SOLUTION OF A LINEAR REGULATOR PROBLEM WITH QUADRATIC PERFORMANCE INDEX AND STATE VARIABLE CONSTRAINTS

Thesis for the Degree of Ph. D.

MICHIGAN STATE UNIVERSITY
HENRY STANLEY MIKA

1970

LIBRARY
Michigan State
University

This is to certify that the

thesis entitled
SOLUTION OF A LINEAR REGULATOR PROBLEM WITH QUADRATIC PERFORMANCE INDEX AND STATE VARIABLE CONSTRAINTS

presented by

Henry Stanley Mika

has been accepted towards fulfillment
of the requirements for

Ph D degree in *E.E.*

John B. Krae
Major professor

Date *June 24, 1970*

O-169

BINDING BY
HOAG & SONS'
BOOK BINDERY INC.

ABSTRACT

SOLUTION OF A LINEAR REGULATOR PROBLEM
WITH QUADRATIC PERFORMANCE INDEX AND
STATE VARIABLE CONSTRAINTS

By

Henry Stanley Mika

A direct method has been developed to obtain the optimal linear constant control law for the linear, time invariant regulator problem with quadratic performance index and state variable constraints. For a scalar control and a diagonal state weighting matrix Q, it is shown that the constant gain matrix K can be obtained without solving the $n(n+1)/2$ matrix Riccati equations. Instead, it is demonstrated that a simple algebraic algorithm defines all the elements of the K matrix given the n elements forming the bottom row which are simply and linearly related to the n coefficients of the closed loop characteristic equation.

A computational algorithm is developed which consists of two basic parts: a digital program that locates the minimum quadratic cost as a function of n elements of the K matrix; and an hybrid program which checks and, if necessary, adjusts these elements to meet the state variable constraints. This adjustment is implemented by transforming the problem into an n-parameter optimization problem and using sensitivity techniques.

The minimum quadratic cost search yields a global minimum and, if the corresponding K matrix yields a feedback control such that all state variable constraints are satisfied, then this is the desired solution. If adjustment of the K matrix is required to meet state variable constraints, then the hybrid program determines an upper bound for the desired solution. Thus the desired solution is bounded by the global minimum and a known upper bound. Further search within this restricted region is continued until the desired solution is obtained. The final result yields the complete feedback loop engineering design information for the optimal control.

The method is illustrated with a solution of a third order system which represents the automatic control of longitudinal spacing between two moving vehicles with acceleration constraints.

SOLUTION OF A LINEAR REGULATOR PROBLEM WITH QUADRATIC PERFORMANCE INDEX AND STATE VARIABLE CONSTRAINTS

By

Henry Stanley Mika

A THESIS

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

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering and Systems Science

1970

G- 66615
1-27-71

PLEASE NOTE:

Some pages have indistinct print. Filmed as received.

UNIVERSITY MICROFILMS.

To my wife

Margaret

ACKNOWLEDGEMENTS

In looking back at the course of events leading to the publication of this thesis, the author finds himself indebted to Dr. John B. Kreer, Chairman of the Guidance Committee and thesis advisor. The successful completion of this work is in large part the result of many stimulating discussions with him where his insight and guidance proved invaluable. The author will always recall this period with some nostalgia and a great sense of gratitude.

The author wishes particularily to acknowledge the constructive comments and suggestions of Dr. Gerald L. Park and Dr. Robert O. Barr both in the preparation of this thesis and during the many occasions for personal discussion. Special thanks are due Dr. Alex C. Bacopoulus for his interest in this work and for his probing questions which the author found both challenging and inspiring. Finally, appreciation is expressed to Dr. Herman E. Koenig who, despite the pressure of many activities, made himself available whenever needed.

The author wishes to express his gratitude to the Division of Engineering Research, Michigan State University for the financial support of this research.

It has been said that no man stands alone. Never has this been more true than in this instance. Many times during

the period of study and research the author's wife, Margaret, has had to make personal sacrifices and share many near-traumatic experiences. Her understanding patience and encouragement have made this possible, and the author can only repay with his love and gratitude.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER I

INTRODUCTION

The last ten or fifteen years have produced significant advances in the field of optimal control. However, despite the current advanced status of the theory, a very limited number of engineering applications have been made. For example, one of the simplest and most useful optimization criteria is the quadratic performance index which leads directly to determination of stable closed loop systems, eliminating much of the "cut and try" effort associated with conventional servomechanism design. In the case of the linear system with the control $u(t)$ unconstrained, the general solution is well known. However, in contrast to the general solution, if the engineering solution is defined as the solution which yields specific design values for a stable feedback control and satisfies all state variable constraints, then to the author's knowledge there exists no direct method to obtain such a solution. In the method described in this thesis, the general solution is used as a necessary condition to obtain the engineering solution for a linear time-invariant system.

In general mathematical form, known as the Problem of Bolza, the problem can be stated as follows:
Given the dynamic system

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

and some performance index

$$J = \varphi(t_f) + \int_{t_o}^{t_f} L[x(t),u(t),t]dt,$$

find the control $u^*(t)$, $t_o \le t \le t_f$, which minimizes (or maximizes) the functional J. When the terminal "cost" $\varphi(t_f)$ is not a consideration, i.e., when

$$J = \int_{t_o}^{t_f} L[x(t),u(t),t]dt$$

it is known as the Problem of Lagrange.

To obtain $u^*(t)$, in general, it is necessary to solve a two-point boundary value problem. In a few instances, it has been possible to obtain analytical solutions; but in most cases, particularily where constraints are imposed on the control $u(t)$ or on the state $x(t)$, the only practical method of solution is with the use of numerical techniques and a digital computer.

The choice of the form of the performance index J affects the complexity of the solution to the two-point boundary value problem as well as the complexity of the physical implementation of the desired control $u^*(t)$. The choice may be obvious and leave no room for flexibility as, for example, the engineering specification that the system perform its function in minimum time. However, if some flexibility of choice is permitted, motivated perhaps by cost of physical implementation, then the form of J is made on the basis of judgement, experience, physical insight and mathematical tractability.

The complicated and sophisticated hardware often needed for the implementation of an optimal controller may very well

be the reason why few actual applications of optimal controls currently exist. One has only to consider the complicated feedback elements required for a minimum-time optimal control for a third order system. The switching surface is represented by an extremely complex mathematical function so that the control or feedback elements may represent the major cost of implementing the entire system.

In the case of linear systems

$$\dot{x} = Ax + Bu,$$

the quadratic performance index

$$J = \tfrac{1}{2} \int_{t_o}^{t_f} [x^T Q x + u^T R u] dt$$

is an example of a functional which possesses many desirable engineering features. It is mathematically convenient and leads to a linear control law. In addition, it possesses the intuitively appealing feature of "least-square-error" minimization used in many areas of scientific and engineering analysis and application. Specifically, it penalizes the system for large excursions from some desired state trajectory while minimizing the control "energy".

There are other, less obvious, advantages to the use of a quadratic performance index which mitigate against the use of other performance criteria. For example, a smooth control $u^*(t)$ is assured as contrasted to a "bang-bang" type, and the physical implementation of the control $u^*(t)$ can be relatively simple and inexpensive.

There is a large class of problems where a smooth control is of paramount importance. For example, in the field of transportation, including automobiles, aircraft, trains, etc., where human comfort and performance are important considerations, it is obvious that a "bang-bang" control of vehicle acceleration would be highly undesirable.

The general solution for the optimal control of a linear system with quadratic performance index and $u(t)$ not constrained is well known and is given by

$$u^*(t) = -R^{-1}B^T K(t)x(t).$$

The matrix K determines the feedback coefficients and is related to the A, B, Q and R matrices through the nonlinear matrix Riccati equation. Since the K matrix contains the feedback gain information, it controls the behavior of the resulting closed loop system. However, since this matrix depends on the weighting matrices Q and R, it follows that the system trajectories in state space are a function of Q and R. Thus, when the designer chooses a specific performance index, he also determines the system behavior in state space.

The relationship between the Q and R matrices and the system trajectories is not readily apparent, and the following trial and error design procedure is typical:

1) assume initial Q and R matrices;
2) solve the matrix Riccati equation;
3) simulate the system;
4) if the system performance is not satisfactory, repeat the process.

This procedure is far from satisfactory because it requires repeated solution of the matrix Riccati equation and there is no systematic scheme for correcting Q and R.

The solution of the matrix Riccati equation may in itself be a very difficult computational problem. It requires the solution of $n(n+1)/2$ simultaneous nonlinear equations, either in the algebraic or in the differential form. In general, the matrix Riccati is a first order differential equation which, when solved backward in time, yields a limiting value for K which is a constant matrix equivalent to the solution obtained for the algebraic or steady state matrix Riccati equation. Satisfactory computational methods for systems of moderately high order do not appear to be available.

The recognition of the weighting matrices Q and R as "design parameters" has led to the postulation of the inverse problem: given K, what are Q and R? On the surface, this seems to be a trivial engineering problem -- since knowing K implies knowing the desired feedback gains which are the primary engineering design results. This has been a major shortcoming of the published results in this area: dependence on a-priori knowledge of either K or closed loop eigenvalues. In addition, solution of the matrix Riccati equation is necessary, sometimes within an iterative loop. In summary, a practical engineering solution has not been available.

In the following chapters a solution is presented to the problem:

Given the linear, time-invariant system with scalar control

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

and a quadratic performance index

$$J = \tfrac{1}{2} \int_0^\infty [x^T Q x + r u^2] dt \ ,$$

find the constant matrix K which minimizes J, defines an optimal control $u^*(t)$ such that no state variables $x_i(t)$ exceed their constraints, and does not depend on the solution of the matrix Riccati equation.

The result is in the form of a computational procedure utilizing both a digital and a hybrid configuration, and the discussion is divided as follows:

Chapter II reviews the existing theoretical background with specific emphasis on the inverse problem;

Chapter III develops the theoretical foundation for the algorithm used to obtain the solution;

Chapter IV describes in detail the computer implementation of the algorithm;

Chapter V shows typical results for a third order system based on the problem of automatic longitudinal spacing between two vehicles;

Chapter VI summarizes the results and suggests future extensions to this work.

All of Chapters III and IV are devoted to a detailed discussion of the original contributions; and any material which is not original is explicitly referenced. Specifically, 1) a Minimum Cost Algorithm is developed which locates the arbitrarily small

neighborhood of the global quadratic cost minimum under the constraints that the Q and K matrices be positive definite and the control is asymptotically stable; 2) a Constraint Function Algorithm is developed which establishes an upper bound on the minimum quadratic cost such that all state variable constraints are satisfied and significantly reduces the search region which contains the K matrix corresponding to this minimum; 3) a simple K matrix algorithm is developed, an integral part of the Minimum Cost and Constraint Function Algorithms, which eliminates the need to solve the $n(n+1)/2$ Riccati equations.

CHAPTER II

REVIEW OF EXISTING CONTRIBUTIONS BASIC TO SOLUTION OF THE PROBLEM

The optimal control solution to the linear regulator problem with quadratic performance index has been known since 1960 when Kalman published his results. In the past seven or eight years recognition has been given the inverse problem, i.e., how to find the weighting matrices in the quadratic performance index given an optimal control law. This chapter reviews these results and other background material pertinent to the new results to be discussed in later chapters. Included in this chapter are brief discussions on topics of controllability and observability, state and control variable constraints and sensitivity analysis.

In general, a linear time-invariant dynamic system may be represented by the state equation

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

and an output equation

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

where x is the $n\times 1$ state vector, u is the $m\times 1$, $m \leq n$, control vector and y is the $r\times 1$ output vector. A, B, C and D are constant and conformable matrices.

The optimal control $u^*(t)$, $t_o \leq t \leq t_1$, is one that minimizes the quadratic performance index

$$J = \tfrac{1}{2} \int_{t_o}^{t_1} [x^T Q x + u^T R u] dt \qquad (3)$$

where Q is a state-variable weighting matrix which is either positive definite or positive semidefinite; and R is the control variable weighting matrix which is positive definite.

The search for this minimum, if it exists, may be implemented in several ways, for example, by the Maximum Principle of Pontryagin [PON-1] which is an efficient generalization of the necessary conditions for a local minimum.

