{W}}{}'$ (or a subset of $\overset{o}{\overline{W}}{}'$) then the partial derivative of λ_i with respect to a exists throughout $\overset{o}{\overline{W}}{}'$ (or that subset of $\overset{o}{\overline{W}}{}'$). Moreover λ_i is continuously differentiable with respect to $\underline{a}$.

Proof: The coefficients in the characteristic polynomial

$$\delta(\lambda_i, a) = \lambda_i^n + d_i(a)\lambda_i^{n-1} + \ldots + d_n(a) \qquad \text{(A15)}$$

are linear (or higher) forms in $\underline{a}$. This follows directly because we required the entries of A(a) to be linear in $\underline{a}$. Then $\partial\delta(\lambda_i, a)/\partial a$ are defined and differentiable with respect to $\underline{a}$ any desired number of times. The same is true of $\partial\delta(\lambda_i, a)/\partial\lambda_i$ if it is defined for a particular $\underline{a}$.

Implicit differentiation of (A15) gives

$$\frac{\partial \lambda_i}{\partial a_j} = - \frac{(\partial \delta(\lambda_i, a)/\partial a_j)}{(\partial \delta(\lambda_i, a)/\partial \lambda_i)} \tag{A16}$$

where the denominator does not vanish for otherwise λ_i is a double root of $\delta(\lambda_i, a)$, which is not possible in $\overset{o}{\overline{Wa}}{}'$. Thus the right hand side exists and is continuously differentiable with respect to λ_i and $\underline{a}_j$. This being true for every j, the theorem is proved.

<u>Theorem A.3.2</u> In the parameter set $\overset{o}{\overline{W}}{}'$ the real and imaginary parts, α_i and β_i of a pair of complex conjugate eigenvalues of $A(a)$, $a \in \overset{o}{\overline{W}}{}'$, have continuous partial derivatives with respect to $\underline{a}$.

<u>Proof:</u> Using $\lambda_{2i-1} = \alpha_i + j\beta_i$ in $\delta(\lambda_{2i-1}, a)$ we have

$$\delta(\lambda_{2i-1}, a) = p_1(\alpha_i, \beta_i, a) + j\, p_2(\alpha_i, \beta_i, a)$$

We now require to solve the simultaneous system

$$\begin{aligned} p_1(\alpha_i, \beta_i, a) &= 0 \\ p_2(\alpha_i, \beta_i, a) &= 0 \end{aligned} \tag{A17}$$

Taking the partials with respect to a_j

$$\begin{bmatrix} \frac{\partial p_1}{\partial \alpha_i} & \frac{\partial p_1}{\partial \beta_i} \\ \frac{\partial p_2}{\partial \alpha_i} & \frac{\partial p_2}{\partial \beta_i} \end{bmatrix} \begin{bmatrix} \frac{\partial \alpha_i}{\partial a_j} \\ \frac{\partial \beta_i}{\partial a_j} \end{bmatrix} + \begin{bmatrix} \frac{\partial p_1}{\partial a_j} \\ \frac{\partial p_2}{\partial a_j} \end{bmatrix} = 0 \tag{A18}$$

Since p_1 and p_2 are linear (or higher) forms in $\underline{a}$, the entries of the second vector exist throughout $\overset{o}{\overline{W}}\,'$. The system can be solved for the first vector because the Jacobian matrix F_{ij} is non-singular. Indeed the determinant of the Jacobian is

$$|F_{ij}| = \frac{\partial p_1}{\partial \alpha_i} \frac{\partial p_2}{\partial \beta_i} - \frac{\partial p_1}{\partial \beta_i} \frac{\partial p_2}{\partial \alpha_i}$$

$$= \left(\frac{\partial p_1}{\partial \alpha_i}\right)^2 + \left(\frac{\partial p_2}{\partial \alpha_i}\right)^2$$

The last step is a consequence of the Cauchy Riemann equations for analytic functions. Now suppose $|F_{ij}| = 0$ then

$$\frac{\partial p_1}{\partial \alpha_i} = 0, \quad \frac{\partial p_2}{\partial \alpha_i} = 0$$

But this implies $\delta\,'(\lambda_{2i-1}) = \frac{\partial p_1}{\partial \alpha_i} + \frac{\partial p_2}{\partial \alpha_i} = 0$

which requires λ_{2i-1} to be an eigenvalue of multiplicity more than one.

Corollary The entries of the Jacobian matrix $\partial \lambda^* / \partial a$ exist and the matrix is

$$\left[\frac{\partial\lambda^*}{\partial a}\right] = - \left[\begin{array}{ccc} F_{11}^{-1} p_{11} & \cdots & F_{1q_a}^{-1} p_{1q_a} \\ \vdots & & \\ \hdashline \vdots & & \vdots \\ \vdots & & \vdots \\ \hline \frac{\partial\lambda_{2m+1}}{\partial a_1} & & \frac{\partial\lambda_{2m+1}}{\partial a_{q_a}} \\ \hdashline \vdots & & \vdots \\ \frac{\partial\lambda_n}{\partial a_1} & \cdots & \frac{\partial\lambda_n}{\partial a_{q_a}} \end{array}\right] \begin{array}{l} \} \; 2 \\ \} \; 2m \\ \} \; n \end{array} \qquad (A19)$$

where F_{ij} is the Jacobian matrix as in (A18) and p_{ij} is the vector

$$p_{ij} = \begin{bmatrix} \frac{\partial p_1}{\partial a_j} \\ \\ \frac{\partial p_2}{\partial a_j} \end{bmatrix}$$ in the same equation.

<u>Theorem A.4</u> If $J = \int_{t_o}^{\infty} f_o(x, u)dt$ exists and f_o is in classes $C(\underline{\overline{X}} \times \overline{U})$ and $D(\underline{\overline{X}})$ then for a fixed u and x, J is a function of the arguments $\lambda^* = \lambda^*(a)$ and $w = (a, b)$, $a \in \overset{o}{\overline{W}}{}'$ and $b \in \overline{W}_b$.

Proof: Invoking the property of the continuous dependence and differentiability of x on the initial parameters w and $\lambda^* = \lambda^*(a)$, we may write $x(t) = x(\lambda^*(a), a, b, t)$. Once again applying this property to (5), we have

$$x_o(t) = x_o(\lambda^*(a), a, b, t)$$

For fixed t, in particular for $t = t_1$ (also for $t_1 = \infty$, since by hypothesis $x_o(\infty) = J$ exists) it follows that

$$x_o(t_1) = J = J(\lambda^*(a), a, b) \tag{A20}$$

Lemma A.5 If λ_i is an eigenvalue of the matrix

$$D = \begin{bmatrix} A & Q \\ G & -A' \end{bmatrix} \quad \text{where} \quad \begin{matrix} Q = Q' \\ G = G' \end{matrix}$$

then $-\lambda_i$ is also an eigenvalue of D.

Proof: The characteristic matrix of D is

$$\begin{bmatrix} \lambda U - A & -Q \\ -G & \lambda U + A' \end{bmatrix} \tag{A21}$$

Its negative transpose is

$$\begin{bmatrix} -\lambda U + A' & G \\ Q & -\lambda U - A \end{bmatrix}$$

Now pre and post multiplying by the non-singular matrices as shown

$$\begin{bmatrix} 0 & -U \\ U & 0 \end{bmatrix} \begin{bmatrix} -\lambda U + A' & G \\ Q & -\lambda U - A \end{bmatrix} \begin{bmatrix} 0 & U \\ -U & 0 \end{bmatrix} = \begin{bmatrix} -\lambda U - A & -Q \\ -G & -\lambda U + A' \end{bmatrix} \qquad \text{(A22)}$$

Except for the sign (A22) must have the same determinant as (A21). But (A22) is obtained simply by replacing λ by $-\lambda$ in (A21). Hence the proof.

APPENDIX B

VARIATIONAL TECHNIQUES AND MAXIMUM PRINCIPLE

The maximum principle originated by Pontryagin and associates is developed in the following as an extension to the calculus of variations.

Let the system be described by

$$\dot{x} = f(x, u) \tag{B1}$$

where x is an n-vector in the n-state-space $\underline{\overline{X}}$ and u is an r-vector. It is assumed that the n-vector function f satisfies conditions for the existence of a unique solution to (B1). The function u(t), $t_o \leq t \leq t_1$, is from a class of admissible functions. In what follows, the class of piece-wise continuous functions or any of its subclasses are considered admissible. In general u(t) is permitted to take values in an arbitrary subset $\overline{U}$ of the r-space.

The initial and final states, $x(t_o) = x^o$ and $x(t_1) = x^1$, are prescribed as two points in the state-space $\overline{X}$. A solution of (B1) passing through x^o at $t = t_o$ and x^1 at $t = t_1$ is called a trajectory and the associated function u(t) is called the control. The performance measure is

$$J = \int_{t_o}^{t_1} f_o(x(t),\ u(t))dt \tag{B2}$$

where f_o satisfies conditions for the existence and uniqueness of the solution to the equation

$$\dot{x}_o = f_o(x(t),\ u(t)) \qquad (B3)$$

It is desired to find the control u(t), from among all admissible controls, and the corresponding trajectory x(t) such that the functional J in (B2) takes on the least possible value. When such is the case u(t) and x(t) are called optimal. The following theorem states a necessary condition for u(t) and x(t) to be optimal.

Theorem B1: For the control u(t) and the trajectory x(t) to be optimal, it is necessary that there exist a non-zero continuous function $\underline{\Psi}(t) = (-1,\ \Psi'(t))$ corresponding to u(t) and x(t) such that

(a) for every t, $t_o \leq t \leq t_1$, the Hamiltonian

$$H(\underline{\Psi}(t),\ x(t), u) = -f_o(x(t),\ u) + \sum_{i=1}^{n} \Psi_i(t)\ f_i(x(t),\ u) \qquad (B4)$$

attain its maximum at the point u = u(t):

$$H(\underline{\Psi}(t),\ x(t),\ u) = M(\underline{\Psi}(t),\ x(t)) \equiv \sup_{\overset{*}{u} \in \overline{U}} H(\underline{\Psi}(t),\ x(t),\ \overset{*}{u}) \qquad (B5)$$

(b) and if $\Psi(t)$, x(t), and u(t) satisfy systems (B12), (B13) and condition (a) then

$$M(\underline{\Psi}(t),\ x(t)) = 0 \qquad t_o \leq t \leq t_1$$

Proof: The set (B1) may be rewritten as

$$\dot{x} - f(x,\ u) = 0 \qquad (B6)$$

Using a multiplier vector function $\Psi(t)$, consider minimizing (without constraints) the functional

$$J^*[x, \dot{x}, u, t_1] = \int_{t_o}^{t_1} f^*(x, \dot{x}, u)dt \tag{B7}$$

where the augmented scalar function f^* is defined as

$$f^*(x, \dot{x}, u) = f_o(x, u) + \Psi'[\dot{x} - f(x, u)] \tag{B8}$$

The increment of J^* is

$$\Delta J^* = J^*[x + \epsilon\, \delta x, \dot{x} + \epsilon\delta \dot{x}, u^*, t_1 + \epsilon\delta t] - J[x, \dot{x}, u, t_1]$$

$$= \int_{t_o}^{t_1} \{f_o(x + \epsilon\delta x, u^*) + \Psi'[\dot{x} + \epsilon\delta \dot{x} - f(x + \epsilon\delta x, u^*]$$

$$-f_o(x, u) \qquad - \Psi'[\dot{x} - f(x, u)]\}\, dt$$

$$+\int_{t_1}^{t_1 + \epsilon\delta t_1} f^*(x + \epsilon\delta x, u^*)dt \qquad \text{B9)}$$

where $u^*(t) \epsilon \bar{U}$ is the perturbed control and $\epsilon \geq 0$ is a real parameter. Adding and subtracting terms gives

$$\Delta J^* = \int_{t_o}^{t_1} \{f_o(x + \epsilon\, \delta x, u) + \Psi'[\dot{x} + \epsilon\delta \dot{x} - f(x + \epsilon\delta x, u)]$$

$$-f_o(x, u) \qquad -\Psi'[\dot{x} - f(x,u)]\}dt$$

$$+\int_{t_o}^{t_1} \{f_o(x + \epsilon\delta x, u^*) - \Psi' f(x + \epsilon\, \delta x, u^*)$$

$$-f_o(x + \epsilon\, \delta x, u) + \Psi' f(x + \epsilon\delta x, u)\} \, dt$$

$$+\int_{t_1}^{t_1+\epsilon\delta t_1} f^*(x + \epsilon\, \delta x, u^*) dt \tag{B10}$$

Expanding the first integral $\Delta J_1{}^*$ in a Taylor series gives

$$\Delta J_1{}^* = \int_{t_o}^{t_1} [\frac{\partial}{\partial x} (f_o - \Psi' f)\, x + \Psi' \delta \dot{x}] dt + 0\,(\epsilon^2)$$

$$= \int_{t_o}^{t_1} [\frac{\partial}{\partial x} (f_o - \Psi' f) - \dot{\Psi}'] \delta x \, dt + \epsilon \Psi' \delta x(t_1) + 0\,(\epsilon^2)$$

It is clear that if $u^* = u$ and $\delta t_1 = 0$, then $\delta x(t_1) = 0$ and the increment reduces to

$$\Delta J_1{}^* = \epsilon \int_{t_o}^{t_1} [\frac{\partial}{\partial x} (f_o - \Psi' f) - \dot{\Psi}'] \, \delta x \, dt + 0(\epsilon^2)$$

where the first term is the variational which must vanish. Since δx is continuous, application of the fundamental Lemma of calculus of variations gives

$$\dot{\Psi} = \left(\frac{\partial f_o}{\partial x}\right)' - \left[\frac{\partial f}{\partial x}\right]' \Psi \qquad \text{(B11)}$$

The equations (B11) and (B6) can be written as a Hamiltonian system

$$\dot{x}_i = \frac{\partial H}{\partial \Psi_i} \qquad i = 1, \ldots n \qquad \text{(B12)}$$

$$\dot{\Psi}_i = - \frac{\partial H}{\partial x_i} \qquad i = 1, \ldots n \qquad \text{(B13)}$$

which must be satisfied for the variation of J to vanish. When such is the case, (B10) becomes

$$\Delta J^* = \epsilon \Psi' \delta x(t_1) + \int_{t_1}^{t_1 + \epsilon \delta t_1} f^*(x + \epsilon \delta , u^*) dt$$

$$+ \int_{t_o}^{t_1} \{ f_o(x + \epsilon \delta x, u^*) - \Psi' f(x + \epsilon \delta x, u^*)$$

$$- f_o(x + \epsilon \delta x, u) + \Psi' f(x + \epsilon \delta x, u) \} dt$$

$$+ 0(\epsilon^2) \qquad \text{(B14)}$$

But

$$\delta x(t_1) \cong \delta x^1 - \dot{x} \delta t_1 \qquad \text{(B15)}$$

in which $\delta x^1 = 0$ since x^1 is fixed. Substituting for $\delta x(t_1)$ and f^* from (B15) and (B8) in (B14) and allowing cancellation of terms there results

$$\Delta J^* = \epsilon[f_o - \Psi' f]\delta t_1$$
$$+ \int_{t_o}^{t_1} [f_o(x + \epsilon\delta x, u^*) - \Psi' f(x + \epsilon\delta x, u^*)$$
$$-f_o(x + \epsilon\delta x, u) + \Psi' f(x + \epsilon\delta x, u)]dt$$
$$+ 0(\epsilon^2) \qquad \text{(B16)}$$

Perturbed Control u^* :

Select instants of time τ_1 and τ such that

$$t_o < \tau_1 < \tau < t_1$$

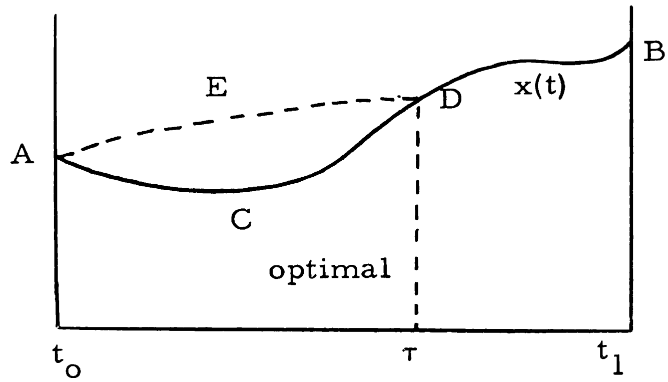
and are points of continuity of u(t). Then for an arbitrary number δt and an arbitrary positive-number δt_1, the following half-open interval is defined

$$I_1 : \quad \tau_1 - \epsilon\delta t_1 < t \leq \tau_1$$

where ϵ is chosen to be the same as earlier and such that $t_o < \tau_1 - \epsilon\delta t_1$. Now let the perturbed control be taken as

$$u^* = \begin{cases} u(t) & \text{if} \quad t \notin I_1 \\ v_1 & \text{if} \quad t \in I_1 \quad \underline{\text{for any}} \quad v_1 \in \bar{U} \end{cases}$$

We now make use of the fact that if x(t) and u(t) are optimal then any sections of x(t) and u(t) are also optimal, i.e., for any τ $t_o \le \tau \le t_1$,



both x(t), $t_o \le t \le \tau$ and u(t), $t_o \le t \le \tau$ are optimal if x(t) and u(t), are optimal. This is a direct consequence of the definition of optimal control.