2.1 Constraints.

The solution of an optimal control problem inherently involves constraints. For example, the solution of equation (3) depends on the constraints imposed by equation (1). As in ordinary calculus, equality constraints of the form

$$f[x(t),u(t),t] - \dot{x} = 0$$

are readily adjoined to J by a Lagrange multiplier or co-state vector λ, so that the problem is one of minimizing

$$J = \tfrac{1}{2} \int_{t_o}^{t_1} \{x^T Q x + u^T R u + \lambda^T [Ax+Bu-\dot{x}]\} dt \ .$$

A more complex problem results when inequality constraints are imposed on the control vector

$$U_i(\max) \geq u_i(t) \geq U_i(\min); \ i = 1,2,.....,m \ .$$

The usual procedure is to change the inequality constraint into an equality constraint as first suggested by Valentine [VAL-1] to form a function

$$v_i(u_i) = [U_i(\max) - u_i(t)][u_i(t) - U_i(\min)] - g_i^2(t) = 0$$

and again adjoining with the Lagrange multiplier Ψ so that

$$J = \tfrac{1}{2}\int_{t_o}^{t_1} \{x^T Q x + u^T R u + \lambda^T[Ax+Bu-\dot{x}] - \sum_{i=1}^{m} \Psi_i v_i(u_i)\}dt \ .$$

The minus sign precedes the last term because $v_i(u_i)$ is negative when constraints are violated and this must reflect an added cost.

In the case of state variable inequality constraints, augmentation of the state vector is implied. Several approaches [KEL-1], [BER-1] of which the following [McG-1] is representative have been successfully applied.

Define $\dot{x}_{n+1} = f_{n+1} = \sum_{i=1}^{\ell} [h_i(x,t)]^2 H(h_i)$

where $[h_i(x,t)]^2 = [x_i(\max) - x_i(t)][x_i(t) - x_i(\min)]$

$$H(h_i) = \begin{cases} 0 & \text{if} \quad h_i(x,t) \geq 0 \\ K_i & \text{if} \quad h_i(x,t) < 0 \end{cases}$$

$\ell \leq n$ is the set defining the constrained state variables

$$x_{n+1}(t_o) = 0$$

$$x_{n+1}(t_1) = 0 \ .$$

Using again the method of Lagrange multipliers results in

$$J = \tfrac{1}{2}\int_{t_o}^{t_1} \{x^T Qx + u^T Ru + \lambda^T (Ax+Bu-\dot{x}) - \sum_{i=1}^{m} \Psi_i v_i(u_i)$$

$$+ \lambda_{n+1}(f_{n+1} - \dot{x}_{n+1})\}dt \ . \qquad (4)$$

Thus, it is possible to formulate a minimization problem with constraints, and computer techniques using the gradient method have been developed that converge to a solution [SAG-1], [WAN-1]. It will be shown in the development of this investigation that in the case of the quadratic performance index, an optimal solution with control and state variable constraints is obtained without the need to solve the complex two-point boundary value problem implied by equation (4).

2.2 Controllability and Observability.

The concepts of controllability and observability were first introduced by Kalman [KAL-1] who showed that they appear as necessary and sometimes sufficient conditions for existence of solutions in control problems, particularily those involving multiple inputs and multiple outputs. One of the best "state of the art" papers is that of Kreindler and Sarachik [KRE-1] and is the source for the following definitions. Discussion will be limited to linear time invariant systems defined by equations (1) and (2).

Definition 1: A plant is said to be completely state-controllable if for any t_o each initial state $x(t_o)$ can be transferred to any final state $x(t_f)$ in a finite time $t_f \geq t$.

It can be shown that a linear time-invariant plant is completely state-controllable if and only if the $n \times mn$ matrix

$$E = [B \mid AB \mid A^2B \mid \text{-------} \mid A^{n-1}B]$$

satisfies the relation

$$\text{rank } E = n \ .$$

One important application of complete state-controllability is that it can be shown [WON-1] that if the system (1) is completely controllable, then there exists a nonsingular matrix H such that

$$x = Hy$$

$$\bar{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 \cdots & 0 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & 1 \\ -a_1 & -a_2 & -a_3 & \cdots & -a_n \end{bmatrix}$$

$$\bar{B} = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}$$

$$y = \bar{A}y + \bar{B}u \ .$$

That is, the system (1) can always be transferred into the phase variable canonical form.

Definition 2: An unforced plant is said to be completely observable on $[t_o,t_f]$ if for given t_o and t_f, every state $x(t_o)$ in X can be determined from the knowledge of $y(t)$ on $[t_o,t_f]$. If the above is true for every t_o and some finite $t_f > t_o$ then the plant is said simply to be completely observable. If the above is true for every t_o and every $t_f \geq t_o$, the plant is said

to be totally observable.

As in the case of controllability, there exists a necessary and sufficient condition for total observability. A linear time-invariant plant is totally observable if and only if the $n \times nm$ matrix

$$F = [C^T \vdots A^T C^T \vdots \text{-------} \vdots (A^T)^{n-1} C^T]$$

satisfies the relation

$$\text{rank } F = n \ .$$

In the specific problem to be discussed hereafter, it will be assumed that C is the unit matrix, and, therefore (1) - (2) is a totally observable system.

2.3 Analytic Solution for Regulator Problem.

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

represent the state equation for a regulator, i.e., a control $u(t)$ is desired such that the state vector $x(t)$ is returned to zero.

Let
$$J = \tfrac{1}{2} \int_0^\infty [x^T Q x + u^T R u] dt \ . \tag{5}$$

If all the matrices are constant, Q and R positive definite, $u(t)$ unconstrained, and the system completely controllable, then it is well known [ATH-1] that a unique optimal control $u^*(t)$ exists and is given by

$$u^*(t) = -R^{-1} B^T K x(t). \tag{6}$$

The constant positive definite symmetric matrix K satisfies

the algebraic matrix Riccati equation

$$KA + A^T K - KBR^{-1}B^T K + Q = 0 \ . \tag{7}$$

Since equation (7) is a system of quadratic algebraic equations, it does not, in general, have a unique solution. Kalman [KAL-2], however, has shown that if there exists a matrix H such that

$$Q = H^T H$$

then total observability of the pair $[A,H]$ assures the existence of a unique positive definite solution. In addition, this condition relaxes the requirement on Q in that it need be only positive semidefinite.

The solution of the matrix Riccati equation may be a formidable task since it requires the solution of a system of $n(n+1)/2$ simultaneous nonlinear equations. Computational stability and computer capacity [BLA-1] set an upper bound on the order of the system which can be handled. A recent technique [KLE-1] indicates that the algebraic matrix Riccati solution can be obtained for $n = 10$. However, it may be more efficient computationally to solve the matrix Riccati equation in differential form by using established Runge-Kutta [ATH-2] procedures. This follows from the fact that the algebraic Riccati equation represents the steady state solution of

$$\dot{K}(t) = -K(t)A - A^T K(t) + K(t)BR^{-1}B^T K(t) - Q \tag{8}$$

with

$$\lim_{t_o \to -\infty} K(t_o) = K.$$

An algorithm developed in Chapter III for a scalar control $u(t)$ eliminates the need to solve the matrix Riccati equation (7) or (8) to obtain the K matrix, and the computational limitation on the order of the system is relaxed.

2.4 The Inverse Problem.

It has been recognized for sometime that the Q and R weighting matrices in the quadratic performance index are really engineering design parameters. Since no generality is sacrificed by assuming R as the unit matrix, the design problem is reduced to finding an acceptable Q matrix. This is the so-called inverse problem.

The general inverse problem was first posed by Kalman [KAL-2] in the following fashion:

Let $$J = \lim_{t_1 \to \infty} \int_o^{t_1} L[x(t,t_o),u(t)]dt$$

then, given a completely-controllable constant linear plant and constant control law, determine all loss functions L of the quadratic form such that the control law minimizes J.

Kalman then proceeded to derive the necessary and sufficient conditions for the solution of this problem in terms of a frequency domain characterization. Let

$Q = H^T H$ be a non-negative definite matrix

K be a symmetric positive definite matrix satisfying the steady state Riccati equation

$$-KA - A^TK = H^TH - KBB^TK$$

$R = [1]$, i.e, a scalar control

$\varphi(s) = (sI - A)^{-1}$

$\Psi(s) = |sI - A|$

$G = A - BB^TK$

$\Psi_G(s) = |sI - G|$.

Theorem (Kalman):

Consider a completely controllable plant and the associated variational problem with a quadratic performance index such that the pair $[A,H]$ is totally observable. Let KB be a fixed control law. Then a necessary and sufficient condition for KB to be an optimal control law is that KB be a stable control law and that the condition

$$|1 + B^TK\varphi(i\omega)B|^2 = 1 + \|H\varphi(i\omega)B\|^2 \tag{9}$$

hold for all real ω.

This is an important fundamental result which establishes a relationship between the time domain of the state variable approach and the frequency domain of conventional feedback theory. In fact, the term $1 + B^TK\varphi(i\omega)B$ represents the return difference in feedback theory.

Theorem (Kalman):

A control law KB is optimal if and only if

a) $|sI - G|$ satisfies the Routh-Hurwitz conditions;

b) $\Psi(\omega^2) = |\Psi_G(i\omega)|^2 - |\Psi(i\omega)|^2$.

This theorem implies that it is possible to characterize optimality in terms of closed loop and open loop characteristic

equations. Unfortunately, the condition b) is not known in general form for application to any n^{th} order system.

These results are fundamental but are not easily applied. Equation (9) implies that an H matrix, if it exists, can be found if K is known. Kalman proposes a solution using spectral factorization, but this implies the non-uniqueness of H and, therefore, non-uniqueness of Q. Furthermore, the actual engineering design solution is the K matrix which determines the feedback coefficients for an optimal control system; and, thus, to obtain H it is pre-supposed that the design information is already known. For these reasons there appears to be little or no evidence that these results have been applied to engineering problems.

Three years after the publication of these results, Athans, et. al. [ATH-3] were proposing a "cut and try" procedure based on the work of Bryson and Ho to obtain acceptable Q and R matrices. Initial values were chosen on the basis of

$$q_{ii}^{-1} = \max x_{ii}^{2}(t)$$

$$r_{jj}^{-1} = \max u_{jj}^{2}(t)$$

that is, the diagonal entries of the Q and R matrices were an inverse function of the maximum magnitude of the variable to be weighted. Again, an a-priori knowledge of system behavior is implied. If the initial "guess" was not satisfactory then adjustments were made in a "cut and try" fashion. Of course, each time Q or R is adjusted a new K matrix must be determined from the matrix Riccati equation. An essentially

"cut and try" procedure similar to the above is still common practice in engineering work.

Rekasius and Hsia [REK-1] attempted a more formal approach which was motivated by the Problem of Letov [LET-1]. In this problem the control $u(t)$ is assumed a scalar with the constraint $|u(t)| \leq 1$ and

$$J = \tfrac{1}{2} \int_0^\infty (x^T Q x + r u^2) dt \ .$$

It was found that under the condition, $\alpha < 0$,

$$\alpha B^T K = B^T K[A - (BB^T K)/r] \tag{10}$$

the control

$$u(x) = \begin{cases} u^*(x) = -(B^T K x)/r & ; \ |u| \leq 1 \\ \text{Sgn } u^*(x) & ; \ |u| \geq 1 \end{cases}$$

is optimal.

Assuming a phase variable canonical form for the A and B matrices, they defined

$$-(KB)/r = k_n \begin{bmatrix} k_1/k_n \\ k_2/k_n \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}. \tag{11}$$

Substituting A, B and (11) into equation (10) results in $(n-1)$ equations

$$\left(\frac{k_1}{k_n}\right)\left(\frac{k_{n-1}}{k_n}\right) - a_n\left(\frac{k_1}{k_n}\right) + a_1 = 0$$

$$\left(\frac{k_2}{k_n}\right)\left(\frac{k_{n-1}}{k_n}\right) - a_n\left(\frac{k_{n-1}}{k_n}\right) + a_2 = 0$$

$$\vdots$$

$$\left(\frac{k_{n-1}}{k_n}\right)^2 - a_n\left(\frac{k_{n-1}}{k_n}\right) - \left(\frac{k_{n-2}}{k_n}\right) + a_{n-1} = 0$$

For an optimal solution to exist, these equations must be independent and have at least one set of real solutions such that $u(x)$ is a stable control law. Once the ratios $k_i/k_n = \gamma_i$, $i = 1,2,.....,n-1$, are known, substitution in equation (11) yields

$$KB = k_{nn}\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ 1 \end{bmatrix}$$

where k_{nn} is the bottom-row, last column element of the K matrix, and once known the optimal control law follows as well as the k_{ni}, $i = 1,2,...,n-1$, elements of the K matrix. As in Kalman's work, Rekasius and Hsia assumed a k_{nn} as well as the rest of the elements that determine the entire K matrix. They then solved for the control law as well as the Q matrix by substitution in the steady state Riccati equation.