Consider the increment on J*up to time $t = \tau$ as given by (B16)

$$\Delta J^{\tau} \equiv J^*[x + \epsilon\delta x,\ \dot{x} + \epsilon\delta \dot{x},\ u^*,\ \tau + \delta t] - J[x,\ \dot{x},\ u, \tau]$$

$$= [f_o(\tau) - \Psi'(\tau)f(\tau)]\delta t$$

$$+\int_{t_o}^{\tau} [f_o(x+ \epsilon\delta x,\ u^*) - \Psi' f(x + \epsilon\delta x,\ u^*)$$

$$-f_o(x + \epsilon\delta x,\ u) - \Psi' f(x + \epsilon\delta x,\ u)]dt$$

$$+ 0(\epsilon^2) \qquad \text{(B 17)}$$

If we set $\delta t_1 = 0$ then for ΔJ^{τ} to be positive for sufficiently small ϵ it is required that

$$-\epsilon[-f_o(\tau) + \Psi'(\tau)\, f(\tau)]\delta t \leq 0$$

But since δt is arbitrary, i.e., it can be positive or negative, the following equation, which will be used later, follows.

$$H(\underline{\Psi}(\tau), f(\tau)) \equiv -f_o(\tau) + \underline{\Psi}'(\tau) f(\tau) = 0 \tag{B18}$$

Lemma B1 For any point, t, of the continuity of u(t), $t_o \leq t \leq \tau$,

$$H(\underline{\Psi}(t), f(x(t), u)) = \sup_{u^* \in \bar{U}} H(\underline{\Psi}(t), f(x(t), u^*)) \tag{B19}$$

Proof: Set $\tau_1 = t, \delta t_1 = 1$ and $t = 0$, then

$$\Delta J^{\tau} = \int_{t_o}^{\tau} [f_o(x + \epsilon\delta x, u^*) - f_o(x + \epsilon\delta x, u)$$

$$- \Psi' f(x + \epsilon\delta x, u^*) + \Psi' f(x + \epsilon\delta x, u)]d\sigma$$

$$+ 0(\epsilon^2) \tag{B 20}$$

$$= \epsilon[f_o(x(t), v_1) - f_o(x(t), u(t))]$$

$$- \epsilon\Psi'[f(x(t), v_1) - f(x(t), u(t))] + 0(\epsilon^2) \tag{B 21}$$

$$= - H(\Psi(t), x(t), v_1) + H(\Psi(t), x(t), u(t)) \tag{B 22}$$

Equation (B 21) followed from (B 20) by application of mean value theorem on integrals which is rewritten in (B22) merely by using the definition of H. Now since u is optimal, $\Delta J \geq 0$. This requires

that

$$H(\underline{\Psi}(t), x(t), v_1) \leq H(\underline{\Psi}(t), x(t), u(t))$$

However, since this inequality is valid for any $v_1 \in \overline{U}$ so

$$H(\underline{\Psi}(t), f(x(t), u(t))) = \sup_{u^* \in \overline{U}} H(\underline{\Psi}(t), f(x(t), u^*)) \quad \text{(B 23)}$$

and Lemma B1 is proved.

It is noted that Lemma B1 is also applicable when $t = \tau$ (since τ is a point of continuity) so that

$$H(\underline{\Psi}(\tau), f(x(\tau), u(\tau))) = \sup_{u^* \in \overline{U}} H(\underline{\Psi}(\tau), f(x(\tau), u^*) \quad \text{(B 24)}$$

(By defining u at the points of discontinuities to be the limit from the left hand side, relation (B 24) is valid on the entire interval). From (B18) and (B 24) the following Lemma follows.

<u>Lemma B2</u> At the instant τ (which was any point of continuity of $u(\tau)$),

$$M(\Psi(\tau), x(\tau)) \equiv \sup_{u^* \in \overline{U}} H(\Psi(\tau), f(x(t), u^*)) = 0 \quad \text{(B 25)}$$

<u>Lemma B3</u> If the relation (B 24), i.e.,

$$H(\underline{\Psi}(t), f(x(t), u(t)) = M(\underline{\Psi}(t), x(t))$$

is satisfied at every point of continuity of u on $I = \{t: t_o \leq t \leq t_1\}$ then the function $M(\underline{\Psi}(t), x(t))$ is a constant on I.

Proof:* Since $u(t) \in \bar{U}$ and $\bar{U}$ is bounded, the set of points $u(t)$ has a compact clusure $\bar{V}$ in the r-space. It is clear then that the function

$$N(\underline{\Psi}, x) = \max_{u \in \bar{V}} H(\underline{\Psi}, x, u) = M(\underline{\Psi}, x) \tag{B26}$$

Thus, we will prove the lemma just for N. Since I is compact there exists a convex and bounded set $\bar{W}$ in the 2n+1 space of $(\underline{\Psi}, x)$ such that $(\underline{\Psi}(t), x(t))$ belongs to $\bar{W}$ for $t \in I$, and $(\Psi(t), x(t), u(t)) \in \bar{W} \times \bar{V}$. However, since $H(\underline{\Psi}, x, u)$ has continuous partial derivatives with respect to all arguments, it follows that for any (Ψ, x) and (Ψ^*, x^*) in $\bar{W}$ and $u \in \bar{V}$ there exists a number K such that

$$H(\underline{\Psi}, x, u) - H(\underline{\Psi}^*, x^*, u) \leq K \|a - a^*\| \tag{B27}$$

$$\text{where } a = (\underline{\Psi}, x)$$
$$a^* = (\underline{\Psi}^*, x^*) \tag{B28}$$

Let u and u* be two points in $\bar{V}$

$$N(\underline{\Psi}, x) = H(\underline{\Psi}, x, u) >: \quad N(\underline{\Psi}^*, x^*) = H(\underline{\Psi}^*, x^*, u^*) \tag{B29}$$

Clearly then

$$\begin{aligned} -K\|a - a^*\| &\leq H(\underline{\Psi}, x, u^*) - H(\underline{\Psi}^*, x^*, u^*) \\ &\leq H(\underline{\Psi}, x, u) - H(\underline{\Psi}^*, x^*, u^*) \end{aligned} \tag{B30}$$

and

* The proof of this lemma closely follows the method of Pontryagin.[2]

$$H(\underline{\Psi}, x, u) - H(\underline{\Psi}^*, x^*, u^*)$$

$$\leq \quad H(\underline{\Psi}, x, u) - H(\underline{\Psi}^*, x^*, u) \leq K \; \|a - a^*\| \tag{B31}$$

These inequalities result from (B2) and(B29) which imply that $-Kd \leq H(\underline{\Psi}, x, u) - H(\underline{\Psi}^*, x^*, u) \leq Kd$ and

$$H(\underline{\Psi}, x, u^*) \leq H(\underline{\Psi}, x, u), \quad H(\underline{\Psi}^*, x^*, u) \leq H(\underline{\Psi}^*, x^*, u^*)$$

respectively.

Now combining (B30) and (B31) one obtains

$$H(\underline{\Psi}, x, u) - H(\underline{\Psi}^*, x^*, u^*)$$

$$= \quad N(\underline{\Psi}, x) - N(\underline{\Psi}^*, x^*) \leq K \; \|a - a^*\|$$

Since a is piecewise smooth, this inequality implies that the function $N(\underline{\Psi}, x)$ is piecewise smooth, i.e. it has a continuous derivative except at a finite number of points. Finally, we show that $N(\underline{\Psi}(t), x(t))$ has a zero derivative on the entire interval $t_o \leq t \leq t_1$ except at a finite number of points.

Let $t = \tau$ be any point at which $\underline{\Psi}(t)$, $x(t)$ have a continuous derivative and $N(\underline{\Psi}(t), x(t))$ has a derivative and let t^* be any arbitrary point of the interval $t_o \leq t \leq t_1$. Then

$$N(\underline{\Psi}(t^*), x(t^*)) \geq H(\underline{\Psi}(t^*), x(t^*), u(\tau))$$

(Here u is optimal). Therefore,

$$N(\underline{\Psi}(t^*), x(t^*)) - N(\underline{\Psi}(\tau), x(\tau)) \geq$$

$$H(\underline{\Psi}(t^*), x(t^*), u(\tau)) - H(\underline{\Psi}(\tau), x(\tau), u(\tau))$$

Suppose $t^* > \tau$. Dividing the above inequality by $t^* - \tau$ and letting it approach zero, there results

$$\frac{d}{dt} N(\underline{\Psi}(t), x(t)) \Bigg|_{t=\tau} \geq \frac{d}{dt} H(\underline{\Psi}(t), x(t), u(\tau)) \Bigg|_{t=\tau}$$

$$= \sum_{i=1}^{n} \frac{\partial H}{\partial \Psi_i} \frac{d\Psi_i}{dt} + \sum_{i=1}^{n} \frac{\partial H}{\partial x_i} \frac{dx_i}{dt}$$

$$= 0$$

Similarly, if $t^* < \tau$, we obtain

$$\frac{d}{dt} N(\Psi(t), u(t)) \Bigg|_{t=\tau} \leq 0$$

Thus $M(\underline{\Psi}(t), x(t)) \equiv 0$ on $t_o \leq t \leq t_1$.

This completes the proof of Theorem B1.

REFERENCES

1. Pontryagin, L. S., Boltyanski, V. G., Gamkrelidze, R. V., and Mischchenko, E.F., "The Maximum Principle in the Theory of Optimal Processes of Control," Proceedings of the First I.C.I.F.A.C., Butterworths, London, 1961.

2. ________, The Mathematical Theory of Optimal Processes, Interscience Publishers, 1962.

3. Koenig, H. E., Tokad, Y., and Kesavan, P., Analysis of Discrete State Physical Systems, McGraw-Hill Book Company. To be published.

4. Frame, J. S., Koenig, H.E., "Matrix Functions and Applications," IEEE SPECTRUM, March, April, May and June, 1964.

5. Zadeh, L. A., and Desoer, C. A., Linear System Theory. McGraw-Hill Book Company, 1963.

6. Schultz, W. C., and Rideout, V.C., "Control System Performance Measures: Past, Present, and Future," IRE Trans. on A-C, February 1961, pp. 22-35.

7. ________, "The Selection and Use of Servo-Performance Criterion," AIEE Trans., Vol. 6, Bt. II, 1957, pp. 383-388.

8. ________, "A General Criterion for Servo Performance," Proc. N.E.C., Vol. 13, pp. 549-560, 1957.

9. Clark, P.N., "Integral of the Error Squared as a Performance Index for Automatic Control Systems," AIEE Trans., Vol. 79, pp. 467-471, 1961.

10. Rekasius, Z. V., "A General Performance Index for Analytical Design of Control Systems," IRE Trans. on A-C., May 1961, pp. 217-222.

11. Bellman, R. and Kalaba, R., "Dynamic Programming and Feedback Control," Proc. I.C.I.F.A.C., Butterworths, London, 1961.

12. Gelfand, I. M., and Fomin, S. V., Calculus of Variations, Prentice Hall, Inc., 1963.

13. Pontryagin, L.S., Ordinary Differential Equations (English Translation)., Addison-Wesley Publishing Company, Inc., 1962.

14. Spang, H.A., "A Review of Minimization Techniques for Non-Linear Functions," SIAM Review, 1962, pp. 343-365.

15. Courant, R., "Variational Methods for the Solution of Problems of Equilibrium and Vibration," Bull. Am. Math. Soc., Vol. 49, pp. 1-23, 1943.

16. Rosen, J.B., "The Gradient Projection Method for Non-Linear Programming, Pt. I. Linear Constraints," SIAM Journal, 8, 181-217 (1960).

17. ______, "Pt. II, Non-Linear Constraints," SIAM Journal, 9, 514-532 (1961).

18. Leitman, , Optimization Techniques, Academic Press,

19. Hahn, W., Theory and Application of Liapunov's Direct Method, (English Translation), Prentice Hall, Inc. 1963.

20. Gantmacher, F.R., The Theory of Matrices (English Translation), Chelsea Publishing Company, Vol. I and II, 1959.

21. Desoer, C.A., "The Bang-Bang Servo Problem Treated by Variational Techniques," Information and Control, Vol. 2(1959), pp. 333-348.

22. Lee, E.B., "Mathematical Aspects of the Synthesis of Linear Minimum Response-Time Controllers," IRE Trans. A-C, Sept. 1960, pp. 283-289.

23. Bellman, R., "On the Bang-Bang Control Problem," Q. of Applied Math., April, 1956, p. 11.

24. Hemesath, N.B., "On the Determination of Time Optimal Controls for Linear Stationary Systems," Unpublished Thesis, Michigan State University, East Lansing, Michigan, 1964.

25. Smith, F.B., Jr., "Time Optimal Control of Higher Order Systems," IRE Trans. A-C, February 1961, pp. 16-21.

26. Hopkin, A.M., "A Phase Plane Approach to the Compensation of Saturating Servomechanisms," AIEE Trans., Pt. I, 1951, pp. 631-639.

27. Bogner, I., Kazda, L.F., "An Investigation of the Switching Criterion for Higher Order Contactor Servomechanism," AIEE Trans., Vol. 3, Pt. II, pp. 188-127, 1954.

28. Rozonoer, L.I., "L.S. Pontryagin Maximum Principle in the Theory of Optimum Systems," A. and R.C., October, 1959, pp.1288-1302, Pt. I; November 1959 pp.1405-1421, Pt.II; December 1959, pp.1517-1533, Pt. III.

29. Rozenman, E.A., "Optimal Control of an Object with Two Controlling Stimuli," A. and R.C., October 1959, pp.1314-1318.

30. Bor-Ramenskii, and Snug Chien, "Optimum Servo Drive with Two Control Parameters," A. and R.C., September 1961, pp. 134-145.

31. Letov, A.M., "The Analytical Design of Control Systems," A. and R.C., October 1961, pp.363-372.

32. Royden, H.L., Real Analysis, The Macmillian Company, 1963.

33. Chang, Jen-wei, "A Problem in the Synthesis of Optimal Systems using Maximum Principle," A. and R.C., October 1961, pp.1170-1176.

34. Fickeise, F.C., "Analogue Methods for Servomechanism Design," AIEE Trans., Vol. 71, Pt.II, pp.244-250, 1952.

35. Lucien, W. Neustadt, "Time Optimal Control Systems with Position and Integral Limits," J. Math. Analysis and Applications, Vol. 3, pp. 406-427, December 1961.

36. Koeing, H.E., and Tokad, Y., "A Homogeneous Equivalent of Nonhomogeneous Linear State Models and its Application to the Analysis of Continuous and Discrete-State Systems," Allerton Conference on Circuit and System Theory, University of Illinois, November 1963.

COMPUTER PROGRAMS FOR OPTIMIZATION OF LINEAR SYSTEMS

By

Vijay K. Jain

(A supplement to a thesis entitled, "Optimization of Linear Systems," by the author)

Michigan State University
Department of Electrical Engineering
1964

DO NOT DETACH

ACKNOWLEDGEMENTS

The author wishes to thank Dr. H. E. Koenig and Dr. H. G. Hedges for their guidance and criticism.

He also wishes to thank the staff of the Computer Center for their cooperation.

INTRODUCTION

This supplement presents the computer programs, for optimization of linear systems, used in the examples of the thesis. These programs have been made sufficiently general and can be used for many optimization problems with slight or no modification. A brief discussion of the theory and the input procedure precedes the programs which have been written in FORTRAN language.

CONTENTS

1. CALCULATION OF CONSTITUENT MATRICES AND MATRIX FUNCTIONS

Mathematical Theory

Let the real matrix A have m-pairs of complex conjugate eigenvalues λ_i, $\bar{\lambda}_i$, $i = 1, \ldots m$ where $\lambda_i = \alpha_i + j\beta_i$ and real eigenvalues λ_i, $i = 2m+1, \ldots n$, which are all distinct.

In terms of complex numbers, the constituent matrices are given by the formula

$$Z_i = \prod_{\substack{k=1 \\ k \neq i}}^{n} [(A - \lambda_k U)/(\lambda_i - \lambda_k)]$$

$$i = 1, \ldots n$$

where it can be easily checked that the constituent matrices Z_i, $i = 1, \ldots 2m$ are complex matrices and Z_i, $i = 2m+1, \ldots n$ are real matrices. Explicitly, they are

$$Z_{2i-1} = \prod_{\substack{1 \leq k \leq n \\ k \neq i}} \frac{[A - (\alpha_k + j\beta_k)U]}{[(\alpha_i + j\beta_i) - (\alpha_k + j\beta_k)]} \prod_{\substack{1 \leq k \leq m \\ k \neq i}} \frac{[A - (\alpha_k - j\beta_k)U]}{[(\alpha_i + j\beta_i) - (\alpha_k - j\beta_k)]}$$

$$\prod_{\substack{2m+1 \leq k \leq m \\ k \neq i}} \frac{[A - \lambda_k U]}{[(\alpha_i + j\beta_i) - \lambda_k]} \cdot \frac{A - (\alpha_i - j\beta_i)U}{j2\beta_i}$$

$$= \prod_{\substack{1 \leq k \leq m \\ k \neq i}} \frac{[(A - \alpha_k U)^2 + \beta_k^2 U]}{[(\alpha_i - \alpha_k) + j(\beta_i - \beta_k)][(\alpha_i - \alpha_k) + j(\beta_i + \beta_k)]} \cdot$$

$$\prod_{\substack{2m+1 \leq k \leq n \\ k \neq i}} \frac{A - \lambda_k U}{(\alpha_i - \lambda_k + j\beta_i)} \cdot \frac{A - (\alpha_i - j\beta_i)U}{j2\beta_i}$$

The constituent matrix Z_{2i} for the eigenvalue $\lambda_{2i} = \bar{\lambda}_{2i-1}$ is

$$Z_{2i} = \prod_{\substack{1 \le k \le m \\ k \ne i}} \frac{[(A - \alpha_k U)^2 + \beta_k^2 U]}{[(\alpha_i - \alpha_k) - j(\beta_i + \beta_k)][(\alpha_i - \alpha_k) - j(\beta_i - \beta_k)]} \cdot$$

$$\prod_{\substack{2m+1 \le k \le n \\ k \ne i}} \frac{A - \lambda_k U}{(\alpha_i - \lambda_k - j\beta_i)} \cdot \frac{A - (\alpha_i + j\beta_i)U}{-j2\beta_i}$$

If Z_{2i-1} and Z_{2i} are written as

$$Z_{2i-1} = A_i + jB_i$$

$$Z_{2i} = A_i - jB_i$$

where A_i and B_i are real matrices then, from direct calculations,

$$A_i = \frac{1}{2} \prod_{\substack{1 \le k \le m \\ k \ne i}} \frac{[(A - \alpha_k U)^2 + \beta_k^2 U]}{[(\alpha_i - \alpha_k)^2 + (\beta_i + \beta_k)^2][(\alpha_i - \alpha_k)^2 + (\beta_i - \beta_k)^2]}$$

$$\prod_{\substack{2m+1 \le k \le n \\ k \ne i}} \frac{A - \lambda_k U}{[(\alpha_i - \lambda_k)^2 + \beta_i^2]} \cdot \frac{d_i(A - \alpha_i U) + c_i \beta_i U}{\beta_i}$$

$$i = 1, \ldots m$$

where the complex number

$$c_i + jd_i = \prod_{\substack{1 \le k \le m \\ k \ne i}} [(\alpha_i - \alpha_k) - j(\beta_i + \beta_k)][(\alpha_i - \alpha_k) - j(\beta_i - \beta_k)]$$

$$\cdot \prod_{2m+1 \le k \le n} (\alpha_i - \lambda_k - j\beta_i)$$

Similarly

$$B_i = -\frac{1}{2} \prod_{\substack{1 \le k \le m \\ k \ne i}} \frac{[(A - \alpha_k U)^2 + \beta_k{}^2 U]}{[(\alpha_i - \alpha_k)^2 + (\beta_i + \beta_k)^2][(\alpha_i - \alpha_k)^2 + (\beta_i - \beta_k)^2]}$$

$$\prod_{\substack{2m+1 \le k \le n \\ k \ne i}} \frac{A - \lambda_k U}{[(\alpha_i - \alpha_k)^2 + \beta_i{}^2]} \cdot \frac{c_i(A - \alpha_i U) - \beta_i dU}{\beta_i}$$

$$i = 1, \ldots m$$

For $i = 2m+1, \ldots n$, the constituent matrices are real and are given as

$$Z_i = \prod_{1 \le k \le m} \frac{[(A - \alpha_k U) - j\beta_k U]\,[A - \alpha_k U) + j\beta_k U]}{[(\lambda_i - \alpha_k) - j\beta_k][(\lambda_i - \lambda_k) + j\beta_k]} \cdot \prod_{\substack{2m+1 \le k \le n \\ k \ne i}} \frac{(A - \lambda_k U)}{(\lambda_i - \lambda_k)}$$

$$= \prod_{1 \le k \le m} \frac{(A - \alpha_k U)^2 + \beta_k{}^2 U}{(\lambda_i - \lambda_k)^2 + \beta_k{}^2} \prod_{\substack{2m+1 \le k \le n \\ k \ne i}} \frac{(A - \lambda_k U)}{(\lambda_i - \lambda_k)}$$

Exponential function of the matrix A is calculated as

$$e^{At} = \sum_{i=1}^{m} [(A_i + jB_i)\, e^{(\alpha_j + j\beta_i)t} + (A_i - jB_i) e^{(\alpha_i - j\beta_i)t}]$$

$$+ \sum_{i=2m+1}^{n} Z_i e^{\lambda_i t}$$

$$= \sum_{i=1}^{m} 2\, e^{\alpha_i t} [A_i \cos \beta_i t - B_i \sin \beta_i t] + \sum_{i=2m+1}^{n} Z_i\, e^{\lambda_i t}$$

Inverse of the matrix A is calculated as

$$A^{-1} = \sum_{i=1}^{m} [(A_i + jB_i) \frac{1}{\alpha_i + j\beta_i} + (A_i - jB_i) \frac{1}{\alpha_i - j\beta_i}] + \sum_{i=2m+1}^{n} \frac{Z_i}{\lambda_i}$$

$$= \sum_{i=1}^{m} 2\, \frac{(A_i \alpha_i - B_i \beta_i)}{\alpha_i^2 + \beta_i^2} + \sum_{i=2m+1}^{n} \frac{Z_i}{\lambda_i}$$

Input Procedure

The following data cards should be used in sequence

1.	Format I2	n
2.	Format I2	m
3.	Format 4F 20.10	A
4.	Format 4F 20.10	α_i, β_i, i=1, ... m
5.	Format 4F 20.10	λ_i i=1, ... n

In the main program before calling the subroutine Lastmat, put in cards for LL = 1, 2 or 3 respectively for exponential,power or inverse of the Matrix. If LL = 1, then also put in a card for TIME = ... If LL = 2, then also put in a card for I POWER = ...

Details of the Program

The programming for calculating constituent matrices is straightforward. The name of the subprogram is CONSMAT.

The subprogram which calculates the matrix functions is also straightforward and has the name LASTMAT.

Auxiliary subroutines used are listed below.

i)	MATADD	Adds a scalar multiple of a square matrix D to another matrix Z, i.e., C = Z + S*D
ii)	MATMULT	Multiplies two matrices Z and D and a scalar S, i.e., C = Z * S * D
iii)	MATEQUI	Equates the matrix Z to C Z = C
iv)	SUBSTI	Converts an nxnxn array ZZ into n matrices of order nxn

```
      PROGRAM MATFUNC
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
10    FORMAT (I2)
12    FORMAT(4F20.10)
      READ 10,N
      READ10,M
      ONE=1.0
      TIV = -1.
      MM=2*M+1
      N1=N-2*M
      READ 12,((A(I,J),J=1,N),I=1,N)
      PRINT 910
910   FORMAT (1H0,19HGIVEN MATRIX -A- IS)
      DO 912 I=1,N
912   PRINT12,(A(I,J),J=1,N)
      IF(M)901,902,901
901   READ 12,(EALP(K),EBET(K),K=1,M)
      PRINT 914
914   FORMAT(1H0,34HCOMPLEX EIGENVALUES HAVE THE PARTS)
918   FORMAT(1H0,4HALP(,I2,2H)=,F15.8, 4HBET(,I2,2H)=,F15.8)
      DO 916 K=1,M
916   PRINT 918,K,EALP(K),K,EBET(K)
902   IF(N1)903,904,903
903   READ 12,(EREL(K),K=1,N1)
      PRINT 920
920   FORMAT(1H0,20HREAL EIGENVALUES ARE)
922   FORMAT(1H0,4HREL(,I2,2H)=,F15.8)
      DO 924 K=1,N1
924   PRINT 922, K,EREL(K)
904   CONTINUE
      DO 16 I=1,N
      DO 16 J=1,N
      O(I,J)=0.0
      IF(I-J)13,14,13
13    U(I,J)=0.
      GO TO 16
14    U(I,J)=1.0
16    CONTINUE
      CALL CONSMAT
20    FORMAT (1H0,8F15.8)
21    FORMAT(1H0,18HCONSTITUENTMATRIX ,I2,2HIS//)
      DO 22 ITH=1,N
      PRINT 21,ITH
      DO 22 I  =1,N
      PRINT 20,(ZZ(I,J,ITH),J=1,N)
22    CONTINUE
      LL = 1
      TIME = 2.
      CALL LASTMAT(LL,TIME,IPOWER,Z)
      PRINT 927
927   FORMAT(1H0,18HMATRIX FUNCTION IS///)
```

```
      DO 930 I=1,N
930   PRINT 12 , (Z(I,J),J=1,N)
      STOP
      END
3000  SUBROUTINE LASTMAT(LL,TIME,IPOWER,Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      CALL MATEQUI (O,Z)
      IF(M)460 ,405,401
401   DO 406 K=1,M
      GO TO(420, 425, 430                             ),LL
402   JTH= 2*K-1
      CALL SUBSTI (   JTH,CC)
      CALL MATADD (Z,S,CC,C)
      GO TO(422, 427, 432                             ),LL
404   JTH =  2*K
      CALL SUBSTI (   JTH,CC)
      CALL MATADD (C,S,CC,D)
      CALL MATEQUI(D,Z)
406   CONTINUE
405   IF(N1) 460,460,407
407   DO 412 K=1,N1
      GO TO (424,429,434                              ),LL
408   JTH = 2* M + K
      CALL SUBSTI(   JTH,CC)
      CALL MATADD (Z,S,CC,D)
      CALL MATEQUI(D,Z)
412   CONTINUE
      GO TO 460
420   S=  EXPF(EALP(K)*TIME)*2.0*COSF(EBET(K)*TIME)
      GO TO 402
422   S= -EXPF(EALP(K)*TIME)*2.0*SINF(EBET(K)*TIME)
      GO TO 404
424   S=EXPF(EREL(K)*TIME)
      GO TO 408
425   IM=IPOWER-1
      ALP1=EALP(K)
      BET1=EBET(K)
      DO 426 KK=1,IM
426   CALL COMMULT(ALP1,BET1,EALP(K),EBET(K))
      S=ALP1 * 2.
      GO TO 402
427   S=BET1 * 2.
      GO TO 404
429   S=EREL(K)**FLOATF(IPOWER)
      GO TO 408
430   S=(EALP(K)/(EALP(K)**2 +EBET(K)**2)) * 2.
      GO TO 402
432   S=(EBET(K)/(EALP(K)**2 +EBET(K)**2)) * 2.
      GO TO 404
434   S= 1./ EREL(K)
      GO TO 408
460   RETURN
      END
```

```
1000  SUBROUTINE CONSMAT
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DENOM=1.0
41    CALL MATEQUI(U,D)
C     FIND REAL AND IMAGINARY PARTS OF COMPLEX CONSTITUENT MATRICES
      IF(M)633,64,633
633   DO 62 ITH=1,M
      UREAL=1.0
      UIMAG=0.0
      DO 50 K=1,M
      IF(ITH-K)519,50,519
519   ALP1= EALP(ITH)-EALP(K)
      BET1=-(EBET(ITH)+EBET(K))
      BET2=-(EBET(ITH)-EBET(K))
      CALL  COMMULT (UREAL,UIMAG,ALP1,BET1)
      CALL  COMMULT (UREAL,UIMAG,ALP1,BET2)
      SQU= ((EALP(ITH)-EALP(K))**2+ (EBET(ITH)+EBET(K))**2)
      SQU= ((EALP(ITH)-EALP(K))**2+ (EBET(ITH)-EBET(K))**2) *SQU
      DENOM =DENOM*SQU
      S=-EALP(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT(C,ONE,C,CC)
      S=EBET(K)**2
      CALL MATADD (CC,S,U,C)
      CALL MATMULT(D,ONE,C,CC)
      CALL MATEQUI(CC,D)
50    CONTINUE
      IF(N1)51,52,51
51    DO 60 K=1,N1
      S=-EREL(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT (D,QNE,C,CC)
      CALL MATEQUI(CC,D)
      ALP1=EALP(ITH)-EREL(K)
      BET1=-EBET(ITH)
      CALL  COMMULT (UREAL,UIMAG,ALP1,BET1)
60    DENOM=DENOM*(ALP1**2 + BET1**2)
52    CONTINUE
      RECIPRO=0.5/(DENOM*EBET(ITH))
      S=-EALP(ITH)*UIMAG +UREAL*EBET(ITH)
      CALL MATMULT(A,UIMAG,U,CC)
      CALL  MATADD(CC,S,U,C)
      CALL MATMULT( D,RECIPRO,C,Z)
      JTH=2*ITH-1
      DO 61  I=1,N
      DO 61  J=1,N
61    ZZ(I,J,JTH)=Z(I,J)
      S=-UREAL
      CALL MATADD(O,S,A,CC)
      S=UREAL*EALP(ITH) +UIMAG*EBET(ITH)
```

```
      CALL MATADD(CC,S,U,C)
      CALL MATMULT(D,RECIPRO,C,Z)
      JTH=2*ITH
      DO 62 I=1,N
      DO 62 J=1,N
62    ZZ(I,J,JTH)=Z(I,J)
C       FIND REAL CONSTITUENT MATRICES
64    IF(N1)67,68,67
67    DO 81 ITH=MM,N
71    CALL MATEQUI(U,D)
      DENOM=1.0
      IF(M)63,72,63
63    DO 72 K=1,M
      DENOM= DENOM*((EREL(ITH -2*M)-EALP(K))**2+EBET(K)**2)
      S=-EALP(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT(C,ONE,C,CC)
      S=EBET(K)**2
      CALL MATADD(CC,S,U,C)
      CALL MATMULT(D,ONE,C,CC)
      CALL MATEQUI(CC,D)
72    CONTINUE
      DO 74 K=1,N1
      IF(ITH-2*M-K)73,74,73
73    DENOM= DENOM* (EREL(ITH-2*M)-EREL(K))
      S=-EREL(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT (D,ONE,C,CC)
      CALL MATEQUI(CC,D)
74    CONTINUE
      S=1.0/DENOM
      CALL MATMULT (D,S,U,Z)
      DO 81 I=1,N
      DO 81 J=1,N
81    ZZ(I,J,ITH)=Z(I,J)
68    RETURN
      END
      SUBROUTINE MATADD  (Z,S,D,C)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     ADDS TWO N,N MATRICES Z+SD=C
      DO 101 I=1,N
      DO 101 J=1,N
101   C(I,J)= Z(I,J)+ S*D(I,J)
      RETURN
      END
      SUBROUTINE MATMULT (Z,S,D,C)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     MULTIPLIES TWO N,N MATRICES  C= Z*S*D
      DO 91 I=1,N
      DO 91 J=1,N
```

```
      C(I,J)=0.0
      DO 91 K=1,N
91    C(I,J)=C(I,J)+ Z(I,K)*D(K,J) *S
      RETURN
      END
      SUBROUTINE COMMULT (A,B,C,D)
C     COMPLEX MULTIPLICATION (APJB)(C+JD)=A+JB
      AA=A
      A=AA*C-B*D
      B=AA*D+B*C
      RETURN
      END
      SUBROUTINE SUBSTI (    JTH,Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DO 240 I=1,N
      DO 240 J=1,N
240   Z(I,J)=ZZ(I,J,JTH)
      RETURN
      END
      SUBROUTINE MATEQUI(C,Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     EQUATES THE MATRIX Z TO  C
      DO 280 I=1,N
      DO 280 J=1,N
280   Z(I,J)=C(I,J)
      RETURN
      END
      END

 3
 0
       0.                    1.                   0.                    -7.
       -7.                   1.                    0.                   -3.75
       -1.
      -.5                     -3.5                        -4.
```

2. SOLUTION OF A LINEAR SYSTEM OF DIFFERENTIAL EQUATIONS TO A STEP-INPUT

Theory For the system of equations

$$\dot{x} = Ax + Bu \quad , \qquad x(0) = x_o$$

the solution is explicitly given as

$$x(t) = e^{At}(x_o + A^{-1}Bu) - A^{-1}Bu$$

Input Procedure

The data cards should be put in the same sequence as in the previous section. In addition, the vector B should be read and the data card for it must follow last, i.e.