Tyler and Tuteur [TYL-1] approached the inverse problem by assuming Q to be a diagonal matrix, R the unit matrix and

obtaining a relationship in the frequency domain which was a function of Q. The procedure involves trial and error adjustments of the Q elements until satisfactory response is achieved. Once Q is established in this fashion, the K matrix is obtained by solving the matrix Riccati equation.

Assuming $\dot{x} = Ax + Bu$

$$y = Cx$$

$$J = \tfrac{1}{2} \int_0^\infty (x^T C^T Q C x + u^T u) dt$$

and using the canonical equations

$$\begin{bmatrix} \dot{x} \\ \cdots \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} A & -BB^T \\ \cdots & \cdots \\ -C^T Q C & -A^T \end{bmatrix} \begin{bmatrix} x \\ \cdots \\ \lambda \end{bmatrix} = F_c \begin{bmatrix} x \\ \cdots \\ \lambda \end{bmatrix}$$

the characteristic equation of the optimal system is defined by

$$|sI - A + BB^T K| = 0 = \prod_{i=1}^{n} (s - \alpha_i) . \qquad (12)$$

However, it is well known that the $2n$ eigenvalues of the F_c matrix consist of the n eigenvalues equivalent to the roots in equation (12) plus n eigenvalues which are the mirror image about the imaginary axis of the s-plane. Therefore,

$$|sI - F_c| = \prod_{i=1}^{n} (s - \alpha_i)(s + \alpha_i)$$

Pre-multiplying F_c yields the following

$$\begin{bmatrix} (sI - A)^{-1} & 0 \\ -(C^TQC)(sI - A)^{-1} & I \end{bmatrix} \begin{bmatrix} (sI - A) & BB^T \\ C^TQC & (sI + A^T) \end{bmatrix}$$

$$= \begin{bmatrix} I & (sI - A)^{-1}BB^T \\ 0 & -C^TQC(sI - A)^{-1}BB^T + (sI + A^T) \end{bmatrix}$$

and

$$|sI - F_c| = |sI - A| \; |(sI + A^T) - C^TQC(sI - A)^{-1}BB^T| = 0 \; .$$

By further manipulation

$$|sI - A| \; |sI + A^T| + (-1)^n \sum_{i=1}^{2(n-1)} k_i N_i(s) N_i(-s) = 0$$

where $N_i(s)$ is a polynomial in s, each k_i consists either of a q_{ii} element or a product of q_{ii} elements.

This rather formidable relationship of $(2n)^{th}$ degree containing 2^n terms must be satisfied by adjusting the q_{ii} elements via root locus techniques until the desired system response is obtained. This leads to a solution for the entire Q matrix which is then used in the matrix Riccati equation to obtain the K matrix. The difficulty of using root locus techniques with a high order system is alone a challenging task.

A technique proposed by Chen and Shen [CHE-1] appears to overcome some of the difficulties associated with the Tyler and Tuteur method. Given a quadratic performance index and specified closed loop eigenvalues, a sensitivity relationship between the eigenvalues and the Q matrix is used to solve iteratively for the desired Q matrix.

Letting $G = A - BB^T K$

it follows that

$$Gu_i = \lambda_i u_i;\ i = 1,2,\ldots,n$$

where λ_i and u_i are the distinct eigenvalues and eigenvectors, respectively. This leads to the Jacobi sensitivity formula [Van-1]

$$d\lambda_i = \langle v_i, dGu_i \rangle \tag{13}$$

where v_i is the reciprocal basis of u_i, i.e., [MOR-1] if

$$U = [u_1 \mid u_2 \mid \ldots \mid u_n]$$

then $$V = [U^T]^{-1} = [v_1 \mid v_2 \mid \ldots \mid v_n] .$$

Since $$KA + A^T K - KBB^T K = -C^T QC$$

then $$dKA + A^T dK - dKBB^T K - KBB^T dK = -C^T dQC$$

or $$dKG + G^T dK + C^T dQC = 0 \tag{14}$$

and $$dG = -BB^T dK \tag{15}$$

By combining equations (13), (14) and (15) it can be shown that

$$\begin{aligned} d\lambda_i &= -\mathrm{tr}(u_i v_i BB^T dK) \\ dKG + G^T dK &= -C^T dQC \end{aligned} \tag{16}$$

Assuming Q is diagonal and given a set of desired eigenvalues, equation (16) and the steady state matrix Riccati equation are used within an iterative loop to obtain a Q matrix.

It is clear that this method suffers from two disadvantages: 1) a desired set of eigenvalues is not always available to the

designer; and 2) an iterative scheme that involves solving the matrix Riccati equation, particularily when the order of the system is moderately large, will be either extremely time-consuming or computationally unstable.

In summary, although the classic inverse problem can be solved, the engineering designer requires a much better tool to do his job.

2.5 Sensitivity Analysis.

In the proposed procedure to obtain a solution to the linear regulator problem with a quadratic performance index and constrained state variables, the problem is essentially reduced to optimizing a n-parameter system using partial derivatives of the form $\frac{\partial x_i}{\partial p_j}$. It is common practice [TOM-1], [PER-1] to define this type of derivative as the sensitivity of state variable x_i to the parameter p_j.

Following the usual method of development, let V be the $n \times n$ matrix whose elements are

$$v_{ij}(t) = \frac{\partial x_i(t)}{\partial p_j}$$

Differentiating with respect to time

$$\dot{v}_{ij}(t) = \frac{d}{dt} \frac{\partial x_i(t)}{\partial p_j}$$

where $x_i = f_i(\tilde{p},t)$.

Before further expansion it is first necessary to show that matters are considerably simplified if the following assumptions are made:

1) $\dfrac{\partial^2 x_i(t)}{\partial t \partial p_j}$ is continuous

2) the system is time invariant, i.e., p is not an explicit function of time.

Then, $$\frac{d\tilde{p}}{dt}\left[\frac{\partial x_i(t)}{\partial p_j}\right] = \frac{\partial^2 x_i(t)}{\partial t \partial p_j} + \frac{\partial^2 x(t)}{\partial \tilde{p} \partial p_j}\frac{d}{dt} = \frac{\partial^2 x(t)}{\partial t \partial p_j}$$

$$\frac{\partial}{\partial p_j}\left[\frac{dx_i(t)}{dt}\right] = \frac{\partial}{\partial p_j}\left[\frac{\partial x_i(t)}{\partial t} + \frac{\partial x_i(t)}{\partial \tilde{p}}\frac{d\tilde{p}}{dt}\right] = \frac{\partial^2 x_i(t)}{\partial p_j \partial t}$$

Since $\dfrac{\partial^2 x_i(t)}{\partial t \partial p_j}$ is continuous, then

$$\frac{\partial^2 x_i(t)}{\partial t \partial p_j} = \frac{\partial^2 x_i(t)}{\partial p_j \partial t}$$

and $$\dot{v}_{ij}(t) = \frac{\partial}{\partial p_j}\left[\frac{dx_i(t)}{dt}\right] = \frac{\partial \dot{x}_i(t)}{\partial p_j}$$

Since $$\dot{x} = Gx$$

and $$\dot{x}_i(t) = g_i(\tilde{x}, \tilde{p}, t)$$

$$\dot{v}_{ij}(t) = \frac{\partial}{\partial p_j}\left[g_i(\tilde{x}, \tilde{p}, t)\right]$$

$$= \frac{\partial g_i}{\partial x_1(t)}\frac{\partial x_1(t)}{\partial p_j} + \frac{\partial g_i}{\partial x_2}\frac{\partial x_2}{\partial p_j} + \cdots\cdots\cdots + \frac{\partial g_i}{\partial x_n(t)}\frac{\partial x_n(t)}{\partial p_j} + \frac{\partial g_i}{\partial p_j}$$

$$= \sum_{r=1}^{n}\left[\frac{\partial g_i}{\partial x_r(t)}\right]\frac{\partial x_r(t)}{\partial p_j} + \frac{\partial g_i}{\partial p_j}$$

$$= \sum_{r=1}^{n}\left[\frac{\partial g_i}{\partial x_r(t)}\right]v_{rj}(t) + \frac{\partial g_i}{\partial p_j} \qquad (17)$$

In matrix form for an n^{th} order system

$$\begin{bmatrix} \dot{v}_{11} & \dot{v}_{12} & \cdots & \dot{v}_{1n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \dot{v}_{n1} & \dot{v}_{n2} & \cdots & \dot{v}_{nn} \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1(t)} & \cdots & \frac{\partial g_1}{\partial x_n(t)} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \frac{\partial g_n}{\partial x_1(t)} & & \frac{\partial g_n}{\partial x_n(t)} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix}$$

$$+ \begin{bmatrix} \frac{\partial g_1}{\partial p_1} & \cdots & \frac{\partial g_1}{\partial p_n} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \frac{\partial g_n}{\partial p_1} & & \frac{\partial g_n}{\partial p_n} \end{bmatrix} \tag{18}$$

Since the matrix G can always be transformed into the phase variable form

$$\begin{aligned} g_1(t) &= x_2(t) \\ g_2(t) &= x_3(t) \\ &\cdot \quad\quad \cdot \\ &\cdot \quad\quad \cdot \\ g_{n-1}(t) &= x_n(t) \\ g_n(t) &= -p_1x_1(t) - p_2x_2(t) - \cdots\cdots - p_nx_n(t) \end{aligned} \tag{19}$$

substituting into equation (18) and then decomposing into $j = 1,2,\ldots,n$ first order differential equations yields

$$\begin{bmatrix} \dot{v}_{1j}(t) \\ \dot{v}_{2j}(t) \\ \cdot \\ \cdot \\ \cdot \\ \dot{v}_{nj}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots\cdots & 0 \\ \cdot & 0 & 1 & \cdots\cdots & 0 \\ \cdot & \cdot & 0 & 1 \cdots\cdots & \cdot \\ \cdot & \cdot & \cdot & 0 & \cdot \\ \cdot & & & & 1 \\ -p_1 & -p_2 & -p_3 & & -p_n \end{bmatrix} \begin{bmatrix} v_{1j}(t) \\ v_{2j}(t) \\ \cdot \\ \cdot \\ \cdot \\ v_{nj}(t) \end{bmatrix} - \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_j(t) \end{bmatrix} \tag{20}$$

or $$\dot{V}_j = GV_j + Bx_j(t)$$

where V_j is the j^{th} column of the $n \times n$ matrix V, and

$$B = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ -1 \end{bmatrix}$$

Since $V_j(t_o) = 0$ for normal physical systems, [KOE-1]

$$V_j(t_i) = \epsilon^{Gt_i} \int_{t_o}^{t_i} \epsilon^{-G\tau} B x_j(\tau) d\tau \qquad (22)$$

Equation (22) will be basic to some of the development in Chapter III.

CHAPTER III

DEVELOPMENT OF A SOLUTION

In this chapter the new developments pertinent to the solution of the linear regulator problem with state variable constraints are presented. A simple algebraic algorithm is developed for a large class of problems to obtain the gain matrix K, associated with the matrix Riccati equation, from an n-parameter optimization solution. Thus it is not necessary to solve the usual $n(n+1)/2$ simultaneous nonlinear equations. The parameter optimization solution is based on a quadratic cost minimization algorithm that follows from the K matrix algorithm and a special constraint function developed for this purpose. Both of these algorithms are described in detail.

3.1 Canonical Transformation.

Given the linear dynamic system

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

where x is the $n\times 1$ state vector, u is the $m\times 1$ control vector, $m \leq n$, A is a constant $n\times n$ matrix and B is a constant $n\times m$ matrix, then for a physically meaningful problem an originally unstable system can always be stabilized by a properly chosen controller [KAL-2]. For the optimal control designed to minimize the quadratic performance index

$$J = \tfrac{1}{2}\int_0^\infty [x^T Q_x + u^T R u]dt \tag{2}$$

where the constant $n\times n$ matrix Q and the constant $m\times m$ matrix R are positive definite, it has been shown [ATH-1] that the unique optimal control exists and is defined by

$$u^* = -R^{-1}B^T K x \tag{3}$$

where K is a unique positive definite symmetric $n\times n$ matrix satisfying

$$KA + A^T K - KBR^{-1}B^T K + Q = 0 \tag{4}$$

Lemma 1: If the symmetric matrix K is unique then the $n\times n$ matrix Q is unique.

Proof: $$Q = -KA - A^T K + KBR^{-1}B^T K$$

Since for a given system and performance index A, B, R are unique, it follows that Q is unique.

Q.E.D.