6. Format 4F 20.10 B

Program Details

In addition to the previous program, the subprogram SOLVX is added which determines the solution vector. Additional subroutines used are

v) MATVEC — Multiplies a square matrix D with a vector E and a scalar S

EE = D * S * E

vi) VECADD — Adds two vectors E1 and E

EE = E1 + S * E

```
      SUBROUTINE MATVEC(D,  S,E,EE)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     MULTIPLIES THE MATRIX D WITH THE VECTOR E ANS SCALAR S, D*S*E= EE
      DO 720 I=1,N
      EE(I) =0.0
      DO 720 J=1,N
720   EE(I)=EE(I) + D(I,J) *E(J)*S
      RETURN
      END
      SUBROUTINE VECADD(E1,S,E,EE)                                      VECADD
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10), E1(10), E(10), EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     ADDS TWO VECTORS EE =E1 + S* E
      DO 725 K=1,N
725   EE(K) =E1(K) + E(K)   * S
      RETURN
      END
      SUBROUTINE  SOLVX  (AA,ZZA       ,E,   X1,TT,R,UU,X)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10) ,AA(10,10),ZZA(10,10,10),CHVEC (10),X1(10),X(10),
     2Y(10),YY(10),E(10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
728   LL=3
      CALL LASTMAT(LL,TIME,IPOWER,Z)
      CALL MATVEC(Z,UU,E,Y)
      CALL VECADD(X1,ONE,Y,EE)
      LL=1
      TM  =  TT * R
      CALL LASTMAT(LL,TM,IPOWER,Z)
12    FORMAT (4F20.10)
      CALL MATVEC(Z,ONE,EE,YY)
      CALL VECADD(YY,TIV, Y, X )
      RETURN
      END
```

3. INTEGRAL-QUADRATIC-ERROR OF A LINEAR SYSTEM TO A STEP INPUT

Mathematical Formulas

The object is to explicitly evaluate the object function

$$J = \int_o^\infty (x - x_{ss})'G(x - x_{ss})dt$$

where x = x(t) is the solution to the linear system

$$\dot{x} = Ax + Bu \qquad\qquad x(o) = 0$$

and $x_{ss} = x(\infty)$ is the steady state of the system. It can be easily checked that when u is a unit step function

$$x - x_{ss} = e^{At}A^{-1}B$$

So that in terms of constituent matrices

$$J = \int_o^\infty B'\Big(\sum_{i=1}^{n}\sum_{k=1}^{n} Z_i'GZ_k \frac{e^{(\lambda_i+\lambda_k)t}}{\lambda_i\lambda_k}\Big)B\,dt$$

$$= (J_1 + J_2 + J_3).$$

where $$J_1 = B'\int_o^\infty \sum_{i=1}^{m}\sum_{k=1}^{m}\{(A_i'+jB_i')G(A_k+jB_k)\frac{e^{[(\alpha_i+j\beta_i)+(\alpha_k+j\beta_k)]t}}{(\alpha_i+j\beta_i)(\alpha_k+j\beta_k)}$$

$$+ (A_i' + jB_i')G(A_k - jB_k)\frac{e^{[(\alpha_i+j\beta_i)+(\alpha_k-j\beta_k)]t}}{(\alpha_i+j\beta_i)(\alpha_k-j\beta_k)}$$

$$+(A_i'-jB_i')G(A_k+jB_k)\frac{e^{[(\alpha_i-j\beta_i)+(\alpha_k+j\beta_k)]t}}{(\alpha_i-j\beta_i)+(\alpha_k+j\beta_k)} + (A_i'-jB_i')G(A_k-jB_k)\cdot$$

$$\frac{e^{[(\alpha_i-j\beta_i)+(\alpha_k-j\beta_k)]t}}{(\alpha_i-j\beta_i)+(\alpha_k-j\beta_k)}\}\,dt$$

$$J_2 = B' \int_0^\infty \sum_{i=1}^{m} \sum_{k=2m+1}^{n} \{(A_i' + jB_i') \frac{e^{(\alpha_i + j\beta_i + \lambda_k)t}}{(\alpha_i + j\beta_i)\lambda_k}$$

$$+ (A_i' - jB_i') \frac{e^{(\alpha_i - j\beta_i + \lambda_k)t}}{(\alpha_i - j\beta_i)\lambda_k}\}. GZ_k B dt$$

$$+ B' \int_0^\infty \sum_{k=2m+1}^{n} \sum_{i=1}^{m} Z_k'G \{(A_i + jB_i) \frac{e^{(\alpha_i + j\beta_i + \lambda_k)t}}{(\alpha_i + j\beta_i)\lambda_k}$$

$$+ (A_i - jB_i) \frac{e^{(\alpha_i - j\beta_i + \lambda_k)t}}{(\alpha_i - j\beta_i) + \lambda_k}\} B\, dt$$

$$J_3 = B' \int_0^\infty \sum_{i=2m+1}^{n} \sum_{k=2m+1}^{n} Z_i'GZ_k \frac{e^{(\lambda_i + \lambda_k)t}}{\lambda_i \lambda_k} B\, dt$$

Upon integration and simplification, algebraic expressions in real domain are obtained as

$$J_1 = B' \sum_{i=1}^{m} \sum_{k=1}^{m} \frac{2}{(\alpha_i^2 + \beta_i^2)(\alpha_k^2 + \beta_k^2)} \left\{ \frac{(\alpha_i + \alpha_k)(P_i'GP_k + Q_i'GQ_k)}{[(\alpha_i + \alpha_k)^2 + (\beta_i - \beta_k)^2]} \right.$$

$$+ \frac{(\beta_i - \beta_k)(P_i'GQ_k - Q_i'GP_k)}{[(\alpha_i + \alpha_k)^2 + (\beta_i - \beta_k)^2]} +$$

$$+ \frac{(\alpha_i + \alpha_k)(P_i'GP_k - Q_i'GQ_k) - (\beta_i + \beta_k)(P_i'GQ_k + Q_i'GP_k)}{[(\alpha_i + \alpha_k)^2 + (\beta_i + \beta_k)^2]} \Bigg\} B$$

$$J_2 = B' \sum_{i=1}^{m} \sum_{k=2m+1}^{n} \Bigg\{ \frac{2[(\alpha_i + \lambda_k) P_i'GZ_k - \beta_i Q_i'GZ_k]}{(\alpha_i^2 + \beta_i^2)\lambda_k[(\lambda_k + \alpha_i)^2 + \beta_i^2]}$$

$$+ \frac{2[(\alpha_i + \lambda_k) Z_k'GP_i - \beta_i Z_k'GQ_i]}{(\alpha_i^2 + \beta_i^2)\lambda_k[(\lambda_k + \alpha_i)^2 + \beta_i^2]} \Bigg\} B$$

and

$$J_3 = B' \sum_{i=2m+1}^{n} \sum_{k=2m+1}^{n} \frac{Z_i'GZ_k}{\lambda_i \lambda_k (\lambda_i + \lambda_k)} B$$

where

$$P_i = A_i \alpha_i + B_i \beta_i \qquad Q_i = A_i \beta_i - B_i \alpha_i$$

Input Procedure

The data cards should be put in in the same sequence as in Section 1. In addition, the weighting matrix G and vector B should be read. The data for them must follow last

6. Format 4F 20.10 G
7. Format 4F 20.10 B

Details of the Program

In addition to program of Section 1, the subprogram, INDEX, is added which determines the object function. Additional subroutines used are

v) TRANSPO — Equates the matrix Z to the transpose of C, i.e.,

$Z = C'$

. MINIMIZATION OF AN OBJECT FUNCTION BY THE METHOD OF STEEPEST DESCENT

(Integral-Quadratic-Error Object Function)

The block diagram in Fig. 1 shows the scheme for minimization of an object function for a linear system with respect to a set of parameters.

Input Procedure

1. The following data cards should be put in the same sequence

1. Format I2	n
2. Format I2	m
3. Format 4F 20.10	A
4. Format I2	order of parameter vector
5. Format 4F 20.10	Initial Parameter
6. Format 4F 20.10	Minimum, maximum values of the parameters
7. Format 4F 20.10	Initial Eigenvalues
8. Format 4F 20.10	Weighting matrix G
9. Format 4F 20.10	B

2. In the main program, an initial value of DM should be adjusted.

3. Subroutine NEWMAT must be so adjusted that the new values of the parameter vector are assigned to form the new matrix.

4. In subroutine LEMPARA adjust the cards from 856 to 857 so as to generate entries of the matrix $\left[\frac{\partial \lambda^*}{\partial a}\right]$.

5. Adjust cards 224 to 212 in ITERATE to introduce differential equations which enable to determine the new eigenvalues for a new parameter vector.

Details of the Program

The following auxiliary subprograms are used.

i)	JACVEC	determines the vector $\left(\frac{\partial J}{\partial \lambda^*}\right)$
ii)	VECSEP	Separates λ^* into complex and real eigenvalues
iii)	VECJOIN	Joins complex and real eigen-values to form λ^*
iv)	VECEQUI	Equates the vector EE to the vector E times a scalar S.
v)	JACPARA	Determine the vector $\left(\frac{\partial J}{\partial a}\right)$
vi)	NEWMAT	Forms the new matrix A for a new parameter vector
vii)	CONSMAT	Finds the constituent matrices and stores in common as an nxnxn array
viii)	INDEX	Calculates the object function 'PINDEX'
ix)	MATADD	Adds a scalar multiple of the square matrix D to the matrix Z
x)	MATMULT	Multiplies two square matrices and a scalar
xi)	COMMULT	Multiplies two complex numbers
xii)	SUBSTI	Separates the nxnxn array ZZ into n matrices nxn.
xiii)	TRIEQUI	Equates two nxnxn matrices.
xiv)	MATCOMP	Gives $P_i = A_i\alpha_i + B_i\beta_i$ $Q_i = A_i\beta_i - B_i\alpha_i$

xv)	TRANPO	Takes the transpose of a matrix Z = C'
xvi)	MATEQUI	Equates the matrix Z to C
xvii)	LEMPARA	Generates the Matrix $\left[\frac{\partial \lambda^*}{\partial a}\right]$
xviii)	STABLE	Checks stability and distinctness of eigenvalues
xix)	ITERATE	(By courtesy of Mr. A. Reiter) Integrates a system of differential equations.

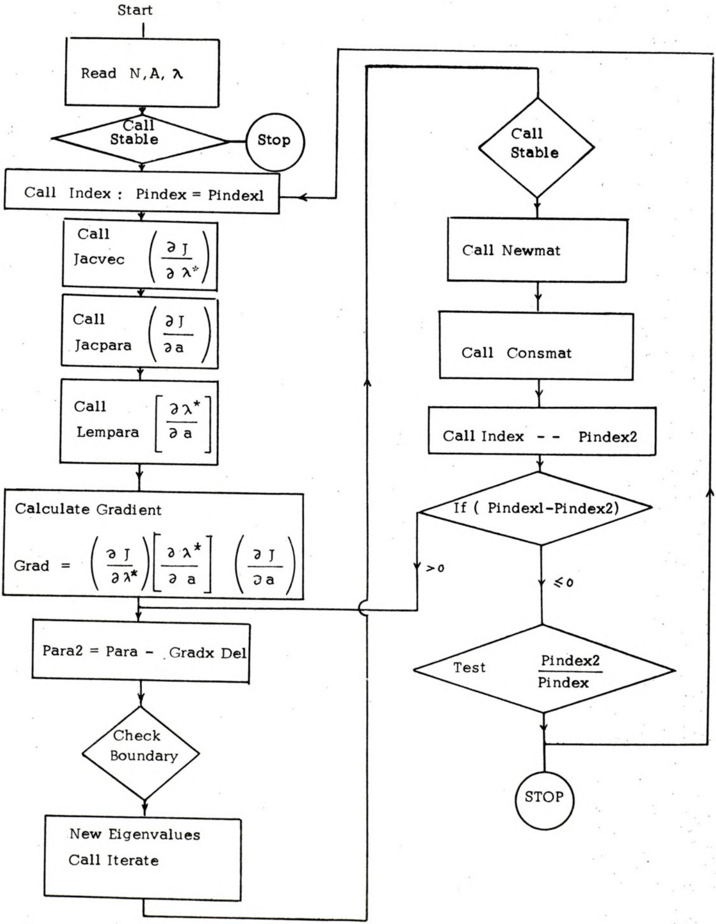

Start
Read N, A, λ
Call Stable
Stop
Call Index : Pindex = Pindexl
Call Jacvec (∂J/∂λ*)
Call Jacpara (∂J/∂a)
Call Lempara [∂λ*/∂a]
Calculate Gradient
Grad = (∂J/∂λ*)[∂λ*/∂a] (∂J/∂a)
Para2 = Para - Gradx Del
Check Boundary
New Eigenvalues
Call Iterate
Call Stable
Call Newmat
Call Consmat
Call Index -- Pindex2
If (Pindexl-Pindex2)
>0
≤0
Test Pindex2/Pindex
STOP