It is known that [KAL-2] any completely controllable system with a scalar control $u(t)$

$$\dot{y} = Fy + Hu$$

can always be transformed into the phase variable canonical form

$$\begin{bmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots\cdots & 0 \\ 0 & 0 & 1 & & 0 \\ \cdot & \cdot & 0 & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ -a_o & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} u \tag{5}$$

Quite often the phase variable form follows directly from the original differential equation for the system

$$x^{(n)} + a_{n-1}x^{(n-1)} + \cdots\cdots + a_1 x^{(1)} + a_o x^{(0)} = bu \qquad (6)$$

By suitable scaling of the state vector z, i.e.,

$$z_i = x_i/b$$

the form in (5) can be transformed into the more convenient form

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \cdot & 0 & 1 & & \cdot \\ \cdot & \cdot & 0 & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & 1 \\ -a_o & -a_1 & \cdots\cdots & & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ b \end{bmatrix} u \qquad (7)$$

which will be used throughout the sequel. In matrix form (7) becomes

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

3.2 Gain Matrix.

One of the major objectives of this research was to eliminate the need for the usual iterative procedure [ATH-2], [KLE-1] to solve the $n(n+1)/2$ equations represented by the algebraic matrix Riccati equation (4) in order to obtain the gain matrix K. It will be shown that this can be done for the system (7), assuming a quadratic performance index structure.

First, it will be assumed that R is the unit matrix. This is commonly done and does not affect the generality of the

results. In effect, the matrix Q is redefined to be the matrix Q_1 such that it "absorbs" the difference between any other R and $R = I$. From (4)

$$Q = -KA - A^TK + KBR^{-1}B^TK$$

and it follows [KAL-2] that, for a completely controllable system and Q positive definite,

$$J(t_o) = \tfrac{1}{2}\int_{t_o}^{\infty} [x^TQx + u^TRu]dt = \tfrac{1}{2}\, x^T(t_o)Kx(t_o) \tag{8}$$

where K is a unique solution. Since A, B and K are unique, by Lemma 1 it is possible to generate a unique matrix Q_1 with $R = I$ such that

$$Q_1 = -KA - A^TK + KBB^TK \tag{9}$$

$$J(t_o) = \tfrac{1}{2}\int_{t_o}^{\infty} [x^TQ_1x + u^Tu]dt = \tfrac{1}{2}\, x^T(t_o)Kx(t_o) \tag{8a}$$

Since K is the same in both cases, the quadratic performance index in (8a) is equivalent to the one in (8).

The second assumption is that the Q matrix be diagonal. In practice this represents the largest class of problems since it is quite often very difficult to ascribe weighting coefficients to state variable cross products.

K Matrix Algorithm:

Given the completely controllable system (7), R the unit matrix and Q a diagonal (with unknown entries) matrix, then the complete symmetric K matrix may be obtained from the n entries in the n^{th} row of the K matrix.

Proof is by expansion of

$$Q = -KA - A^T K + KBB^T K$$

Taking advantage of the fact that $k_{ij} = k_{ji}$, it can be readily established that

$$q_{11} = 2a_o k_{n,1} + b^2 k_{n,1}^2 \tag{10}$$

$$q_{ii} = -2k_{i-1,i} + 2a_{i-1} k_{n,i} + b^2 k_{n,i}^2 \quad ; \; i = 2,3,....,n \tag{11}$$

$$q_{ij} = -k_{1,j-1} + a_o k_{n,j} + a_{j-1} k_{n,1} + b^2 k_{n,1} k_{n,j}$$

$$j = 2,3,.....,n \tag{12}$$

$$q_{ij} = -k_{i,j-1} - k_{i-1,j} + a_{i-1} k_{n,j} + a_{j-1} k_{n,i} + b^2 k_{n,i} k_{n,j}$$

$$i = 2,3,.....,n$$

$$j = 1,2,.....,n \tag{13}$$

Since Q is diagonal, all $q_{ij} = 0$ when $i \neq j$. Then, beginning with equation (12)

$$k_{1,j-1} = a_o k_{n,j} + a_{j-1} k_{n,1} + b^2 k_{n,1} k_{n,j}$$

$$j = 2,3,.....,n \tag{14}$$

showing that all first row entries in the K matrix can be obtained if the $k_{n,i}$ entries are known, $i = 1,2,.....,n$.

Continuing with equation (13), since all the $k_{i-1,j}$ are known from the previous row computation, a relation for $k_{i,j-1}$ is obtained

$$k_{i,j-1} = -k_{i-1,j} + a_{i-1} k_{n,j} + a_{j-1} k_{n,i} + b^2 k_{n,i} k_{n,j}$$

$$i = 2,3,....,n$$

$$j = 1,2,....,n \tag{15}$$

The originally unknown q_{ii} elements are then computed directly from the knowledge of the K matrix obtained via equations (10) and (11).

For example, with $n = 3$

$$k_{11} = a_o k_{32} + a_1 k_{31} + b^2 k_{31} k_{32}$$

$$k_{12} = a_o k_{33} + a_2 k_{31} + b^2 k_{31} k_{33}$$

$$k_{22} = -k_{31} + a_1 k_{33} + a_2 k_{32} + b^2 k_{32} k_{33}$$

Together with the three elements of the n^{th} row of the K matrix, these define the complete matrix since K is symmetric. And

$$q_{11} = 2a_o k_{31} + b^2 k_{31}^2$$

$$q_{22} = -2k_{12} + 2a_1 k_{32} + b^2 k_{32}^2$$

$$q_{33} = -2k_{32} + 2a_2 k_{33} + b^2 k_{33}^2$$

It is obvious that if $k_{n,i}$, $i = 1,2,\ldots,n$, are available, the entire K matrix is readily determined with simple algebra; and solution of the $n(n+1)/2$ simultaneous nonlinear equations becomes unnecessary. The method of obtaining the $k_{n,i}$ forms the bulk of this research effort.

3.3 Autonomous Form.

Since $$u^* = -R^{-1} B^T K x$$

for $R = 1$, i.e., a scalar control,

$$u^* = -b(k_{n1}x_1 + k_{n2}x_2 + \ldots\ldots + k_{nn}x_n)$$

Thus, it is clear that the optimal control u^* is a function

of only the n^{th} row K matrix elements. Substituting in (1)

$$\dot{x} = (A - BB^T K)x = Gx \tag{16}$$

where

$$G = \begin{bmatrix} 0 & 1 & 0 & 0 \\ . & 0 & 1 & . \\ . & . & 0 & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & 1 \\ -(a_o+b^2k_{n1}) & -(a_1+b^2k_{n2}) & \cdots\cdots\cdots & -(a_{n-1}+b^2k_{nn}) \end{bmatrix} \tag{17}$$

Let $$p_i = a_{i-1} + b^2 k_{ni}$$

$$G = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots\cdots\cdots & 0 \\ . & 0 & 1 & 0 & & . \\ . & . & 0 & 1 & & . \\ . & . & . & 0 & & . \\ . & . & . & . & & . \\ . & . & . & . & & 1 \\ -p_1 & -p_2 & -p_3 & \cdots\cdots\cdots\cdots & & -p_n \end{bmatrix} \tag{18}$$

The problem can now be posed as, given the autonomous system

$$\dot{x} = Gx$$

find the parameters p_i so that the quadratic performance index cost

$$J(t_o) = \tfrac{1}{2} x^T(t_o) K x(t_o)$$

is minimized. In this sense the problem becomes one of parameter optimization; and since the parameters are simply and linearly

related to the k_{ni} elements, once the optimum parameters p_i have been found, the entire K matrix is known by use of the K matrix algorithm.

Constraints on state variables often exist and preclude a simple choice of K which minimizes the cost J. The following theorem is new proof that the cost J is a Lyapunov function and leads to a general cost minimization criterion which can be used to minimize the cost in any subinterval of $[t_o,\infty]$.

Lemma 2: Given the completely controllable linear time invariant system

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

and the cost functional

$$J = \tfrac{1}{2}\int_o^\infty [x^T Qx + u^T Ru]dt$$

where u is not constrained and K is a symmetric positive definite matrix satisfying

$$-KA - A^T K + KBR^{-1}B^T K - Q = 0$$

Q positive semi-definite and R positive definite, then the matrix $KB(R^{-1})^T B^T K$ is positive definite.

Proof:

$u^T Ru$ is positive definite by hypothesis

$u^* = -R^{-1}B^T Kx$ is the optimal control

Substituting,

$$u^{*T} Ru^* = -x^T KB(R^{-1})^T R - R^{-1}B^T Kx$$

$$= x^T KB(R^{-1})^T B^T K x$$

which implies that the matrix $KB(R^{-1})^T B^T K$ is positive definite.

Q.E.D.

Corollary: If $R = 1$, and K is the $n \times n$ matrix and

$$B = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ b \end{bmatrix}$$

then the matrix

$$[m_{ij}] = [k_{in} k_{nj}]$$

is positive definite.

Proof is by direct substitution of R and B into the result of Lemma 2.

Theorem I:

Given the completely controllable linear time invariant system

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

and

$$u = -R^{-1} B^T K x$$

Q and R positive definite and K the symmetric positive definite matrix satisfying the steady state Riccati equation

$$KA + A^T K - KBR^{-1} B^T K + Q = 0,$$

then $x^T K x$ defines a Lyapunov function.

Proof:

$x^T K x$ is positive definite by hypothesis

$x^T K x = 0$ if and only if $x = 0$

$$\dot{x} = Ax - BR^{-1} B^T K x = A - BR^{-1} B^T K x = Gx$$

$$\frac{d}{dt}\ x^T Kx \ = \ x^T Kx + x^T Kx \ = \ (Gx)^T Kx + x^T KGx \ = \ x^T\ B^T K + KG\ x$$

Substituting in the Riccati equation, and since

$$A = G + BR^{-1}B^T K$$

$$K\ G + BR^{-1}B^T K + G + BR^{-1}B^T K\ ^T K - KBR^{-1}B^T K + Q = 0$$

$$KG + G^T K + KB(R^{-1})^T B^T K + Q = 0$$

then $$KG + G^T K = -Q - KB(R^{-1})^T B^T K$$

Since Q is positive definite by hypothesis and $KB(R^{-1})^T B^T K$ is positive definite by Lemma 2, then $KG + G^T K$ is negative definite. Since $\frac{d}{dt}\ x^T Kx = x^T\ KG + G^T K\ x$ then it follows that $\frac{d}{dt}\ x^T Kx$ is negative definite. Since the system is asymptotically stable by hypothesis, then $x^T Kx$ is a Lyapunov function.

Q.E.D.

It follows that given the constant matrix K and t_1, $t_o \leq t_1 \leq \infty$, the cost J_1 for the time interval $[t_o, t_1]$ is

$$J_1 = \tfrac{1}{2}[x^T(t_o)Kx(t_o) - x^T(t_1)Kx(t_1)]$$

3.4 Minimum Cost Algorithm.

The objective is to find the minimum value of the performance index J subject to the following restrictions:

1) the system $\dot{x} = Gx$ is asymptotically stable
2) the weighting matrices Q and R are positive definite
3) the gain matrix K is positive definite
4) the constraints on the state variables are satisfied.

It is convenient to let $t_1 = \infty$, then

$$J = \tfrac{1}{2} x^T(t_o)Kx(t_o)$$

For convenience of notation the argument of the state vector will be dropped, so that

$$J = \tfrac{1}{2} x^T K x \tag{19}$$

It can be readily shown that by expanding (19)

$$J = \tfrac{1}{2}[k_{11}x_1^2 + k_{22}x_2^2 + \ldots + k_{nn}x_n^2 + 2(k_{12}x_1x_2 + \ldots + k_{1n}x_1x_n +$$

$$k_{23}x_2x_3 + \ldots\ldots + k_{n-1,n}x_{n-1}x_n)]$$

where the k_{ij} are the elements of the K matrix.
In more compact form

$$J = \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} k_{ij}x_i x_j \tag{20}$$

It has been shown earlier that given the completely controllable system

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

u a scalar control, J a quadratic cost functional with Q diagonal and positive definite and $R = [1]$, then the positive definite gain matrix which defines the optimal control can be determined from equations (14) and (15). That is, all elements of the K matrix are uniquely determined once the bottom row elements are known. Therefore, J can be found if the vector

$$K_N = \begin{bmatrix} k_{n1} \\ k_{n2} \\ . \\ . \\ k_{nn} \end{bmatrix} \tag{21}$$

is known.