Figure 1.

```
      PROGRAM OPTIMUM
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      DIMENSION CHVEC(10),DJDL(10),AA(10,10),PARA(10),DJDPARA(10),P(10,1
     10),GRAD(10),PARA2(10),PARAMIN(10),PARAMAX(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
9090  FORMAT (1H ,39H OPTIMIZATION OF INTEGRAL SQUARE ERROR /////)
      PRINT 9090
10    FORMAT (I2)
12    FORMAT (4F20.10)
      READ 10,N
      READ10,M
      ONE=1.0
      TIV = -1.
      MM=2*M+1
      N1=N-2*M
      READ 12,((A(I,J),J=1,N),I=1,N)
      PRINT 910
910   FORMAT (1H0,19HGIVEN MATRIX -A- IS)
      DO 912 I=1,N
912   PRINT12,(A(I,J),J=1,N)
      READ 10, NPARA
      READ 12, (PARA(K), K=1,NPARA)
      PRINT 928,(KN,PARA(KN), KN=1,NPARA)
928   FORMAT(1H0,31HTHE INITIAL PARAMETER VECTOR IS/ 5(5HPARA(,I2,3H)= ,
     1F5.2,1X))
      READ 12, (PARAMIN(KA),PARAMAX(KA) ,KA=1,NPARA )
925   FORMAT (1H0, 46H MINIMUM AND MAXIMUM VALUES OF PARAMETERS ARE )
      PRINT 925
      DO 926 KA=1,NPARA
926   PRINT 12,     PARAMIN(KA),PARAMAX(KA)
      IF(M)901,902,901
901   READ 12,(EALP(K),EBET(K),K=1,M)
      PRINT 914
914   FORMAT(1H0,34HCOMPLEX EIGENVALUES HAVE THE PARTS)
918   FORMAT(1H0,4HALP(,I2,2H)=,F15.8, 4HBET(,I2,2H)=,F15.8)
      DO 916 K=1,M
916   PRINT 918,K,EALP(K),K,EBET(K)
902   IF(N1)903,904,903
903   READ 12,(EREL(K),K=1,N1)
      PRINT 920
920   FORMAT(1H0,20HREAL EIGENVALUES ARE)
922   FORMAT(1H0,4HREL(,I2,2H)=,F15.8)
      DO 924 K=1,N1
924   PRINT 922, K,EREL(K)
904   CONTINUE
      READ 12,((G(I,J),J=1,N),I=1,N)
950   FORMAT (1H0,25H THE WEIGHTING MATRIX IS )
      PRINT 950
      DO 952 IA=1,N
952   PRINT 12, ( G(IA,JA), JA=1,N )
      READ 12,( E(J),J=1,N)
951   FORMAT (1H0,  20H THE MATRIX -B- IS   )
```

```
      PRINT 951
      PRINT 12, ( E(JA), JA=1,N )
      DO 16 I=1,N
      DO 16 J=1,N
      O(I,J)=0.0
      IF(I-J)13,14,13
13    U(I,J)=0.
      GO TO 16
14    U(I,J)=1.0
      LSTOP= 0
16    CONTINUE
      CALL STABLE
      PRINT 930
930   FORMAT (1H0,34HTHE SYSTEM IS STABLE INITIALLY AND)
      CALL CONSMAT
2222  CALL INDEX(G,E,PINDEX)
      L1094 = 0
      DM = .1
23    FORMAT (1H0,21HPERFORMANCE INDEX IS=,F15.8,//////)          CYCLE ST
      PRINT 23,PINDEX
      CALL VECJOIN(CHVEC)
      CALL        JACVEC(CHVEC,DJDL, G,E)
      CALL VECSEP (CHVEC)
982   FORMAT(15H0VECTOR DJDL IS)
      PRINT 982
      PRINT 12, (DJDL(KT), KT=1,N )
      CALL        JACPARA( G,E,PARA ,DJDPARA,NPARA)
984   FORMAT(18H0VECTOR DJDPARA IS)
      PRINT 984
      PRINT 12, (DJDPARA(KT), KT=1,NPARA )
      CALL        LEMPARA(NPARA,PARA,P         )                   START
      PRINT 986
986   FORMAT(1H0,22HTHE MATRIX LEMPARA IS )
      DO 988 IA=1,NPARA
988   PRINT 12 ,(P(IA,JA),JA=1,N)
      KTH = 0
      DO 1010 IA=1,NPARA
1004  GRAD(IA) = DJDPARA(IA)
      DO 1010 JA=1,N
1010  GRAD(IA) = GRAD(IA) + P(IA,JA) * DJDL(JA)
1925  FORMAT (1H ,14H GRADIENT IS  ,10F8.4//)
      PRINT1925, (GRAD(KA), KA=1,NPARA)
      L1078= 1
      PINDEX1 = PINDEX
      CALL VECJOIN ( EE )
1007  KTH = KTH + 1
      DEL= EXPF(DM*FLOATF(KTH))  - 1.
      PRINT 1930, DEL
1930  FORMAT(1H ,6HDEL IS,F20.10)                                  OUT
C     DM  MAY BE ADJUSTED ABOVE                                    ADJUST
      DO 1014 IA=1,NPARA
1014  PARA2(IA)= PARA(IA) -GRAD(IA) *DEL
      DO 1023 L=1,NPARA
```

```
      IF(PARA2(L)-PARAMIN (L) )1020,1020,1021
1020  PARA2 (L) = PARAMIN (L)
1025  FORMAT (1H ,12H PARAMETER  ,I2,12H  IS CLIPPED )
      PRINT 1025, L
1021  IF(PARA2(L)-PARAMAX( L) )1023,1022,1022
1022  PARA2(L) = PARAMAX (L)
      PRINT 1025, L
1023  CONTINUE
      CALL VECJOIN(CHVEC)
1065  DO 1072 K=1,NPARA
      NNT =  0
      T = PARA(K)
      DT= (PARA2(K) - PARA(K))/ 10.
      PRINT 12, DT
      CT = DT
      IF(DT) 1071,1072,1071
1071  CALL ITERATE (CHVEC , T,DT,K,PARA)
      DT = CT
      NNT = NNT + 1
      IF (NNT - 12) 1073,5000,5000
1073  IF(ABSF   ( (PARA2(K)-T)/PARA2(K)) -.01)1072,1072,1071
1072  CONTINUE
      CALL VECSEP(CHVEC )
      CALL STABLE
      DO 1077 K=1,NPARA
1077  CALL NEWMAT (PARA2, NPARA, A, K)
      CALL CONSMAT
      CALL INDEX (G,E,PINDEX2)
      CALL VECSEP (EE)
      GO TO (1078,1090),L1078
1078  IF(PINDEX1- PINDEX2)1079,1079,1074
1074  PINDEX1= PINDEX2
      PRINT 1920,(PARA2(KA),KA=1,NPARA)
1920  FORMAT(1H ,35H           PARAMETER VECTOR IS          ,/10(5X,F15.8))
      PRINT 1910,  PINDEX2
1910  FORMAT(1H ,39H                          IMPROVED INDEX IS,F15.8)
      GO TO 1007
1079  IF(KTH - 4)1081,1081,1085
1081  DM = DM* EXPF( FLOATF(KTH)   )/ 500.
      KTH = 0
1092  FORMAT (1H0,40X,9H WENT TO ,I5,5X,F15.8, 3(F10.5,1X)   )
      I1082 =1082
      PRINT 1092,I1082, PINDEX2,(PARA2(IA),IA=1,NPARA )
      L1094 = L1094 + 1
      IF (L1094 - 3) 1084, 1085,1085
1085  L1078 = 2
      L1094 = 0
      KTH = KTH - 2
      I1083 =1083
      PRINT 1092,I1083, PINDEX2,(PARA2(IA),IA=1,NPARA )
1084  GO TO 1007
1090  CONTINUE
      PRINT 12, DM
      DM = DM * FLOATF(KTH+ 1   )/10.
```

```
      PRINT 12, DM
      DO 1075 KA=1,NPARA
1075  PARA(KA) = PARA2(KA)
      CALL VECSEP (CHVEC)
      PINDEX = PINDEX2
1080  LSTOP = LSTOP + 1
      IF( LSTOP - 5) 23,23,5000
5000  STOP                                                          5000
      END
      SUBROUTINE JACVEC(CHVEC,DJDL, G,E)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      DIMENSION CHVEC(10),DJDL(10),DER(2)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     HOLD THE MATRIX A   AND CONSTITUENT MATRICES CONSTANT.   CHANGE CHVEC
      DO 612 K=1,N
      DL =.01
605   DO 610 MR=1,2
607   CH    =CHVEC(K)
      CHVEC(K) = CH     * (1. +DL)
      CALL VECSEP (CHVEC)
      CALL INDEX (G,E,PINDEX2)
      CHVEC(K) = CH     * (1.-DL)
      CALL VECSEP(CHVEC)
      CALL INDEX (G,E,PINDEXO)
12    FORMAT (4F20.10)
      DER(MR) = (PINDEX2 - PINDEXO)/(2.0*DL*CH)
      CHVEC(K) = CH
610   DL =0.1*DL
      IF(.00
     11-ABSF((DER(2)-DER(1))/(.0001*DER(1)-.9999*DER(2))))606,612,612
606   MR = 2
      DER(1) =  DER(2)
      GO TO 607
612   DJDL(K) = DER(2)
      RETURN
      END
      SUBROUTINE VECSEP (EE)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     SEPERATES THE VECTOR EE INTO REAL AND COMPLEX PARTS
      IF(M)673,674,673
673   DO 671 K=1,M
      EALP(K)=EE(2*K-1)
671   EBET(K) =EE(2*K)
674   IF(N1)669,670,669
669   DO 672 K=1,N1
672   EREL(K) =EE(2*M+K)
670   RETURN
      END
      SUBROUTINE VECJOIN(EE)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
```

```
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     JOINS THE COMPLEX AND REAL PARTS OF THE VECTOR
      IF(M)677,678,677
677   DO 675 K=1,M
      EE(2*K-1) = EALP(K)
675   EE(2*K)   = EBET(K)
678   IF(N1)679,680,679
679   DO 676 K=1,N1
676   EE(2*M+K) = EREL(K)
680   RETURN
      END
      SUBROUTINE VECEQUI(E,S,EE)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     MULTIPLIES A VECTOR  WITH SCALAR EE = E * S
      DO 678 K=1,N
678   EE(K) =E(K) * S
      RETURN
      END
      SUBROUTINE JACPARA( G,E,PARA , DJDPARA, NPARA )
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      DIMENSION CHVEC(10),          DER(2),PARA(10),AA(10,10),ZZA(10,10,1
     10),DJDPARA(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     HOLD THE EIGENVALUES CONSTANT BUT VARY  A   AND  ZZ
      CALL MATEQUI(A,AA)
      CALL TRIEQUI (ZZ,ZZA)
      CALL VECJOIN(CHVEC)
      CALL VECJOIN(EE)
      DO 612 K=1,NPARA
      DL =.01
605   DO 610 MR=1,2
607   PA  = PARA(K)
      PARA(K) = PA * (1. - DL)
      T = PA
      LT = 0
      PINDEX2 = 0.
608   DT= (PARA(K) - PA)
      PINDEX0 = PINDEX2
      LT = LT + 1
      IF(DT) 1071,1072,1071
1071  CALL ITERATE (CHVEC , T,DT,K,PARA)
1072  CALL VECSEP (CHVEC)
      CALL NEWMAT(PARA  ,NPARA, AA,K)
      CALL CONSMAT
      CALL VECSEP (EE)
      CALL INDEX (G,E,PINDEX2)
      CALL VECEQUI (EE, ONE, CHVEC)
      PARA(K) = PA * (1. + DL)
      GO TO (608, 609) , LT
609   DER(MR) = (PINDEX2 - PINDEX0)/(2.0*DL*PA)
```

```
602    FORMAT (2F20.10)
       PARA(K) = PA
610    DL =0.1*DL
       IF(.00
      11-ABSF((DER(2)-DER(1))/(.0001*DER(1)-.9999*DER(2))))606,612,612
606    MR = 2
       DER(1) =  DER(2)
       GO TO 607
612    DJDPARA(K) = DER(2)
       CALL TRIEQUI (ZZA, ZZ)
       CALL MATEQUI(AA,A)
       CALL VECSEP (EE)
       RETURN
       END
       SUBROUTINE NEWMAT(PARA ,NPARA,AA,K)
       DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
      1REL(10),AA(10,10),PARA(10)
       COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
       GO TO (801 ,802                        ), K
801    A(2,1) = - PARA (1)
802    A(3,3) = -PARA(2)
       RETURN
       END
1000   SUBROUTINE CONSMAT
       DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
      1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
       COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
       DENOM=1.0
41     CALL MATEQUI(U,D)
C      FIND REAL AND IMAGINARY PARTS OF COMPLEX CONSTITUENT MATRICES
       IF(M)633,64,633
633    DO 62 ITH=1,M
       UREAL=1.0
       UIMAG=0.0
       DO 50 K=1,M
       IF(ITH-K)519,50,519
519    ALP1= EALP(ITH)-EALP(K)
       BET1=-(EBET(ITH)+EBET(K))
       BET2=-(EBET(ITH)-EBET(K))
       CALL  COMMULT (UREAL,UIMAG,ALP1,BET1)
       CALL  COMMULT (UREAL,UIMAG,ALP1,BET2)
       SQU= ((EALP(ITH)-EALP(K))**2+ (EBET(ITH)+EBET(K))**2)
       SQU= ((EALP(ITH)-EALP(K))**2+ (EBET(ITH)-EBET(K))**2) *SQU
       DENOM =DENOM*SQU
       S=-EALP(K)
       CALL MATADD (A,S,U,C)
       CALL MATMULT(C,ONE,C,CC)
       S=EBET(K)**2
       CALL MATADD (CC,S,U,C)
       CALL MATMULT(D,ONE,C,CC)
       CALL MATEQUI(CC,D)
50     CONTINUE
       IF(N1)51,52,51
51     DO 60 K=1,N1
       S=-EREL(K)
```

```
      CALL MATADD (A,S,U,C)
      CALL MATMULT (D,ONE,C,CC)
      CALL MATEQUI(CC,D)
      ALP1=EALP(ITH)-EREL(K)
      BET1=-EBET(ITH)
      CALL  COMMULT (UREAL,UIMAG,ALP1,BET1)
60    DENOM=DENOM*(ALP1**2 + BET1**2)
52    CONTINUE
      RECIPRO=0.5/(DENOM*EBET(ITH))
      S=-EALP(ITH)*UIMAG +UREAL*EBET(ITH)
      CALL MATMULT(A,UIMAG,U,CC)
      CALL  MATADD(CC,S,U,C)
      CALL MATMULT( D,RECIPRO,C,Z)
      JTH=2*ITH-1
      DO 61 I=1,N
      DO 61 J=1,N
61    ZZ(I,J,JTH)=Z(I,J)
      S=-UREAL
      CALL MATADD(O,S,A,CC)
      S=UREAL*EALP(ITH) +UIMAG*EBET(ITH)
      CALL MATADD(CC,S,U,C)
      CALL MATMULT(D,RECIPRO,C,Z)
      JTH=2*ITH
      DO 62 I=1,N
      DO 62 J=1,N
62    ZZ(I,J,JTH)=Z(I,J)
C       FIND REAL CONSTITUENT MATRICES
64    IF(N1)67,68,67
67    DO 81 ITH=MM,N
71    CALL MATEQUI(U,D)
      DENOM=1.0
      IF(M)63,72,63
63    DO 72 K=1,M
      DENOM= DENOM*((EREL(ITH -2*M)-EALP(K))**2+EBET(K)**2)
      S=-EALP(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT(C,ONE,C,CC)
      S=EBET(K)**2
      CALL MATADD(CC,S,U,C)
      CALL MATMULT(D,ONE,C,CC)
      CALL MATEQUI(CC,D)
72    CONTINUE
      DO 74 K=1,N1
      IF(ITH-2*M-K)73,74,73
73    DENOM= DENOM* (EREL(ITH-2*M)-EREL(K))
      S=-EREL(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT (D,ONE,C,CC)
      CALL MATEQUI(CC,D)
74    CONTINUE
      S=1.0/DENOM
      CALL MATMULT (D,S,U,Z)
      DO 81 I=1,N
      DO 81 J=1,N
```

```
      CALL MATADD (A,S,U,C)
      CALL MATMULT (D,ONE,C,CC)
      CALL MATEQUI(CC,D)
      ALP1=EALP(ITH)-EREL(K)
      BET1=-EBET(ITH)
      CALL  COMMULT (UREAL,UIMAG,ALP1,BET1)
60    DENOM=DENOM*(ALP1**2 + BET1**2)
52    CONTINUE
      RECIPRO=0.5/(DENOM*EBET(ITH))
      S=-EALP(ITH)*UIMAG +UREAL*EBET(ITH)
      CALL MATMULT(A,UIMAG,U,CC)
      CALL  MATADD(CC,S,U,C)
      CALL MATMULT( D,RECIPRO,C,Z)
      JTH=2*ITH-1
      DO 61  I=1,N
      DO 61  J=1,N
61    ZZ(I,J,JTH)=Z(I,J)
      S=-UREAL
      CALL MATADD(O,S,A,CC)
      S=UREAL*EALP(ITH) +UIMAG*EBET(ITH)
      CALL MATADD(CC,S,U,C)
      CALL MATMULT(D,RECIPRO,C,Z)
      JTH=2*ITH
      DO 62  I=1,N
      DO 62  J=1,N
62    ZZ(I,J,JTH)=Z(I,J)
C       FIND REAL CONSTITUENT MATRICES
64    IF(N1)67,68,67
67    DO 81  ITH=MM,N
71    CALL MATEQUI(U,D)
      DENOM=1.0
      IF(M)63,72,63
63    DO 72 K=1,M
      DENOM= DENOM*((EREL(ITH -2*M)-EALP(K))**2+EBET(K)**2)
      S=-EALP(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT(C,ONE,C,CC)
      S=EBET(K)**2
      CALL MATADD(CC,S,U,C)
      CALL MATMULT(D,ONE,C,CC)
      CALL MATEQUI(CC,D)
72    CONTINUE
      DO 74 K=1,N1
      IF(ITH-2*M-K)73,74,73
73    DENOM= DENOM* (EREL(ITH-2*M)-EREL(K))
      S=-EREL(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT (D,ONE,C,CC)
      CALL MATEQUI(CC,D)
74    CONTINUE
      S=1.0/DENOM
      CALL MATMULT (D,S,U,Z)
      DO 81  I=1,N
      DO 81  J=1,N
```

```
31     ZZ(I,J,ITH)=Z(I,J)
58     RETURN
       END
2000   SUBROUTINE INDEX(G,E,PINDEX)
       DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
      1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
       DIMENSION PI (10,10),QI (10,10),PK (10,10),QK (10,10)
       DIMENSION PTI(10,10),QTI(10,10),PTK(10,10),QTK(10,10)
       COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C      CALCULATE INDEX, BOTH CONST MATRICES COMPLEX
108    CALL MATEQUI(O,Z)
       IF(M)109,140,109
109    DO 110 I=1,M
130    IPI=2*I-1
       CALL SUBSTI   (IPI,PI)
       IQI=2*I
       CALL SUBSTI   (IQI,QI)
       CALL MATCOMP(PI,QI)
       CALL TRANSPO(PI,PTI)
       CALL TRANSPO(QI,QTI)
       DO 110 K=1,M
       IPK=2*K-1
       CALL SUBSTI(   IPK,PK)
       IQK=2*K
       CALL SUBSTI   (IQK,QK)
       CALL MATCOMP(PK,QK)
       CALL TRANSPO(PK,PTK)
       CALL TRANSPO(QK,QTK)
       DENOM= (EALP(I)**2+EBET(I)**2)*(EALP(K)**2+EBET(K)**2)
       ALP1=EALP(I)+EALP(K)
       BET1=EBET(I)+EBET(K)
       BET2= EBET(I)-EBET(K)
       UNUM1= ALP1/(ALP1**2+BET2**2)
       UNUM2= BET2/(ALP1**2+BET2**2)
       UNUM3= ALP1/(ALP1**2+BET1**2)
       UNUM4= BET1/(ALP1**2+BET1**2)
       CALL MATMULT(PTI,UNUM1,G,C)
       CALL MATMULT (C,ONE,PK,CC)
       CALL  MATADD (O,ONE,CC,D)
       CALL MATMULT (QTI,UNUM1,G,C)
       CALL MATMULT (C,ONE,QK,CC)
       CALL  MATADD (D,ONE,CC,C)
       CALL MATMULT (PTI,UNUM2,G,CC)
       CALL MATMULT (CC,ONE,QK,D)
       CALL  MATADD (D,ONE,C,CC)
       CALL MATMULT(QTI,UNUM2,G,C)
       CALL MATMULT (C,ONE,PK,D)
       CALL  MATADD (CC,TIV,D,C)
       CALL MATMULT (PTI,UNUM3,G,D)
       CALL MATMULT (D,ONE,PK,CC)
       CALL  MATADD (CC,ONE,C,D)
       CALL MATMULT(QTI,UNUM3,G,C)
       CALL MATMULT (C,ONE,QK,CC)
       CALL  MATADD (D,TIV,CC,C)
```

```
      CALL MATMULT (PTI,UNUM4,G,CC)
      CALL MATMULT (CC,ONE,QK,D)
      CALL  MATADD (C,TIV,D,CC)
      CALL MATMULT(QTI,UNUM4,G,C)
      CALL MATMULT (C,ONE,PK,D)
      CALL  MATADD (CC,TIV,D,C)
      S=2.0/DENOM
      CALL MATMULT (C,S,U,D)
      CALL  MATADD (Z,ONE,D,C)
      CALL MATEQUI (C,Z)
110   CONTINUE
C     CALCULATING INDEX.ONEMATRIX COMPLEX, OTHER REAL
      IF(N1)131,150,131
131   DO 140 I=1,M
      IPI=2*I-1
      CALL SUBSTI(   IPI,PI)
      IQI=2*I
      CALL SUBSTI(   IQI,QI)
      CALL MATCOMP(PI,QI)
      CALL TRANSPO(PI,PTI)
      CALL TRANSPO(QI,QTI)
      CONTINUE
      DO 140 K=1,N1
136   IPK=2*M+K
134   CALL SUBSTI(   IPK,PK)
      CALL TRANSPO(PK,PTK)
138   DENOM= (EALP(I)**2+EBET(I)**2)*EREL(K)*((EREL(K)+ EALP(I))**2+EBET
     1(I)**2)
      UNUM1=2.0*(EALP(I)+EREL(K))/DENOM
      UNUM2= -2.0* EBET(I)/DENOM
      CALL MATMULT(PTI,UNUM1,G,C)
      CALL MATMULT (C,ONE,PK,CC)
      CALL  MATADD (O,ONE,CC,D)
      CALL MATMULT(QTI,UNUM2,G,C)
      CALL MATMULT (C,ONE,PK,CC)
      CALL  MATADD (D,ONE,CC,C)
      CALL MATMULT (PTK,UNUM1,G,D)
      CALL MATMULT (D,ONE,PI,CC)
      CALL  MATADD (C,ONE,CC,D)
      CALL MATMULT(PTK,UNUM2,G,C)
      CALL MATMULT (C,ONE,QI,CC)
      CALL  MATADD (D,ONE,CC,C)
      CALL  MATADD (Z,ONE,C,D)
      CALL MATEQUI (D,Z)
140   CONTINUE
C     CA6CULATING INDEX. BOTH CONST. MATRICES REAL
      DO 150  I=1,N1
      IPI= 2*M + I
      CALL SUBSTI(   IPI,PI)
      CALL TRANSPO(PI,PTI)
      DO 150 K=1,N1
      IPK=2*M+K
      CALL SUBSTI(   IPK,PK)
```

```
      CALL TRANSPO(PK,PTK)
      UNUM=-1./(EREL(I)*EREL(K) *(EREL(I)+EREL(K)))
      CALL MATMULT(PTI,UNUM,G,C)
      CALL MATMULT (C,ONE,PK,CC)
      CALL  MATADD (Z,ONE,CC,D)
146   CALL MATEQUI (D,Z)
150   CONTINUE
      PINDEX=0.0
      DO 162 L=1,N
      EE(L)=0.0
      DO 160 J=1,N
160   EE(L)= Z(L,J)*E(J)     +EE(L)
162   PINDEX=E(L)*EE(L)  + PINDEX
      RETURN
      END
      SUBROUTINE VECADD(E1,S,E,EE)                                      VECADD
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10), E1(10), E(10), EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     ADDS TWO VECTORS EE =E1 + S* E
      DO 725 K=1,N
725   EE(K) =E1(K) + E(K)   * S
      RETURN
      END
      SUBROUTINE MATADD  (Z,S,D,C)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     ADDS TWO N,N MATRICES Z+SD=C
      DO 101 I=1,N
      DO 101 J=1,N
101   C(I,J)= Z(I,J)+ S*D(I,J)
      RETURN
      END
      SUBROUTINE MATMULT (Z,S,D,C)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     MULTIPLIES TWO N,N MATRICES  C= Z*S*D
      DO 91 I=1,N
      DO 91 J=1,N
      C(I,J)=0.0
      DO 91 K=1,N
91    C(I,J)=C(I,J)+ Z(I,K)*D(K,J) *S
      RETURN
      END
      SUBROUTINE COMMULT (A,B,C,D)
C     COMPLEX MULTIPLICATION (APJB)(C+JD)=A+JB
      AA=A
      A=AA*C-B*D
      B=AA*D+B*C
      RETURN
      END
      SUBROUTINE SUBSTI (    JTH,Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
```

```
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DO 240 I=1,N
      DO 240 J=1,N
240   Z(I,J)=ZZ(I,J,JTH)
      RETURN
      END
      SUBROUTINE TRANSPO(C, Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DO 250 I=1,N
      DO 250 J=1,N
250   Z (J,I)=C(I,J)
      RETURN
      END
      SUBROUTINE TRIEQUI(ZB,ZZA)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),ZZA(10,10,10),ZB(10,10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DO 710 I =1,N
      DO 710 J =1,N
      DO 710 K =1,N
710   ZZA(I,J,K) = ZB(I,J,K)
      RETURN
      END
      SUBROUTINE MATCOMP  (C ,Z )
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DO 270 I=1,N
      DO 270 J=1,N
      CC(I,J)=EALP(I)*C (I,J)+EBET(I)*Z (I,J)
270   D (I,J)=EBET(I)*C (I,J)-EALP(I)*Z (I,J)
      CALL MATEQUI(CC,C)
      CALL MATEQUI(D ,Z)
      RETURN
      END
      SUBROUTINE MATEQUI(C,Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     EQUATES THE MATRIX Z TO  C
      DO 280 I=1,N
      DO 280 J=1,N
280   Z(I,J)=C(I,J)
      RETURN
      END
      SUBROUTINE LEMPARA(NPARA,PARA,P        )                         LEMPARA
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),PARA(10),P(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C    GENERATES THE TRANSPOSE OF D(LEMDA)/D(PARA)  DIMENSION P(PARA,N)
      IF(M)851,852,851
```

```
851   S=1.
852   IF(N1)856,857,856
856   DO 855 J=1,N1
      JTH= 2 * M + J
      DENOM =3.*EREL(J)**2+(7.+PARA(2))*EREL(J)+7.*PARA(2)+PARA(1)+3.75
      P(1,JTH)=  -(EREL(J) + PARA(2))/ DENOM
855   P(2,JTH)=-( EREL(J)**2 + 7.*EREL(J) + PARA(1) )/DENOM
857   RETURN
      END
      SUBROUTINE STABLE                                                STABLE
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
829   FORMAT(26H0THE REAL PART OF THE PAIR,I2,7HIS ZERO)
830   FORMAT(26H0THE REAL PART OF THE PAIR,I2,11HIS POSITIVE)
831   FORMAT(26H0THE REAL EIGENVALUE       ,I2,7HIS ZERO)
832   FORMAT(26H0THE REAL EIGENVALUE       ,I2,11HIS POSITIVE)
850   FORMAT(26H0THE REAL EIGENVALUE       ,I2,11HIS REPEATED)
851   FORMAT(1H ,23H THE COMPLEX EIGENVALUE,I2,11HIS REPEATED)
      IF(M) 815,816,815
815   CONTINUE
837   FORMAT (1H ,5(F5.2,1X,F5.2,1X) )
      DO 816 K=1,M
      IF(EALP(K)) 817,824,825
817   DO 816 KM=1,M
      IF(KM-K)838,816,838
838   IF(ABSF(( EALP(KM)-EALP(K))/EALP(K))-.001)841,841,816
841   IF(ABSF(( EBET(KM)-EBET(K))/EBET(K))-.001)842,842,816
816   CONTINUE
      IF(N1)818,820,818
818   CONTINUE
839   FORMAT(1H ,10F5.2 )
      DO 820 K=1,N1
      IF(EREL(K))819,826,827
819   DO 820 KM=1,N1
      IF(KM-K)845,820,845
845   IF(ABSF(( EREL(KM)-EREL(K))/EREL(K))-.001) 849,849,820
820   CONTINUE
      GO TO 840
824   PRINT 829,K
      GO TO 835
825   PRINT 830, K
      GO TO 835
842   PRINT 850, K
      GO TO 835
826   PRINT 831,K
      GO TO 835
827   PRINT 832,K
      GO TO 835
849   PRINT 851, K
      GO TO 835
835   STOP 835
840   RETURN
```

```
      END
      SUBROUTINE ITERATE(X, T, DT, LRUNGA, PARA)                         ITERATE
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),C(3,5),X(15), XW(15), RATIO(15), DXDT(15),PARA(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DTMIN = .005
   48 DO 49 I=1,N
   49 XW(I)=X(I)
      TW=T
      L=1
      GO TO 200
   50 DO 51 J=1,N
      C(1,J)=DXDT(J) *DT
   51 X(J)=XW(J)+C(1,J)*0.5
      T=TW+DT*0.5
      L=2
      GO TO 200
   52 DO 53 J=1,N
      C(2,J)=DXDT(J)*DT
   53 X(J)=XW(J)+C(2,J)*0.5
      L=3
      GO TO 200
   54 DO 55 J=1,N
      C(3,J)=DXDT(J)*DT
   55 X(J)=XW(J)+C(3,J)
      T=TW+DT
      L=4
      GO TO 200
   56 DO 25 J=1,N
   25 DXDT(J)=DXDT(J)*DT
      IF(DTMIN-DT) 57,72,72
   57 DO 68 I=1,N
   58 IF(C(1,I)*C(2,I))  61,59,59
   59 IF(C(2,I)*C(3,I))  61,60,60
   60 IF(C(3,I)*DXDT(I)) 61,62,62
   61 DT=DT*0.48
      GO TO 48
   62 XMAX=ABSF(C(1,I))
      XMIN=ABSF(C(1,I))
      DO 66 J=2,4
      IF(J-4)  21,22,22
   22 ZIP=ABSF(DXDT(I))
      GO TO 23
   21 ZIP=ABSF(C(J,I))
   23 IF(XMAX-ZIP)  63,64,64
   63 XMAX=ZIP
      GO TO 66
   64 IF(XMIN-ZIP) 66,66,65
   65 XMIN=ZIP
   66 CONTINUE
      IF(XMIN)  61,61,67
   67 IF((XMAX/XMIN)-1.5) 68,61,61
   68 RATIO(I)=XMAX/XMIN
      DO 69 I=1,N
```

```
      IF(RATIO(I)-1.15) 69,70,70
   69 CONTINUE
   72 TW=TW-DT
      DT=DT+DT
   70 DO 71 I=1,N
   71 XW(I)=XW(I)+(C(1,I)+C(2,I)+C(2,I)+C(3,I)+C(3,I)+DXDT(I))/6.
      TW=TW+DT
      DO 80 I=1,N
   80 X(I)=XW(I)
      T=TW
      RETURN
  200 CONTINUE
      INSERT THE DIFFERENTIAL EQUATIONS AFTER THESE COMMENT CARDS.
      THEY MUST BE VALID FORTRAN STATEMENTS IN THE FOLLOWING FORM...
      DXDT(K)= SOME FUNCTION OF T, X(1),X(2), ... X(N), FOR N LESS THAN 10
      AND K=1 TO N.  (IE, THERE WILL BE EXACTLY N EQUATIONS)
      IF(M)223,224,223
23    GO TO (201                                        ),LRUNGA
01    S=1.
24    CONTINUE
      IF(N1)225,250,225
25    DO 212 J=1,N1
      JM = 2 *M + J
26    GO TO (231, 211),LRUNGA
31    DENOM =3.*  X(J) **2+(7.+PARA(2))*  X(J) +7.*PARA(2)+  T    +3.75
      DXDT(JM) = -( X(JM) + PARA(2) )/DENOM
      GO TO 212
11    DENOM =3.*  X(J) **2+(7.+    T    )*  X(J) +7.*  T    +PARA(1)+3.75
      DXDT(JM)= - (X(J)**2 + 7.*X(J) + PARA(1)   )/DENOM
12    CONTINUE
50    GO TO (50,52,54,56) ,L
      END
      END

3
0
       0.                 1.                 0.                    -2.
       -7.                1.                  0.                   -3.75
       -1.
2                                                                       NPARA
       2.                 1.
      1.                 6.                  .9                    1.1
      -.175836           -1.92962            -5.8945
        1.                 0.                0.                    0.
      0.                 0.                  0.                   0.
       0.
       0.                 0.                 1.
```