Since $p_i = a_{i-1} + b^2 k_{ni}$; $i = 1,2,.....,n$

a region in p-space of the system $\dot{x} = Gx$ can be readily established which assures asymptotic stability by the use of the Routh Criterion, i.e., forming the array from the characteristic equation

$$\lambda^n + p_n\lambda^{n-1} + ... + p_2\lambda + p_1 = 0$$

1	p_{n-1}	p_{n-3}	.....
p_n	p_{n-2}	p_{n-4}	.....
$\frac{p_n p_{n-1} - p_{n-2}}{p_n}$	.....		
.			
.			

Since a zero entry in the first column is not permitted, there will be, in general, n constraints that must be satisfied such that each element in the first column, starting with the second row, must be positive.

The Routh Criterion is used to set bounds on the minimum values of k_{ni}, $i = 1,2,....,n$. For example, assuming a third order system

$$\lambda^3 + p_3\lambda^2 + p_2\lambda + p_1 = 0$$

$$\begin{matrix} 1 & p_2 \\ p_3 & p_1 \\ \dfrac{p_2p_3 - p_1}{p_3} & 0 \\ p_1 & \end{matrix}$$

then

$$p_3 = a_2 + b^2k_{33} > 0$$

$$k_{33} > -a_2/b^2$$

$$p_1 = a_o - b^2k_{31} > 0$$

$$k_{31} > -a_o/b^2$$

$$p_2p_3 > p_1$$

$$(a_1 + b^2k_{32})(a_2 + b^2k_{33}) > (a_o + b^2k_{31})$$

$$k_{32} > 1/b^2\left[\frac{a_o + b^2k_{31}}{a_2 + b^2k_{33}} - a_1\right]$$

The strict inequalities define open sets so that one must in practice adjust the relationships by choosing $\epsilon_i > 0$, arbitrarily close to zero, such that

$$\min k_{31} = a_o/b^2 + \epsilon_1$$

$$\min k_{32} = 1/b^2\left[\frac{a_o + b^2k_{31}}{a_2 + b^2k_{32}}\right] - \frac{a_1}{b^2} + \epsilon_2$$

$$\min k_{33} = a_2/b^2 + \epsilon_3$$

This is defined as the ϵ-minimal constraint set and may lead to an ϵ-optimal solution, i.e., there exists a neighborhood, however small, so that if $\tilde{p}(\epsilon)$ is the ϵ-optimal parameter solution,

then there exists a $\delta = \|\tilde{p} - \tilde{p}(\epsilon)\| > 0$ such that

$$J(\tilde{p}) < J(\tilde{p}(\epsilon)).$$

Quite often the designer must obtain the minimum cost optimal solution within a given range of eigenvalues as determined by the desired response of the closed loop system. In this case, the maximum and minimum values of the k_{ni} can be found by the following procedure. Given the eigenvalues λ_i for an asymptotically stable system then [TUR-1]

$$p_n = -(\lambda_1 + \lambda_2 + \ldots\ldots + \lambda_n)$$

$$p_{n-1} = (\lambda_1\lambda_2 + \ldots \lambda_1\lambda_n + \lambda_2\lambda_3 + \ldots + \lambda_2\lambda_n + \ldots + \lambda_{n-1}\lambda_n)$$

$$\vdots$$

$$p_1 = (-1)^n(\lambda_1\lambda_2 \ldots\ldots \lambda_n)$$

and $$k_{ni} = (p_i - a_{i-1})/b^2$$

Criteria for the positive definiteness of the Q matrix are readily established in terms of the K_N vector, since

$$q_{ii} > 0 \quad ; \quad i = 1,2,\ldots\ldots,n$$

$$q_{ij} = 0 \quad ; \quad i,j = 1,2,\ldots\ldots,n \qquad i = j$$

then equations (10), (11), (12) and (13) yield a set of equations (22) which is defined as the Q_{ii} Criterion.

$$q_{11} = 2a_o + k_{n1} + b^2 k_{n1}^2 > 0$$

$$q_{22} = -2k_{12} + 2a_1 k_{n2} + b^2 k_{n2}^2 > 0$$

.

.

.

$$q_{n-1,n-1} = -2_{n-2,n-1} + 2a_{n-2} k_{n,n-1} + b^2 k_{n,n-1}^2 > 0$$

$$q_{nn} = -2k_{n-1,n} + 2a_{n-1} k_{n,n} + b^2 k_{n,n}^2 > 0 \qquad (22)$$

The Q_{ii} Criterion has the effect of a filter which passes only the K matrices that satisfy the constraint that the Q matrix be positive definite. However, it is used in the Minimum Cost Algorithm as a means of generating a succession of K_N vectors in a systematic fashion so that the arbitrarily small neighborhood of the absolute minimum of J is located. Computationally, it is convenient to start with the last equation in (22), i.e., start with an initial $k_{n,n}$ and find a satisfactory $k_{n,n-1}$. Having obtained $k_{n,n-1}$, the relationship for $q_{n-1,n-1}$ is used to solve for a satisfactory $k_{n-2,n-1}$. But $k_{n-2,n-1}$ is in general a function of the vector K_N, therefore, $k_{n,n-2}$ is adjusted until $q_{n-1,n-1} > 0$ is satisfied. The process is continued with $q_{n-2,n-2}$, etc. This procedure is flow-charted in Figure 3-1.

3.5 Search Procedure.

The solution for an absolute minimum or ϵ-minimum of J is achieved by the adjustments on the K_N vector, as shown in Figure 3-1, within the region defined by the restrictions and

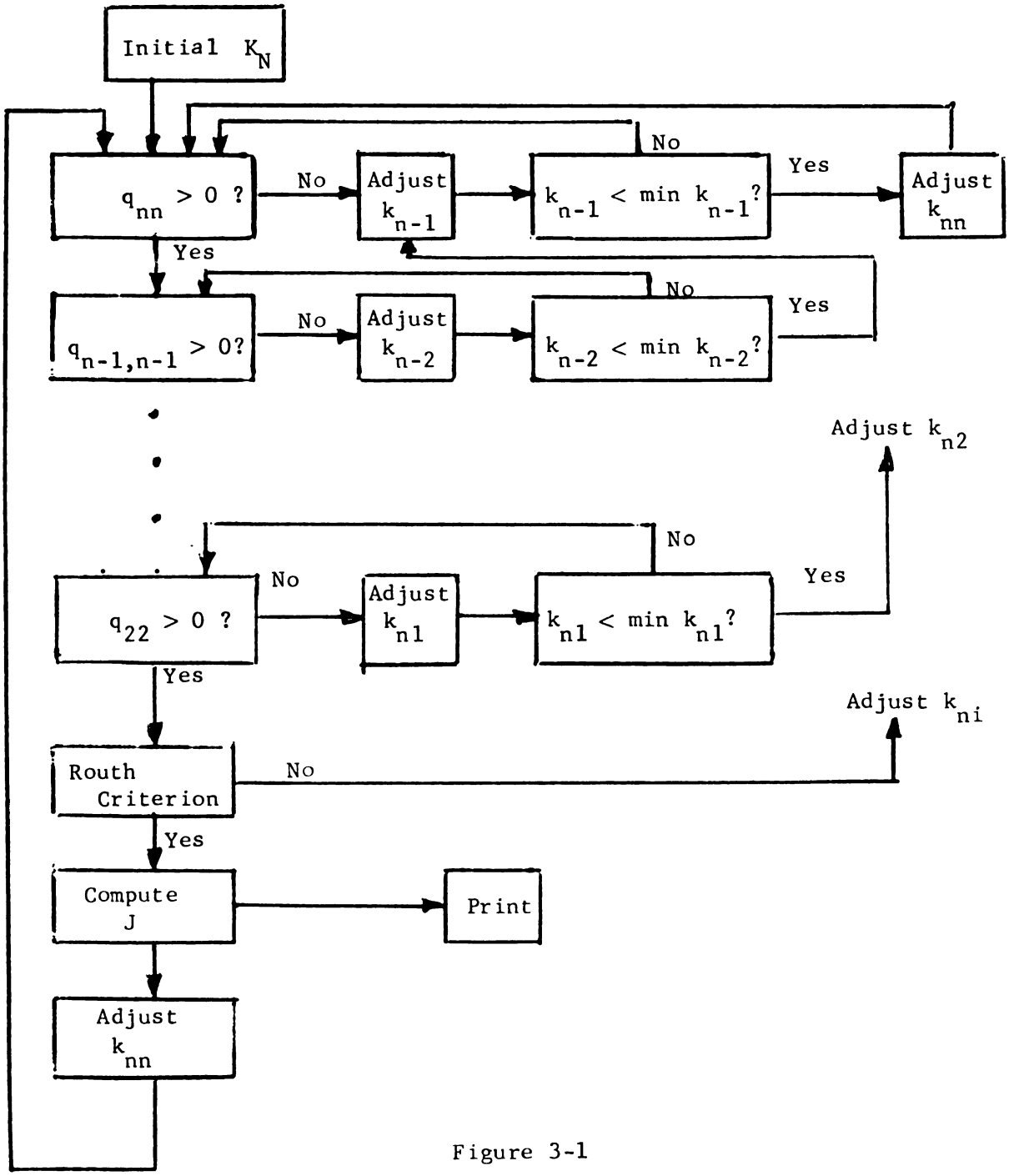


Figure 3-1

Minimum Cost Search Procedure

constraints.

One iterative scheme would be a gradient search using

$$\begin{bmatrix} k_{n1}(i+1) \\ k_{n2}(i+1) \\ \cdot \\ \cdot \\ k_{nn}(i+1) \end{bmatrix} = \begin{bmatrix} k_{n1}(i) \\ k_{n2}(i) \\ \\ \\ k_{nn}(i) \end{bmatrix} - \beta \frac{\nabla J}{\|\nabla J\|} \tag{23}$$

where the constant β determines the step size and

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial k_{n1}} \\ \frac{\partial J}{\partial k_{n2}} \\ \cdot \\ \cdot \\ \frac{\partial J}{\partial k_{nn}} \end{bmatrix} \tag{24}$$

For example, assuming $n = 3$

$$J = 1/2[k_{11}x_1^2 + k_{22}x_2^2 - k_{33}x_3^2 + 2(k_{11}x_1x_2 + k_{13}x_1x_3 + k_{23}x_2x_3)]$$

Using equations (14) and (15)

$$J = 1/2[(a_o k_{32} + a_1 k_{31} + b^2 k_{31}k_{32})x_1^2 + (-k_{31} + a_1 k_{33} + a_2 k_{32} + b^2 k_{32}k_{33})x_2^2 + k_{33}x_3^2 + 2k_{31}x_1x_3 + 2k_{23}x_2x_3 + 2(a_o k_{33} + a_2 k_{31} + b^2 k_{31}k_{33})x_1x_2]$$

$$\frac{\partial J}{\partial k_{31}} = 1/2[(a_1 + b^2 k_{32})x_1^2 - x_2^2] + x_1x_3 + (a_2 + b^2 k_{33})x_1x_2$$

It is useful to know that there are initial conditions $x(t_o)$ such that if the gradient $\nabla J = 0$ exists, it defines a unique vector K_N. Using equations (10) through (15) in equation (20)

$$\nabla J = b^2 M K_N - F \tag{25}$$

$$M = \begin{bmatrix} 0 & x_1^2 & 2x_1x_2 & 2(x_1x_3-x_2^2) & \cdots & 2(x_1x_{n-1}-x_2x_{n-2}) \\ x_1^2 & 0 & x_2^2 & 2x_2x_3 & \cdots & 2(x_2x_{n-1}-x_3x_{n-2}) \\ 2x_1x_2 & x_2^2 & 0 & x_3^2 & \cdots & 2(x_3x_{n-1}-x_4x_{n-2}) \\ \cdot & & & & & \cdot \\ \cdot & & & & & \cdot \\ \cdot & & & & & \cdot \\ \cdot & & & & & 2(x_{n-3}x_{n-1}-x_{n-2}x_{n-2}) \\ \cdot & & & & & \\ \cdot & & & & & 2x_{n-2}x_{n-1} \\ \cdot & & & & & \\ \cdot & & & & & x_{n-1}^2 \\ \cdot & & & & & \\ \cdot & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix}$$

that is, M is a symmetric $n \times n$ matrix with zero entries on the diagonal.

$$F = \begin{bmatrix} f_1(a_i,\tilde{x}) \\ f_2(a_i,\tilde{x}) \\ \cdot \\ \cdot \\ f_n(a_i,\tilde{x}) \end{bmatrix} \quad ; \quad i = 0,1,2,\ldots,n-1$$

If $\nabla J = 0$, then equation (25) becomes

$$MK_N = \frac{1}{b^2} F$$

and if M is nonsingular then the vector K_N can be uniquely obtained for a given system from the initial conditions. Under these conditions if K_N is not in the region of interest then there exist no extrema in the region of search and the minimum of J should be on the boundary of the search region.