4. TRAJECTORY OBJECT-FUNCTION FOR RAMP INPUT

The mathematical formulas have already been given in the main thesis.

Input Procedure

In addition to the data cards as in the previous case, read the cards and put in the data for the matrices C and D and vector Alpha.

10.	Format 4F20.10	C
11.	Format 4F20.10	D
12.	Format 4F20.10	Alpha

Details of Program

Replace the INDEX subroutine in Program 3 for quadratic error by INDEX subroutine for Trajectory error which is given in the following pages. The calling card should also be adjusted accordingly.

The auxiliary subroutines consist of the subroutines in the previous section and

xx)	VECVEC	Finds the innerproduct of two vectors.

```
      SUBROUTINE INDEX (G,E,PINDEX,    C,    D, ALPHA )
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),          C(10,10),CC(10)    ,D(10,10),E(10),G(10,10),EE(10)
     2 ,ZI(10,10),ZK(10,10),ALPHA(10),DD(10),EGI(10),EGK(10),CI(10),CK(1
     3 0),DI(10),DK(10), EI(10), EK(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
2     FORMAT (1H , 5X, I2,2X, 5F20.10   )
      T = 10.
      PINDEX = 0.
      DO 150 I=1,N
      DO 104 LA=1,N
      CI(LA) = C(LA,I)
      DI(LA)= D(LA,I)
      DO 104 LB=1,N
04    ZI(LA,LB) =ZZ(LA,LB,I )
      S= -EREL(I) ** 2
      CALL MATVEC(ZI,S,E , EI)
      CALL MATVEC (G,ONE,EI,EGI)
      EXLI = EXPF(EREL(I) *T)
      EXAI = EXPF(ALPHA(I) *T)
      TLI = EREL(I) *T
      TAI = ALPHA(I) *T
      DO 150 K=1,N
      DO 108 LA=1,N
      CK(LA) = C(LA,K)
      DK(LA)= D(LA,K)
      DO 108 LB=1,N
08    ZK(LA,LB) =ZZ(LA,LB,K )
      S= -EREL(K) ** 2
      CALL MATVEC(ZK,S,E , EK)
      CALL MATVEC (G,ONE,EK,EGK)
      EXLK = EXPF(EREL(K) *T)
      EXAK = EXPF(ALPHA(K) *T)
      TLK = EREL(K) *T
      TAK = ALPHA(K) *T
      AAIK = ALPHA(I) + ALPHA(K)
      SIK = EREL(I) + EREL(K)
      ALIK=ALPHA(I) + EREL(K)
      ALKI=ALPHA(K) + EREL(I)
      EXALIK = EXPF(ALIK* T)
      EXALKI = EXPF(ALKI* T)
      S = (EXPF(SIK) -1.) / SIK
      CALL VECEQUI(EI, S, EE)
      S = ( EXALI* (TLI- 1.)+1.)/((EREL(I)**2)*(EREL(K))   )   +
     1 (EXLI- 1.)/(EREL(I)* EREL(K)  .)
      CALL VECEQUI( E,S,DD)
      CALL VECADD ( EE,ONE,DD,CC)
      CALL VECVEC (EGI, ONE,CC   ,PINDEX2)
      PINDEX  = PINDEX + PINDEX2
      S = (EXLK*(TLK- 1.)+1.)/(EREL(I) *EREL(K)**2)+ (EXLK-1.)/(EREL(K)*
     1EREL(I)**2)
      CALL VECEQUI (EK,S,EE )
      S=(T**3 )/(3.*EREL(I)* EREL(K)) +  (SIK*T**2)/(3.*(EREL(I)**2)*(ER
     1  EL(K)**2) )  + T/ ( (EREL(I)* EREL(K)) **2 )
```

```
      CALL VECADD (EE,S, E,CC)
      CALL MATVEC (G, ONE, EI, EE)
      CALL VECVEC( CC, ONE, EE, PINDEX2 )
      PINDEX = PINDEX + PINDEX2
3     S = ( EALKI - 1.)/ALIK
      CALL VECEQUI (CK, S, CC)
      S= (  (EXALKI*(ALKI*T -1.) +1. )/(EALKI **2)   )
      CALL VECEQUI (DK, S, DD)
      CALL VECADD (CC, ONE, DD, EE)
      CALL VECVEC (EGI, ONE, EE, PINDEX2 )
      PINDEX = PINDEX + PINDEX2
      S= (EXAK*(TAK- 1.)+ 1.)/(EREL(I)*ALPHA(K)**2 ) + (EXLI-1.)/(ALPHA(K)
     1 K) *EREL(I)**2 )
      CALL VECEQUI (CK, S, CC)
      S= (EXAK* (TAK**2 - 2.*TAK +2.) -2. )/(EREL(I)* ALPHA(K)**3) +
     1  (EXAK* (TAK- 1.) + 1.)/ ((EREL(I)*ALPHA(K))**2 )
      CALL VECEQUI (DK, S, DD)
      CALL VECADD (CC, ONE, DD, EE)
      CALL MATVEC (G,ONE ,EE, DD)
      CALL MATVEC ( ZI, ONE, E, CC)
      CALL VECVEC(CC, ONE, DD, PINDEX2 )
      PINDEX = PINDEX + PINDEX2
5     S= (EXALIK- 1. )/ALIK
      CALL VECEQUI (CI,S, CC)
      S= - (EXALIK* (ALIK-1.) +1.) /(ALIK**2)
      CALL VECEQUI (DI, S,DD)
      CALL VECADD(CC, ONE, DD, EE)
      CALL VECVEC (EE,ONE, EK,PINDEX2)
      PINDEX = PINDEX + PINDEX2
      S= (EXAI* (TAI-1.)+ 1.)/(EREL(K)* ALPHA(I)**2) + (EXLK-1.)/ (ALPHA
     1  (I)* EREL(K)**2)
      CALL VECEQUI (CI, S, CC)
      S= (EXAI*(TAI**2 -2. *TAI+ 2.) -2.)/ (EREL(K)*ALPHA(I)**3)   +
     1  (EXAI * (TAI- 1.)+ 1.)/ ((EREL(K)* ALPHA(I) )**2 )
      CALL VECEQUI (DI,S, DD)
      CALL VECADD (CC, ONE,DD,EE)
      CALL MATVEC (G,ONE,EE,CC)
      CALL MATVEC (ZK, ONE,E,DD)
      CALL VECVEC (CC,ONE, DD,PINDEX2)
      PINDEX = PINDEX + PINDEX2
      IF (AAIK) 7 ,22, 7
      S= (EXPF(AAIK*T )-1.)/AAIK
      GO TO 17
2     S= T
7     CALL MATVEC (G,S,CK,CC)
      CALL VECVEC (CI, ONE, CC, PINDEX0 )
      IF (AAIK ) 13,14,13
3     S= (EXPF(AAIK*T) *(AAIK*T- 1.) +1.)/ (AAIK**2 )
      GO TO 18
4     S=(T**2  )*(AAIK*T)
8     CALL MATVEC (G, ONE, CK, CC)
      CALL VECVEC (DI, S, CC, PINDEX1)
      CALL MATVEC (G, ONE, DK, DD)
      CALL VECVEC (CI, S, DD, PINDEX2)
      PINDEX = PINDEX+ PINDEX0+ PINDEX1 +PINDEX2
```

```
      IF (AAIK) 8,16, 8
      S= AAIK* T
      S= EXPF(S) * (S**2 - 2.*S+ 2.)/(AAIK **3)
      GO TO 19
6     S=(T**3) * EXPF(AAIK*T) /3.
9     CALL MATVEC (G, ONE, DK,DD)
      CALL VECVEC (DI, S, DD, PINDEX2)
      PINDEX = PINDEX + PINDEX2
50    CONTINUE
      RETURN
      END
```

5. STABILIZATION OF A LINEAR SYSTEM

Mathematical Theory

The new criterion object function to be used is

$$J = \sum_{i=1}^{n} e^{Re\, \lambda_i}$$

Input Procedure

The data cards to be input are

1.	Format I2	n
2.	Format I2	m
3.	Format 4F 20.10	A
4.	Format I2	No. of parameters
5.	Format 4F 20.10	Parameters
6.	Format 4F 20.10	Minimum, Maximum values of Parameters
7.	Format 4F 20.10	Eigenvalues

Replace the INDEX subprogram by the INDEX subprogram for stabilizing given in the following pages. Replace the subroutine STABLE by subroutine ROOTS given on the next page.

```
      SUBROUTINE INDEX (G,E, PINDEX, EE )
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),E(10),EE(10), CC(10),DD(10),G(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      PINDEX = 0.
      IF(M)151,153,151
51    DO 152 K=1,M
52    PINDEX = PINDEX +         EXPF (EALP(K)   )
53    IF(N1)154,160,154
54    DO 156 K=1,N1
56    PINDEX = PINDEX +         EXPF (EREL(K)   )
50    RETURN
      END
      SUBROUTINE ROOTS
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
51    FORMAT(1H ,23H THE COMPLEX EIGENVALUE,I2,11HIS REPEATED)
52    FORMAT(26H0THE REAL EIGENVALUE       ,I2,11HIS REPEATED)
      IF(M) 815,816,815
15    DO 816 K=1,M
      IF (EALP(K)   ) 817,850,850
17    DO 816 KM=1,M
      IF(KM-K)838,816,838
38    IF(ABSF(( EALP(KM)-EALP(K))/EALP(K))-.001)841,841,816
41    IF(ABSF(( EBET(KM)-EBET(K))/EBET(K))-.001)842,842,816
16    CONTINUE
      IF(N1)818,820,818
18    DO 820 K=1,N1
      IF(EREL(K))819,850,850
19    DO 820 KM=1,N1
      IF(KM-K)845,820,845
45    IF(ABSF(( EREL(KM)-EREL(K))/EREL(K))-.001) 849,849,820
20    CONTINUE
59    FORMAT (1H0,28H THE STABLE EIGENVALUES ARE   , 6(F15.8, 2X )   )
50    PRINT 859,  (EALP(KN),EBET(KN), KN=1,M)   , (EREL(KN),KN=1,N1)
      GO TO 840
42    PRINT 851, K
      GO TO 835
49    PRINT 852, K
35    STOP 835
50    PRINT 869,  (EALP(KN),EBET(KN), KN=1,M)   , (EREL(KN),KN=1,N1)
59    FORMAT (1H0,28H       UNSTABLE   ROOTS       , 6(F15.8, 2X)    )
40    RETURN
      END
```

6. TIME OPTIMAL TRAJECTORIES

Theory

Time optimal control for a linear system of the form

$$\dot{x} = Ax + bu$$

with

$$u \leq 1$$

must satisfy the necessary condition that

$$u = \text{sign } b'x$$

where Ψ is a solution to the adjoint system

$$\Psi = A'$$

i.e.,

$$\dot{\Psi} = e^{-A't}\, \Psi^{o} \qquad\qquad \Psi^{o} = \quad (0)$$

According to a theorem by Pontryagin, for every Ψ^{o}, the above solution determines a unique optimal control and correspondingly a unique optimal trajectory. Thus taking $x(0) = 0$ and Ψ^{o} arbitrary a net of optimal trajectories can be generated by letting time run negatively.

Input Procedure

The following data cards should be put in the same sequence.

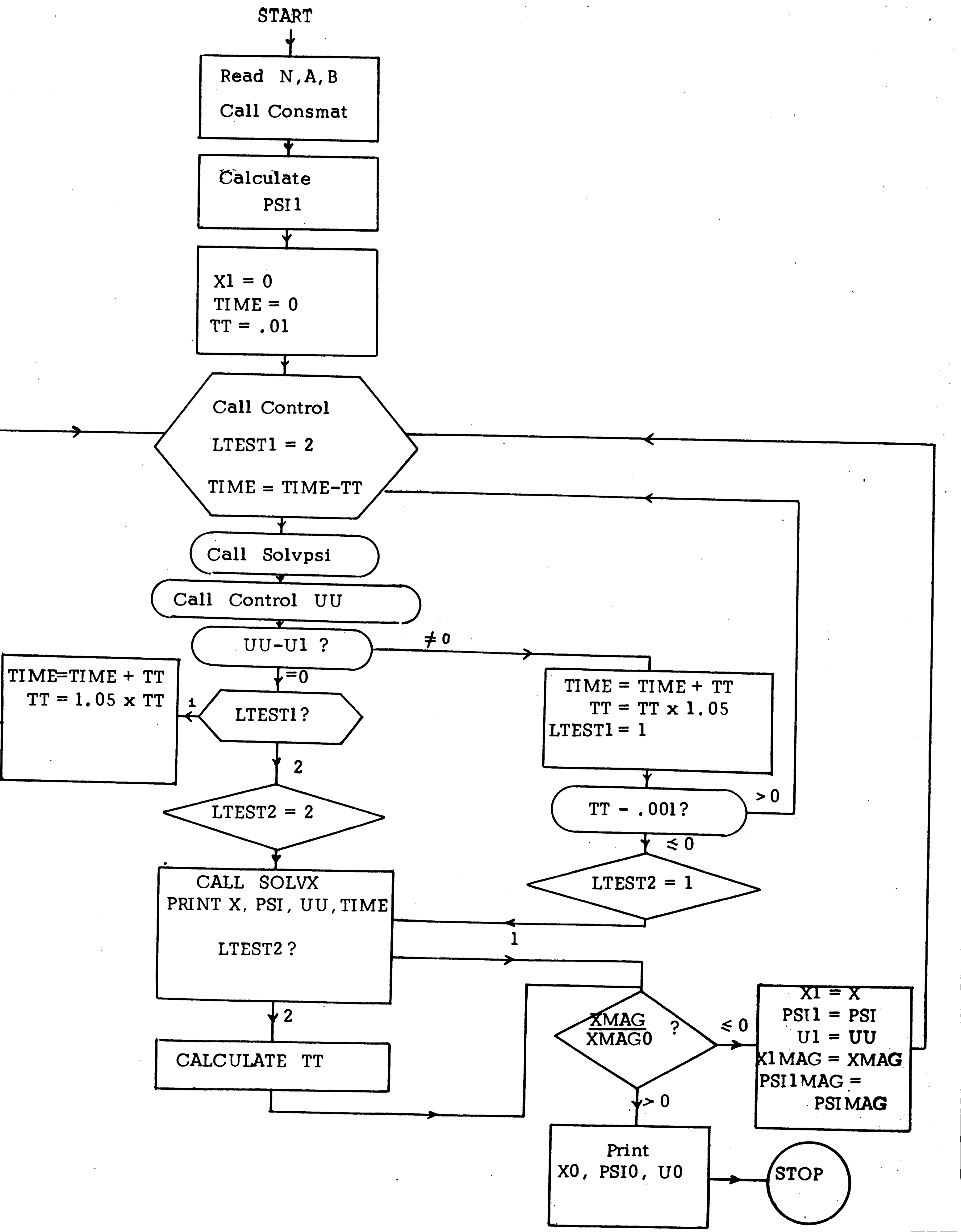
1.	Format I2	n
2.	Format I2	m
3.	Format 4F 20.10	A
4.	Format 4F 20.10	α_i, β_i, $i = 1, \ldots, m$

5. Format 4F20.10 — λ_i, i = 2m+1, ... n

6. Format 4F20.10 — b

7. Format 4F20.10 — Magnitude of Final Value

Figure 2 shows schematically the details of the program.