Unfortunately, if K_N lies within the region of interest nothing can be said in general about this extremal, if it exists, since sufficiency conditions for a minimum or a maximum cannot be satisfied. To show this

$$\begin{bmatrix} \frac{\partial^2 J}{\partial k_{n1}^2} & \frac{\partial^2 J}{\partial k_{n1}\partial k_{n2}} & \cdots\cdots\cdots & \frac{\partial^2 J}{\partial k_{n1}\partial k_{nn}} \\ \frac{\partial^2 J}{\partial k_{n2}\partial k_{n1}} & & & \\ \vdots & & & \\ \frac{\partial^2 J}{\partial k_{nn}\partial k_{n1}} & \cdots\cdots\cdots\cdots\cdots\cdots & & \frac{\partial^2 J}{\partial k_{nn}^2} \end{bmatrix} = b^2 M$$

Since the m_{11} element of the M matrix is always zero, it is clear that M can never be positive or negative definite; and, therefore, the sufficiency condition for a maximum or a minimum cannot be satisfied. In fact, it is possible for K_N to be a saddle point.

One disadvantage with the gradient method is that the computer program becomes quite involved for the general case which includes the possibility of M being singular. On the other hand, because of the simplicity of the function to be minimized and because the region of search is easily defined, it has been possible to prepare a simple digital computer program based on the exhaustive search technique that is extremely fast and very satisfactory. For example, the neighborhood of the absolute minimum of a third order system was obtained with sufficient accuracy in about one minute on an IBM 1800 computer, most of the time being consumed by the printer. Adequate resolution was obtained after four successive contractions of the search region.

The procedure is summarized as follows:

1. Determine the maximum and minimum k_{ni}, $i = 1,2,\ldots,n$, and use as input to the computer program;
2. Choose α_i, $0 < \alpha_i < 1$, to increment k_{ni}, i.e.,

$$\Delta k_{ni} = \alpha_i[\max k_{ni} - \min k_{ni}]$$

It has been found that $\alpha_i = 0.1$ gives satisfactory results and has been incorporated in the computer program;

3. Starting with the n-tuple $[\max k_{n1}, \max k_{n2}, \ldots, \max k_{nn}]$ use the scheme in Figure 3-1 until the whole region is scanned;
4. Determine the minimum J obtained from Step 3 and form a neighborhood about the corresponding K_N in K_N-space that includes the absolute minimum of J. This contracted region now defines new maxima and minima for the k_{ni};
5. Repeat Steps 3 and 4 until the desired accuracy in locating

the minimum of J is achieved;

6. If the value obtained in Step 5 is such that all state variable constraints are satisfied, then this is the desired solution to the minimization problem.

In general, the above procedure does not yield a solution where the state variable constraints are satisfied. To implement the search for this solution a special constraint function has been developed which forms the basis of a hybrid computer program that complements the Minimum Cost Algorithm program.

3.6 Constraint Function.

Any type of constraint function is acceptable providing it can be formulated in mathematical terms and at the same time reflects a measure of "goodness" with respect to a physically meaningful criterion. For example, Kalman [KAL-3] discusses the Lyapunov function of a stable system as useful in evaluating transient response. Thus, given a constraint in terms of "Lyapunov distance" from the origin in state space as a function of time, an upper bound can be found and invoked as criterion of goodness.

If $V(x,t)$ is some Lyapunov function, then

$$\dot{V}(x,t) = \frac{\dot{V}(x,t)}{V(x,t)} V(x,t) \leq cV(x,t)$$

where c is a suitably chosen constant. Then by a well-known Lemma [BEL-1]

$$V(x,t) \leq \exp[-c(t - t_o)] V(x_o,t_o)$$

It has been shown that the matrices P and H exist for a

time invariant system such that P and H are symmetric and positive definite and

$$V(x) = x^T Px$$

$$\dot{V}(x) = -x^T Hx$$

Kalman continues, showing that if λ_{min} is the minimum eigenvalue of the matrix HP^{-1}, then

$$c = \lambda_{min}$$

Thus, the Lyapunov distance is bounded by a function of the longest time constant. Such a bound is not new in the field of differential equations [CES-1].

It is well known that the characteristic equation for the G matrix in equation (16) is

$$\lambda^n + p_n\lambda^{n-1} + \ldots\ldots\ldots + p_2\lambda + p_1 = 0$$

As shown previously,

$$p_n = -(\lambda_1 + \lambda_2 + \ldots\ldots\ldots + \lambda_n) = 0$$

$$p_{n-1} = (\lambda_1\lambda_2 + \lambda_1\lambda_3 + \ldots + \lambda_1\lambda_n + \lambda_2\lambda_3 + \ldots\ldots + \lambda_{n-1}\lambda_n)$$

$$\cdot$$

$$\cdot$$

$$p_1 = (-1)^n(\lambda_1\lambda_2 \ldots\ldots \lambda_n)$$

and, therefore, there exists a direct relationship between the eigenvalues of G and the parameters p_i. However, it is quite possible for a reasonable change in p_i to have a relatively small effect on the minimum eigenvalue; and there could be regions in p-space such that the criterion based on λ_{min} may be too

conservative, i.e., relatively insensitive.

In contrast to the minimum eigenvalue type of constraint, a constraint based on maximum allowable state variable excursion is usually a well-known value. The designer is well aware of the limitations such as maximum acceleration, stress, etc. It is this type of constraint that has been chosen:

$$x_i(\max) \geq x_i(t) \geq x_i(\min); \text{ all } t \tag{26}$$

The constraint region defined by equation (26) is still not in a completely satisfactory form since it only provides upper and lower bounds on the constrained state variables. While it reduces the region of search, it still represents an uncountable number of solutions. The search for the best solution within the constraint region is made possible by changing equation (26) from an inequality constraint to an equality constraint. To implement this need, a constraint function C is defined which serves as a test function that senses the deviation and the direction of the deviation from the constraint bounds. In its structure the function is similar to the penalty functions used in adjoining to the performance index J in conventional optimal control design involving constraints.

Define

$$h_i = [x_i(\max) - x_i(t_i)][x_i(t_i) - x_i(\min)]$$
$$t_i \ni |x_i(t_i)| \geq x_i(t) \; ; \quad t_i, t \in [t_o, T] \tag{27}$$

Then it is clear that

$$h_i \geq 0 \quad \text{if} \quad x_i(\max) \geq x_i(t_i) \geq x_i(\min)$$

$$h_i < 0 \quad \text{if} \quad x_i(t_i) > x_i(\max) \quad \text{or} \quad x_i(t_i) < x_i(\min)$$

Let H be a function of h_i such that

$$H(h_i) = \begin{cases} 0 & \text{if} \quad h_i \geq 0 \\ 1 & \text{if} \quad h_i < 0 \end{cases} \tag{28}$$

if at least one state variable exceeds its bounds .

Let H be function of h_i such that

$$H(h_i) = 1 \; ; \; i \in \ell \leq n \tag{29}$$

if all the state variables lie at or inside their respective constraints and ℓ defines a subspace of E^n which contains the constrained state variables.

A constraint function C is formed such that

$$C = \sum_{i \in \ell} [x_i(\max) - x_i(t_i)][x_i(t_i) - x_i(\min)] \, H(h_i) \tag{30}$$

which is used to test the solution obtained with the Minimum Cost Algorithm for constraint violation; and if the constraints are violated, it is used to obtain a new estimate for the K_N vector such that all constraints are satisfied.

3.6.1 Geometric Interpretation.

Assume that x_i and x_j are two constrained state variables. Expanding equation (27),

$$h_i = -x_i^2(t_i) + [x_i(\max) + x_i(\min)]x_i(t_i) - x_i(\max)x_i(\min)$$

$$h_j = -x_j^2(t_j) + [x_j(\max) + x_j(\min)]x_j(t_j) - x_j(\max)x_j(\min)$$

Each of the h-functions represents a parabola as shown in Figure 3-2 with its maximum value occurring at

$$x_i(t_i) = \frac{x_i(\max) + x_i(\min)}{2} \tag{31}$$

Since the C-Function represents the sum of two parabolas in $x_i \times x_j \times R$ space, it can be readily visualized as shown in Figure 3-3. The shaded region represents the surface of allowable solutions, $C \geq 0$. This surface meets the plane $C = 0$ at exactly four points a,b,c and d which define the four minima. Below the plane $C = 0$, the surface $C < 0$ is a paraboloid truncated at the top by the circle a,b,c,d in the plane $C = 0$.