START
Read N,A,B
Call Consmat
Calculate PSI1
X1 = 0
TIME = 0
TT = .01
Call Control
LTEST1 = 2
TIME = TIME-TT
Call Solvpsi
Call Control UU
UU-U1 ?
≠ 0
=0
TIME = TIME + TT
TT = TT x 1.05
LTEST1 = 1
TIME=TIME + TT
TT = 1.05 x TT
1
LTEST1?
2
LTEST2 = 2
TT - .001?
> 0
≤ 0
LTEST2 = 1
CALL SOLVX
PRINT X, PSI, UU, TIME
LTEST2 ?
1
2
CALCULATE TT
XMAG / XMAG0 ?
≤ 0
X1 = X
PSI1 = PSI
U1 = UU
X1MAG = XMAG
PSI1MAG = PSIMAG
> 0
Print
X0, PSI0, U0
STOP

```
      PROGRAM PONTRY
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      DIMENSION AA(10,10),ZZA(10,10,10),AT(10,10),ZZT (10,10,10),X1(10),PONTRY
     1X(10),PSI1(10),PSI(10),OVEC(10),CHVEC(10),CHVECT(10),PSI2(10)
     2,X2(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
090   FORMAT (1H ,38H PONTRY FOR   COMPLEX   ROOTS         /////)
      PRINT 9090
0     FORMAT (I2)
2     FORMAT(4F20.10)
      READ 10,N
      READ10,M
      ONE=1.0
      TIV = -1.
      MM=2*M+1
      N1=N-2*M
      READ 12,((A(I,J),J=1,N),I=1,N)
      PRINT 910
10    FORMAT (1H0,19HGIVEN MATRIX -A- IS)
      DO 912 I=1,N
12    PRINT12,(A(I,J),J=1,N)
      IF(M)901,902,901
01    READ 12,(EALP(K),EBET(K),K=1,M)
      PRINT 914
14    FORMAT(1H0,34HCOMPLEX EIGENVALUES HAVE THE PARTS)
18    FORMAT(1H0,4HALP(,I2,2H)=,F15.8, 4HBET(,I2,2H)=,F15.8)
      DO 916 K=1,M
16    PRINT 918,K,EALP(K),K,EBET(K)
02    IF(N1)903,904,903
03    READ 12,(EREL(K),K=1,N1)
      PRINT 920
20    FORMAT(1H0,20HREAL EIGENVALUES ARE)
22    FORMAT(1H0,4HREL(,I2,2H)=,F15.8)
      DO 924 K=1,N1
24    PRINT 922, K,EREL(K)
04    CONTINUE
      READ 12,(E(J), J=1,N)
      READ 12, XMAGO
      DO 16 I=1,N
      OVEC(I) =0.
      DO 16 J=1,N
      O(I,J)=0.0
      IF(I-J)13,14,13
3     U(I,J)=0.
      GO TO 16
4     U(I,J)=1.0
5     CONTINUE
      CALL CONSMAT
0     FORMAT (1H0,8F15.8)
1     FORMAT(1H0,18HCONSTITUENTMATRIX ,I2,2HIS//)
      DO 22 ITH=1,N
      PRINT 21,ITH
      DO 22 I  =1,N
```

```
     PRINT 20,(ZZ(I,J,ITH),J=1,N)
2    CONTINUE
     PRINT 1908
908  FORMAT (1H ,42H  AT ,  CHVECT AND ZZT  ARE                    )
     CALL MATEQUI( A,AA)                                                    PONTRY
     CALL TRIEQUI(ZZ,ZZA)
     CALL TRANSPO(AA,C)
     CALL MATADD (O,TIV,C,AT)
910  FORMAT (1H ,8F15.8,/ )
     DO 1911 IA=1,N
911  PRINT 1910,     (AT(IA,JA), JA=1,N )
     CALL VECJOIN(CHVEC)
     CALL VECADD (OVEC, TIV,CHVEC, CHVECT)
     PRINT 1910, (CHVECT(JA) ,JA=1,N)
     CALL VECSEP( CHVECT)
     CALL MATEQUI(AT,A)
     CALL CONSMAT
     CALL TRIEQUI(ZZ,ZZT)
     DO 1912 ITH =1,N
     DO 1912 IA=1,N
912  PRINT  1910,  (ZZT(IA,JA,ITH) ,JA=1,N)
     DIV = 100.
     IW = 210
     IU = 2 *IW - 1
     NNT = 0
     PSI2(1) = -1.
19   PSI2(2)= EXPF(FLOATF(1 -IW)/DIV) -EXPF (- (FLOATF( 1 -IW)/ DIV))
     PSI2(2) =   PSI2(2)   * ABSF(PSI2(2))
     DO 600  KP=1,2
     IF(KP- 1)3201,3201,3202
201  PSI2(1) = -1.
     KR = 120
     KQ = 130
     GO TO 3203
202  PSI2(1) = 1.
     KR = 70
     KQ = 80
203  DO 600 IV = KR,KQ, 2
     IF(IV-IW) 521,528,521
28   IV=IV+1
21   PSI2(2)= EXPF(FLOATF(IV-IW)/DIV) -EXPF (- (FLOATF( IV-IW)/ DIV))
     PSI2(2) =   PSI2(2)   * ABSF(PSI2(2))
     CALL VECEQUI (PSI2, PSI1)
     NNT = NNT + 1
14   FORMAT (1H2,10HTRAJECTORY, I3,6HSTARTS/////)
     PRINT 514,NNT
     PRINT 512, (PSI1(KK), KK=1,N)
12   FORMAT(1H ,17HINITIALLY PSI IS ,10 (1X,E20.10)///)
     X2MAG = 0.
     CALL VECEQUI (OVEC, X2 )
     CALL MAGVEC (PSI1 ,PSI1MAG)
     NNA= 0
     CALL VECEQUI (OVEC,X1)
26   TIME=0.
     TT=.01                                                                 TT
     CALL CONTROL(E,PSI1,U1)
```

```
      PRINT 1900,U1
 900  FORMAT (1H ,7HCONTROL,F5.2)
527   LTEST1 = 2
529   TIME= TIME - TT
      CALL          SOLVPSI(AT,ZZT,CHVECT,PSI1,TT,TIV ,U1,PSI )
      CALL CONTROL(E,PSI,UU)
      IF(UU-U1)533,532,533
532   GO TO(550,551),LTEST1
551   LTEST2 = 2
565   CALL          SOLVX  (AA,ZZA,CHVEC ,E,X1,TT,TIV ,U1,X)
      CALL MAGVEC(X,XMAG)
2920  FORMAT (1H , 9H XMAG IS ,F10.5)
      PRINT 2920, XMAG
      DO 562 KA =1,N
562   EE(KA) = X(KA) - X2(KA)
      CALL MAGVEC (EE, EMAG)
      IF( ABSF(EMAG/(.001*XMAG+ .999*X2MAG) )  - .1 ) 567,568,568
568   NNA = NNA+1
      IF(NNA - 30)590,590,601
590   PRINT 569,NNA,(X(KA),KA=1,N),(PSI(KA),KA =1,N)
569   FORMAT(1H , 20X,6HPOINT(,I3,4H)    ,6(F10.5,5X))
571   FORMAT(1H ,20X, 10HCONTROL IS,F5.2,5X,8HTIME IS ,F6.3)
      PRINT  571,UU,TIME
      X2MAG = XMAG
      CALL VECEQUI (  X , X2 )
      GO TO(560,561),LTEST2
560   IF(XMAG/XMAG0-20.0)567,567,601
567   CALL VECEQUI(X,X1)
      CALL VECEQUI (PSI,PSI1)
      U1=UU
      X1MAG=XMAG
      PSI1MAG=PSIMAG
      GO TO 527
561   CALL MAGVEC(PSI,PSIMAG)
      DO 563 KA =1,N
563   EE(KA) = PSI(KA) - PSI1(KA)
      CALL MAGVEC (EE, EMAG)
      TT =.2  / (1. +  EMAG/PSI1MAG )
C     THE CONSTANTS HERE MAY BE ADJUSTED
      GO TO 560
533   TIME =TIME +TT
      TT=TT/2.
      LTEST1 =1
      IF(TT -    .01) 575,575,529
575   LTEST2=1
      GO TO 565
550   TIME= TIME + TT
      TT = TT* 1.05
      GO TO 527
601   IF (NNT- 40)600,600,5000
600   CONTINUE
5000  STOP
      END
1000  SUBROUTINE LASTMAT(LL,TIME,IPOWER,Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
```

```
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      CALL MATEQUI (O,Z)
      IF(M)460 ,405,401
01    DO 406 K=1,M
      GO TO(420, 425, 430                          ),LL
02    JTH= 2*K-1
      CALL SUBSTI (   JTH,CC)
      CALL MATADD (Z,S,CC,C)
      GO TO(422, 427, 432                          ),LL
04    JTH =  2*K
      CALL SUBSTI (   JTH,CC)
      CALL MATADD (C,S,CC,D)
      CALL MATEQUI(D,Z)
06    CONTINUE
05    IF(N1) 460,460,407
07    DO 412 K=1,N1
      GO TO (424,429,434                           ),LL
08    JTH = 2*M + K
      CALL SUBSTI(   JTH,CC)
      CALL MATADD (Z,S,CC,D)
      CALL MATEQUI(D,Z)
12    CONTINUE
      GO TO 460
20    S=  EXPF(EALP(K)*TIME)*2.0*COSF(EBET(K)*TIME)
      GO TO 402
22    S= -EXPF(EALP(K)*TIME)*2.0*SINF(EBET(K)*TIME)
      GO TO 404
24    S=EXPF(EREL(K)*TIME)
      GO TO 408
25    IM=IPOWER-1
      ALP1=EALP(K)
      BET1=EBET(K)
      DO 426 KK=1,IM
26    CALL COMMULT(ALP1,BET1,EALP(K),EBET(K))
      S=ALP1 * 2.
      GO TO 402
27    S=BET1 * 2.
      GO TO 404
29    S=EREL(K)**FLOATF(IPOWER)
      GO TO 408
30    S=(EALP(K)/(EALP(K)**2 +EBET(K)**2)) * 2.
      GO TO 402
32    S=(EBET(K)/(EALP(K)**2 +EBET(K)**2)) * 2.
      GO TO 404
34    S= 1./ EREL(K)
      GO TO 408
60    RETURN
      END
000   SUBROUTINE CONSMAT
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DENOM=1.0
1     CALL MATEQUI(U,D)
      FIND REAL AND IMAGINARY PARTS OF COMPLEX CONSTITUENT MATRICES
      IF(M)633,64,633
```

```
533   DO 62 ITH=1,M
      UREAL=1.0
      UIMAG=0.0
      DO 50 K=1,M
      IF(ITH-K)519,50,519
519   ALP1= EALP(ITH)-EALP(K)
      BET1=-(EBET(ITH)+EBET(K))
      BET2=-(EBET(ITH)-EBET(K))
      CALL  COMMULT (UREAL,UIMAG,ALP1,BET1)
      CALL  COMMULT (UREAL,UIMAG,ALP1,BET2)
      SQU= ((EALP(ITH)-EALP(K))**2+ (EBET(ITH)+EBET(K))**2)
      SQU= ((EALP(ITH)-EALP(K))**2+ (EBET(ITH)-EBET(K))**2) *SQU
      DENOM =DENOM*SQU
      S=-EALP(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT(C,ONE,C,CC)
      S=EBET(K)**2
      CALL MATADD (CC,S,U,C)
      CALL MATMULT(D,ONE,C,CC)
      CALL MATEQUI(CC,D)
50    CONTINUE
      IF(N1)51,52,51
51    DO 60 K=1,N1
      S=-EREL(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT (D,ONE,C,CC)
      CALL MATEQUI(CC,D)
      ALP1=EALP(ITH)-EREL(K)
      BET1=-EBET(ITH)
      CALL  COMMULT (UREAL,UIMAG,ALP1,BET1)
60    DENOM=DENOM*(ALP1**2 + BET1**2)
52    CONTINUE
      RECIPRO=0.5/(DENOM*EBET(ITH))
      S=-EALP(ITH)*UIMAG +UREAL*EBET(ITH)
      CALL MATMULT(A,UIMAG,U,CC)
      CALL  MATADD(CC,S,U,C)
      CALL MATMULT( D,RECIPRO,C,Z)
      JTH=2*ITH-1
      DO 61  I=1,N
      DO 61  J=1,N
61    ZZ(I,J,JTH)=Z(I,J)
      S=-UREAL
      CALL MATADD(O,S,A,CC)
      S=UREAL*EALP(ITH) +UIMAG*EBET(ITH)
      CALL MATADD(CC,S,U,C)
      CALL MATMULT(D,RECIPRO,C,Z)
      JTH=2*ITH
      DO 62 I=1,N
      DO 62 J=1,N
62    ZZ(I,J,JTH)=Z(I,J)
        FIND REAL CONSTITUENT MATRICES
4     IF(N1)67,68,67
67    DO 81 ITH=MM,N
81    CALL MATEQUI(U,D)
      DENOM=1.0
```

```
      IF(M)63,72,63
63    DO 72 K=1,M
      DENOM= DENOM*((EREL(ITH -2*M)-EALP(K))**2+EBET(K)**2)
      S=-EALP(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT(C,ONE,C,CC)
      S=EBET(K)**2
      CALL MATADD(CC,S,U,C)
      CALL MATMULT(D,ONE,C,CC)
      CALL MATEQUI(CC,D)
72    CONTINUE
      DO 74 K=1,N1
      IF(ITH-2*M-K)73,74,73
73    DENOM= DENOM* (EREL(ITH-2*M)-EREL(K))
      S=-EREL(K)
      CALL MATADD (A,S,U,C)
      CALL MATMULT (D,ONE,C,CC)
      CALL MATEQUI(CC,D)
74    CONTINUE
      S=1.0/DENOM
      CALL MATMULT (D,S,U,Z)
      DO 81 I=1,N
      DO 81 J=1,N
81    ZZ(I,J,ITH)=Z(I,J)
68    RETURN
      END
      SUBROUTINE MATADD  (Z,S,D,C)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      ADDS TWO N,N MATRICES Z+SD=C
      DO 101 I=1,N
      DO 101 J=1,N
101   C(I,J)= Z(I,J)+ S*D(I,J)
      RETURN
      END
      SUBROUTINE MATMULT (Z,S,D,C)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      MULTIPLIES TWO N,N MATRICES  C= Z*S*D
      DO 91 I=1,N
      DO 91 J=1,N
      C(I,J)=0.0
      DO 91 K=1,N
91    C(I,J)=C(I,J)+ Z(I,K)*D(K,J) *S
      RETURN
      END
      SUBROUTINE COMMULT (A,B,C,D)
      COMPLEX MULTIPLICATION (APJB)(C+JD)=A+JB
      AA=A
      A=AA*C-B*D
      B=AA*D+B*C
```

```
      RETURN
      END
      SUBROUTINE SUBSTI (    JTH,Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DO 240 I=1,N
      DO 240 J=1,N
240   Z(I,J)=ZZ(I,J,JTH)
      RETURN
      END
      SUBROUTINE TRANSPO(C, Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DO 250 I=1,N
      DO 250 J=1,N
250   Z (J,I)=C(I,J)
      RETURN
      END
      SUBROUTINE MATEQUI(C,Z)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10), D(10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      EQUATES THE MATRIX Z TO  C
      DO 280 I=1,N
      DO 280 J=1,N
80    Z(I,J)=C(I,J)
      RETURN
      END
      SUBROUTINE TRIEQUI(ZB,ZZA)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),ZZA(10,10,10),ZB(10,10,10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      DO 710 I =1,N
      DO 710 J =1,N
      DO 710 K =1,N
10    ZZA(I,J,K) = ZB(I,J,K)
      RETURN
      END
      SUBROUTINE MAGVEC(EE,EEMAG)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      S=0.0
      DO 715 K=1,N
15    S= S+ EE(K)**2
      EEMAG=SQRTF(S)
      RETURN
      END
      SUBROUTINE VECSEP (EE)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      SEPERATES THE VECTOR EE INTO REAL AND COMPLEX PARTS
```

```
      IF(M)673,674,673
673   DO 671 K=1,M
      EALP(K)=EE(2*K-1)
671   EBET(K) =EE(2*K)
674   IF(N1)669,670,669
669   DO 672 K=1,N1
672   EREL(K) =EE(2*M+K)
670   RETURN
      END
      SUBROUTINE VECJOIN(EE)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C     JOINS THE COMPLEX AND REAL PARTS OF THE VECTOR
      IF(M)677,678,677
677   DO 675 K=1,M
      EE(2*K-1) = EALP(K)
675   EE(2*K)   = EBET(K)
678   IF(N1)679,680,679
679   DO 676 K=1,N1
676   EE(2*M+K) = EREL(K)
680   RETURN
      END
      SUBROUTINE VECEQUI(E,  EE)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
C      EQUATES  THE VECTOR  EE  TO  E
      DO 678 K=1,N
78    EE(K) =E(K)
      RETURN
      END
      SUBROUTINE MATVEC(D,  S,E,EE)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10),C(10,10),CC(10,10),D(10,10),E(10),G(10,10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      MULTIPLIES THE MATRIX D WITH THE VECTOR E ANS SCALAR S, D*S*E= EE
      DO 720 I=1,N
      EE(I) =0.0
      DO 720 J=1,N
20    EE(I)=EE(I) + D(I,J) *E(J)*S
      RETURN
      END
      SUBROUTINE VECADD(E1,S,E,EE)                                      VECADD
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10), E1(10), E(10), EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      ADDS TWO VECTORS EE =E1 + S* E
      DO 725 K=1,N
25    EE(K) =E1(K) + E(K)    * S
      RETURN
      END
      SUBROUTINE  SOLVPSI(AT,ZZT,CHVECT,      X1,TT,R,UU,X)
```

```
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10)  ,AT(10,10),ZZT(10,10,10),CHVECT(10),X1(10),X(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      CALL MATEQUI(AT,A)
      CALL TRIEQUI(ZZT,ZZ)
      CALL VECSEP(CHVECT)
      TM  =  TT * R
      LL=1
      CALL LASTMAT(LL,TM,IPOWER,Z)
 12   FORMAT (4F20.10)
      CALL MATVEC(Z,ONE,X1, X )
      RETURN
      END
      SUBROUTINE  SOLVX  (AA,ZZA,CHVEC ,E,   X1,TT,R,UU,X)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),Z(10,10)  ,AA(10,10),ZZA(10,10,10),CHVEC  (10),X1(10),X(10),
     2Y(10),YY(10),E(10),EE(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      CALL MATEQUI(AA,A)
      CALL TRIEQUI(ZZA,ZZ)
      CALL VECSEP(CHVEC  )
 728  LL=3
      CALL LASTMAT(LL,TIME,IPOWER,Z)
      CALL MATVEC(Z,UU,E,Y)
      CALL VECADD(X1,ONE,Y,EE)
      LL=1
      TM  =  TT * R
      CALL LASTMAT(LL,TM,IPOWER,Z)
 2    FORMAT (4F20.10)
      CALL MATVEC(Z,ONE,EE,YY)
      CALL VECADD(YY,TIV, Y, X )
      RETURN
      END
      SUBROUTINE CONTROL (E,PSI,UU)
      DIMENSIONA(10,10),U(10,10),O(10,10),ZZ(10,10,10),EALP(5),EBET(5),ECOMMON
     1REL(10),E(10),PSI(10)
      COMMON ONE,TIV,N,M,N1,MM,A,U,O,ZZ,EALP,EBET,EREL
      UU=0.0
      DO 730 K=1,N
 730  UU=UU + E(K) * PSI(K)
      UU = SIGNF (ONE, UU )
      RETURN
      END
      END

 2
 1
      0.0                   1.0                   -1.0                  -1.8
       -.9                  .436
       .0                   1.0
        1.                                                                    XMAGO
```