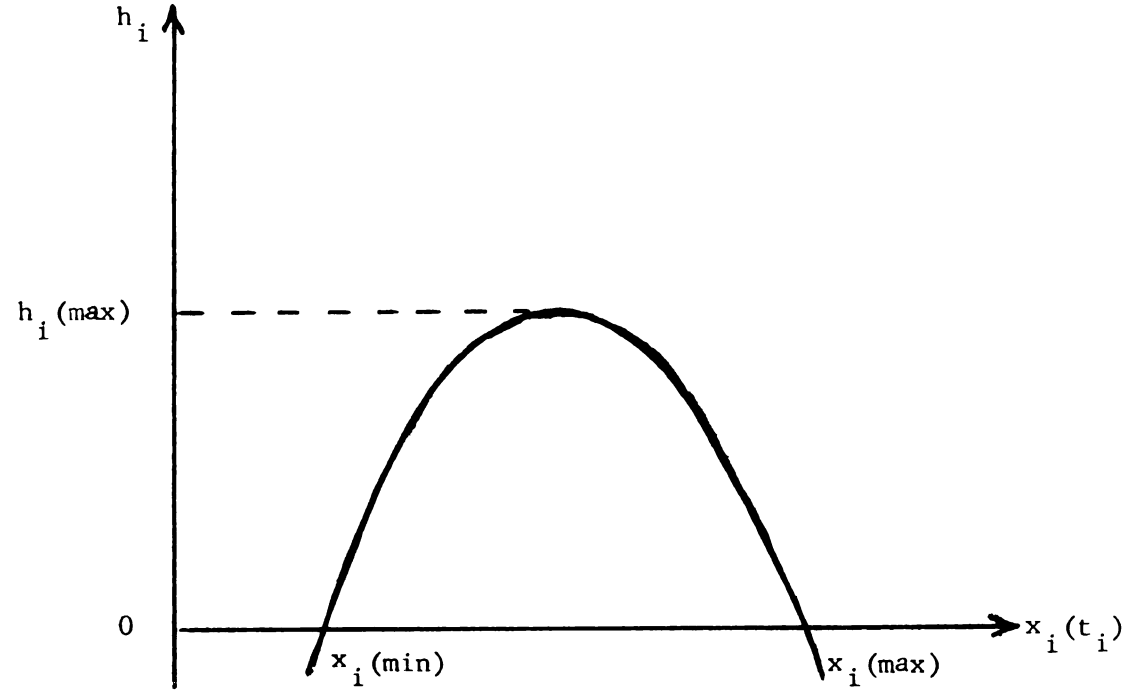


Figure 3-2

h_i Function

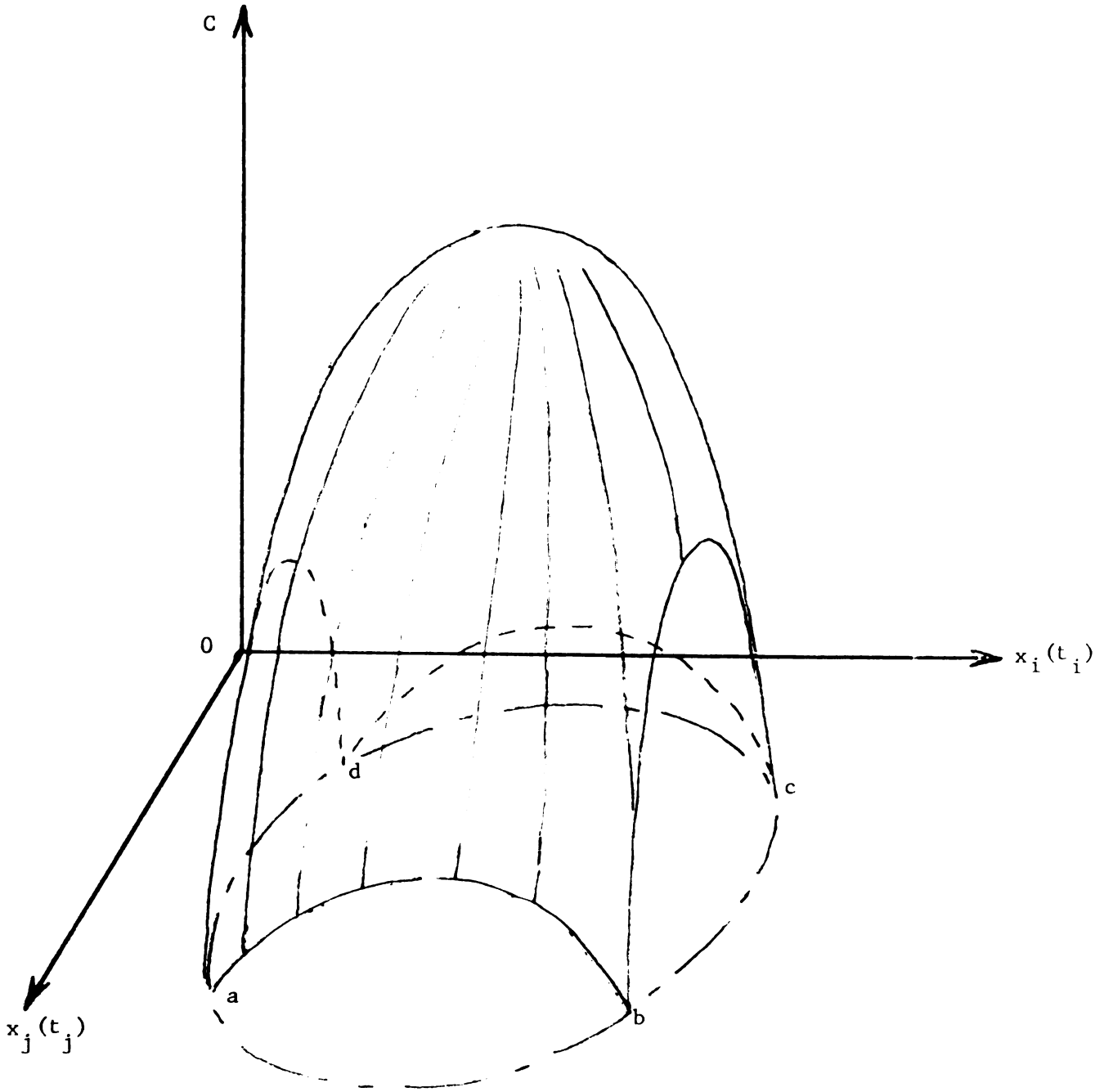


Figure 3-3

C-Function in Two-Space

Theorem II:

Given the constraint function

$$C = \sum_{i \in \ell} [x_i(\max) - x_i(t_i)][x_i(t_i) - x_i(\min)]H(h_i)$$

$$x_i(\max) \geq x_i(t_i) \geq x_i(\min)$$

where ℓ is the number of constrained state variables, $\ell \leq n$,

$\frac{\partial C}{\partial x_i(t_i)}$ and $\frac{\partial^2 C}{\partial x_j(t_j)\partial x_i(t_i)}$, $i,j \in \ell$, exist and are continuous,

then C has at most 2^ℓ minima, all of which occur at $C = 0$.

Proof:

Clearly, if $x_i(\max) \geq x_i(t_i) \geq x_i(\min)$ then

$$h_i = [x_i(\max) - x_i(t_i)][x_i(t_i) - x_i(\min)] \geq 0$$

but $h_i = 0$ if and only if either $x_i(t_i) = x_i(\max)$ or $x_i(t_i) = x_i(\min)$. If $x_i(t_i) > x_i(\max)$ or $x_i(t_i) < x_i(\min)$ then $h_i < 0$ and these $x_i(t_i)$ are not allowable.

Since all $H(h_i) = 1$, $i \in \ell$,

$$C = \sum_{i \in \ell} h_i$$

then $C = 0$ if either each $h_i = 0$ or some $h_i < 0$.

But $h_i < 0$ implies $x_i(\max) \geq x_i(t_i) \geq x_i(\min)$ does not hold.

Therefore, $C = 0$ if and only if each $h_i = 0$.

Then the number of ways that C can be zero is the number of ways that $\sum_{i \in \ell} h_i = 0$ holds.

Since each h_i can be zero in two ways, then $\sum_{i \in \ell} h_i$ can be zero in at most 2^ℓ ways.

Since $C < 0$ is not allowable, then the zeros of C correspond

to the minima of C when all $x_i(t_i)$, $i \in \ell$, attain one of their respective bounds. Therefore, C can have at most 2^ℓ minima when $x_i(t_i) = x_i(\max)$ or $x_i(t_i) = x_i(\min)$.

To show no other minima of C exist inside the constraint region: the necessary condition for an extremum is that

$$\frac{\partial C}{\partial x_i(t_i)} = 0, \quad i \in \ell$$

$$\frac{\partial C}{\partial x_i(t_i)} = \frac{\partial h_i}{\partial x_i(t_i)} = -2x_i(t_i) + [x_i(\max) + x_i(\min)]$$

implies $x_i(t_i) = \dfrac{x_i(\max) + x_i(\min)}{2}$ at extremum, i.e., each $x_i(t_i)$ corresponding to an extremum is unique since $x_i(\max)$ and $x_i(\min)$ are unique. These extrema then correspond to a unique maximum for C since

$$\frac{\partial^2 C}{\partial x_i^2(t_i)} = -2 < 0$$

$$\frac{\partial^2 C}{\partial x_i(t_i)\partial x_j(t_j)} = \frac{\partial^2 C}{\partial x_j(t_j)\partial x_i(t_i)} = 0, \quad i \neq j$$

i.e., the $\ell \times \ell$ matrix

$$\begin{bmatrix} \dfrac{\partial^2 C}{x_i^2(t_i)} & \cdots\cdots & \dfrac{\partial^2 C}{x_i(t_i)x_j(t_j)} \\ \vdots & & \\ \dfrac{\partial^2 C}{x_j(t_j)x_i(t_i)} & \cdots & \dfrac{\partial^2 C}{x_j^2(t_j)} \end{bmatrix} = -\begin{bmatrix} 2 & & & & & \\ & 2 & & 0 & & \\ & & 2 & & & \\ & 0 & & 2 & & \\ & & & & 2 & \\ & & & & & 2 \end{bmatrix}$$

is negative definite. Therefore, all the minima of C occur at $C = 0$ when constraints are invoked.

Q.E.D.

3.6.2 C-Function Iterative Procedure.

In the event that the parameter solution obtained by the Minimum Cost Algorithm is such that at least one of the state variable constraints is violated, then it is necessary to adjust the $\tilde{p}$ vector; and the C-Function provides a satisfactory means. Given $C < 0$, the objective is to develop a convergent iterative procedure so that successive estimates of the parameter vector $\tilde{p}$ yield a solution for $C = 0$. Clearly, this solution yields a cost J_c which is an upper bound for the desired solution.

Since the iteration is on $\tilde{p}$, it is necessary to transform the C-Function into p-space. The gradient of C in x-space is defined by the $\ell \times 1$ vector

$$\nabla_x C = \begin{bmatrix} \frac{\partial C}{\partial x_i(t_i)} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial C}{\partial x_j(t_j)} \end{bmatrix} \quad , \; i,j \in \ell \tag{32}$$

while in p-space it is defined as the $n \times 1$ vector

$$\nabla_p C = \begin{bmatrix} \frac{\partial C}{\partial p_1} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial C}{\partial p_n} \end{bmatrix} \tag{33}$$

Since $x_i(t_i) = f_i[p_1, p_2, \ldots\ldots, p_n, x_i(t_o)]$ if the partial derivatives $\frac{\partial x_i(t_i)}{\partial p_j}$ exist, then

$$\nabla_p C = \begin{bmatrix} \frac{\partial x_i(t_i)}{\partial p_1} & \ldots\ldots\ldots\ldots & \frac{\partial x_j(t_j)}{\partial p_1} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ \frac{\partial x_i(t_i)}{\partial p_n} & & \frac{\partial x_j(t_j)}{\partial p_n} \end{bmatrix} \begin{bmatrix} \frac{\partial C}{\partial x_i(t_i)} \\ \\ \\ \\ \frac{\partial C}{\partial x_j(t_j)} \end{bmatrix}$$

$$= S \nabla_x C \tag{34}$$

In the computation scheme it was assumed that these partial derivatives exist and successive estimates of $\tilde{p}$ were obtained using equation (34). However, in the event that these derivatives do not exist -- and this can happen, it will be shown -- special measures have been incorporated in the program to allow the iteration to continue to a solution. A more detailed discussion is presented in the following section of this chapter.

It is common practice to define the partial derivative $\frac{\partial x_i(t_i)}{\partial p_m}$ as the sensitivity of state variable x_i to the parameter

p_m at $t = t_i$. Thus, the S matrix in equation (34) may be defined as the sensitivity matrix of the system. Following the development on sensitivity analysis in Chapter II,

$$\begin{bmatrix} \frac{\partial x_1(t_i)}{\partial p_m} \\ \frac{\partial x_2(t_i)}{\partial p_m} \\ \vdots \\ \frac{\partial x_3(t_i)}{\partial p_m} \end{bmatrix} = -\epsilon^{Gt_i} \int_{t_o}^{t_i} \epsilon^{-G\tau} B x_m(\tau) d\tau \tag{35}$$

In the hybrid computer implementation of the parameter search, equation (35) is solved repeatedly on the analog computer at the various times t_i, t_j, etc. until the entire S matrix is determined.

An iterative algorithm for the C-Function search must in the least indicate the direction in which the $\tilde{p}$ vector must be changed for a closer estimate to the solution. The gradient $\nabla_p C$ obviously has this characteristic. This leads to the steepest ascent or descent scheme [PER-1], [KEL-1], [BEK-1]

$$\tilde{p}(i+1) = \tilde{p}(i) + \alpha \frac{\nabla_p C(i)}{\|\nabla_p C(i)\|} \qquad \text{(maximization)} \tag{36}$$

$$\tilde{p}(i+1) = \tilde{p}(i) - \alpha \frac{\nabla_p C(i)}{\|\nabla_p C(i)\|} \qquad \text{(minimization)} \tag{37}$$

where $\|\nabla_p C(i)\|$ is the Euclidean norm of the gradient vector

at the i^{th} iteration and α is a constant that determines the step size and is usually obtained empirically. The determination of α is usually a part of a "cut and try" loop within the iterative procedure since it depends on the dynamics of the particular system, i.e., the rate at which the gradient may be changing.

It is obvious that an iterative procedure that not only senses the direction but also provides an estimate of the required step size is highly desirable. This indeed is the basic appeal of the Newton-Raphson [PER-1], [BEK-1] method.

$$\tilde{p}(i+1) = \tilde{p}(i) - \frac{C(i)\nabla_p C(i)}{\|\nabla_p C(i)\|^2} \tag{38}$$

It has further advantage of quadratic convergence. However, care must be exercised in case $\nabla_p C(i) \to 0$.

Both the gradient and a modified version of the Newton-Raphson methods are used in the C-Function algorithm to take advantage of both procedures. The Newton-Raphson method in equation (38) is modified to include a step size constant γ

$$\tilde{p}(i+1) = \tilde{p}(i) - \gamma \frac{C(i)\nabla_p C(i)}{\|\nabla_p C(i)\|^2} \tag{39}$$

which has been found to be very helpful in

1) stabilizing any effects due to discontinuities induced by t_i;
2) overcoming the inherent accuracy limitations, particularily near $C = 0$, of an analog computer when measuring the S matrix elements.

In summary, the entire computational procedure is outlined in Figure 3-4. The initial region defined by $\max K_N$ and $\min K_N$

is searched for the global quadratic cost minimum, J_G, by the Minimum Cost Algorithm. The K matrix corresponding to an arbitrarily close estimate of J_G is checked for state variable constraint violations. If all constraints are satisfied, then this is the desired solution and the search is ended. If one or more state variable constraints are violated then the C-Function Algorithm is used to obtain the $C = 0$ solution, thus defining a K matrix such that all constraints are satisfied and a corresponding quadratic cost J_C which is clearly an upper bound for the desired solution. Using the same search region, increment K_N in the fashion described in Section 3.5, comparing the cost $J(i)$ at each point with J_C. If $J(i)$ is greater than J_C then continue search; but if $J(i)$ is less than J_C, check state variable constraints. If constraints are violated continue search; but if constraints are satisfied then $J(i)$ becomes the new (lower) upper bound. After the entire region is scanned, the entire process may be repeated for a closer estimate to the desired solution by choosing the smaller region obtained from the results of the previous search.

The detailed computational description of the Minimum Cost and C-Function Algorithms is covered in Chapter IV.

3.6.3 Discontinuity in p-Space.

The parameter search is based on measurements made at t_i, $i \in \ell$, $t_o \leq t_i \leq T$, when $x_i(t)$ attains its maximum magnitude as shown in Figure 3-5. If $x_i(t_i)$ exceeds the constraint an adjustment is made in $\tilde{p}$. It is not uncommon that in

adjusting $\tilde{p}$ to decrease the amplitude of peak 1, peak 2 may become the dominant one as shown in Figure 3-6, resulting in a sudden jump in the value of t_i and a discontinuity in $\nabla_p C$.

Safeguards are incorporated in the computer program so that in this case:

1) possible oscillation of the solution between peak 1 and peak 2 is quickly "damped" out by the step size constant γ in equation (39). The programming details will be described in Chapter IV, and an example of an actual case will be shown in Chapter V.

2) the peak closest to its constraint is chosen automatically as the solution after a finite number of iterations.

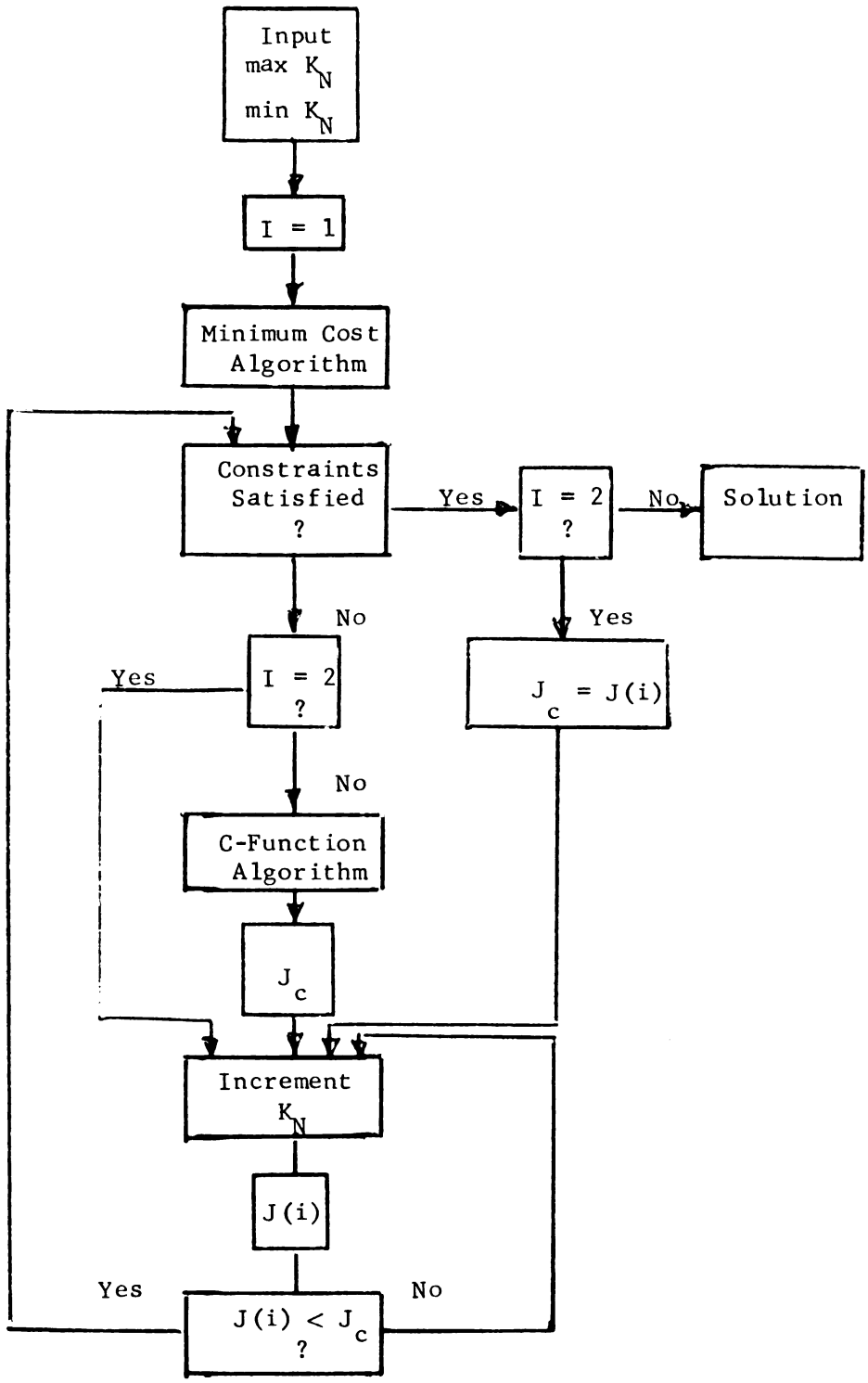


Figure 3-4

Algorithm to Obtain the Minimum

Cost K Matrix

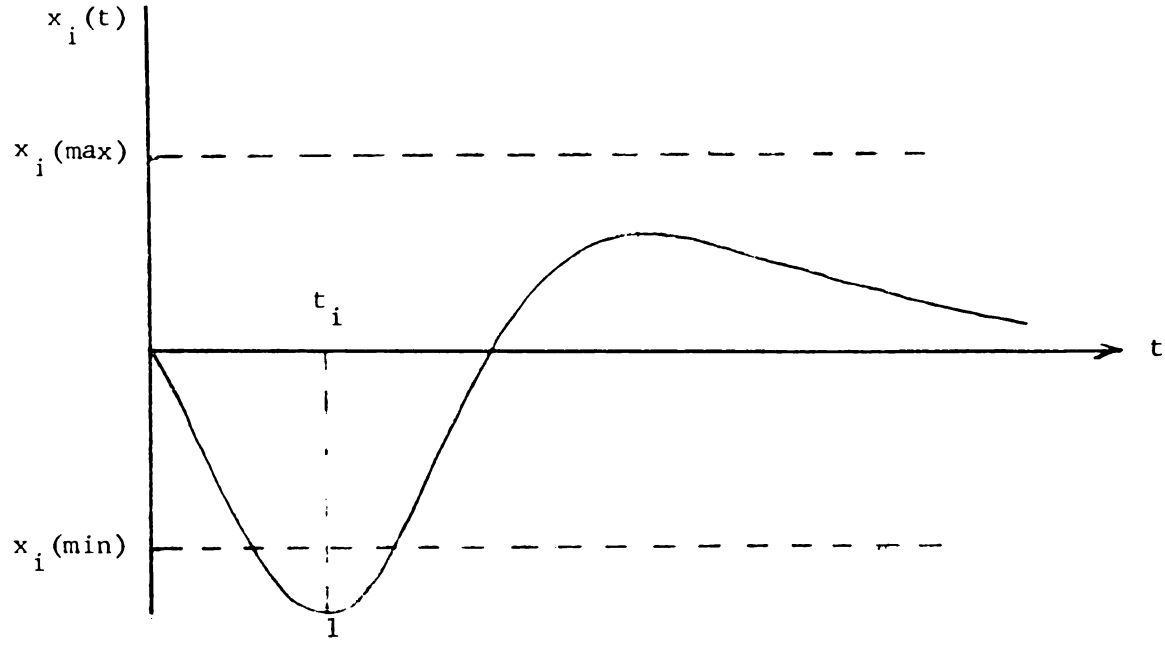


Figure 3-5

Peak 1 Dominant

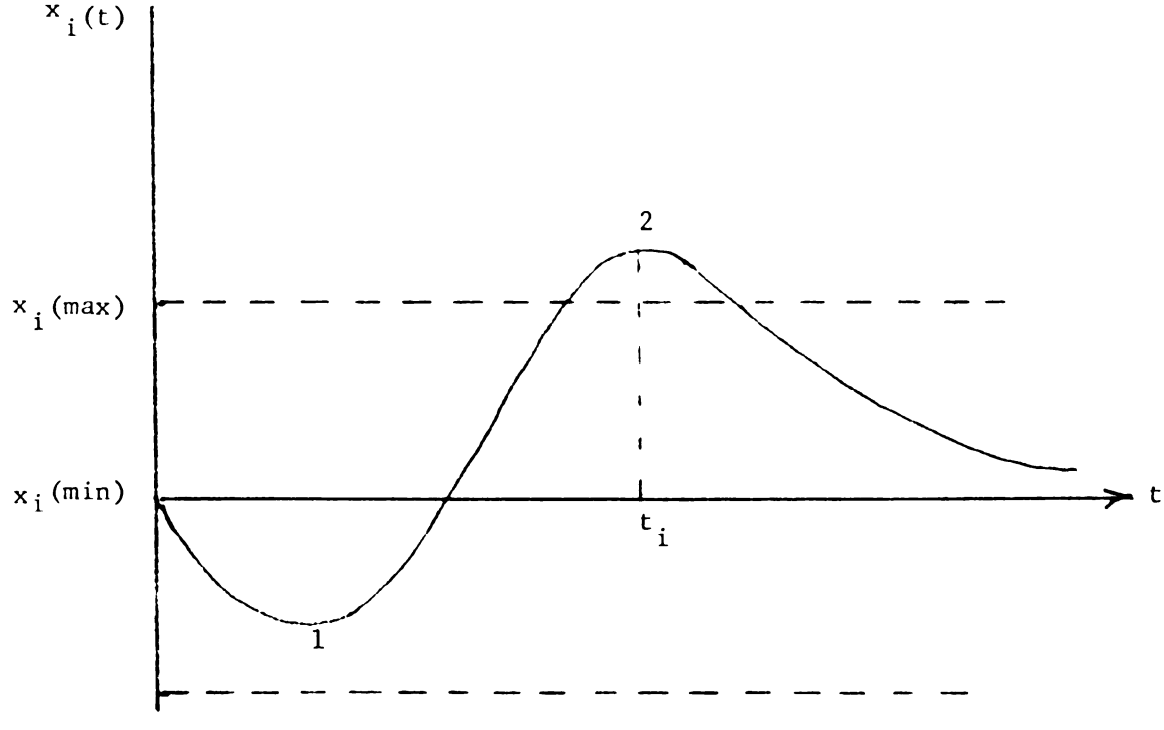


Figure 3-6

Peak 2 Dominant

CHAPTER IV

COMPUTER IMPLEMENTATION

The application of the concepts presented in the previous chapter generated a digital computer program for the Minimum Cost Algorithm and an hybrid program for the C-Function Algorithm. The computational details as well as the various procedures are described in this chapter.

4.1 Minimum Cost Algorithm Program.

This digital computer program locates the arbitrarily small neighborhood of the absolute minimum of

$$J = 1/2\ x^T(t_o)Kx(t_o)$$

by adjusting the K_N vector within the constraints imposed by the Routh and the Q_{ii} Criteria. Basically, an exhaustive search procedure is used, but the program has been written so that a gradient search can be readily implemented if desired. As mentioned in Chapter III, the exhaustive search approach is practicable because of the simplicity of the function to be minimized and because the region of search can usually be identified readily.

In respect to region identification, the initial search region is the hypercube in n-space, each "side" of which is equal to

$$[\max k_{ni} - \min k_{nj}]\ ;\ i = 1,2,.....,n$$

This defines a vector

$$\max K_N = [\max k_{n1}, \max k_{n2}, \ldots\ldots\ldots, \max k_{nn}]$$

which is chosen such that if the vector

$$[k^*_{n1}, k^*_{n2}, \ldots\ldots\ldots, k^*_{nn}]$$

corresponds to the absolute minimum of J, then

$$k^*_{ni} \leq \max k_{ni} \; ; \; i = 1,2,\ldots\ldots,n$$

A starting value for $\max K_N$ can be readily obtained by either invoking the relationships between the eigenvalues and the k_{ni} in the case a range of acceptable eigenvalues is available to the designer, or using the C-Function Algorithm and a "large enough" K_N guess.

The method of choosing a $\min K_N$ vector

$$\min K_N = [\min k_{n1}, \min k_{n2}, \ldots\ldots, \min k_{nn}]$$

was discussed in Chapter III.

One of the advantages to the exhaustive search procedure is that successive contractions of the search region can be readily identified and easily implemented within the computer program.

In the interest of clarity, a computer program for a third order system will be described. Extension to any n^{th} order system follows directly and presents no computational difficulties. The flow chart for this program is shown in Figures 4-1(a), (b), (c), (d), (e) and (f), and the listing is contained in Appendix A.

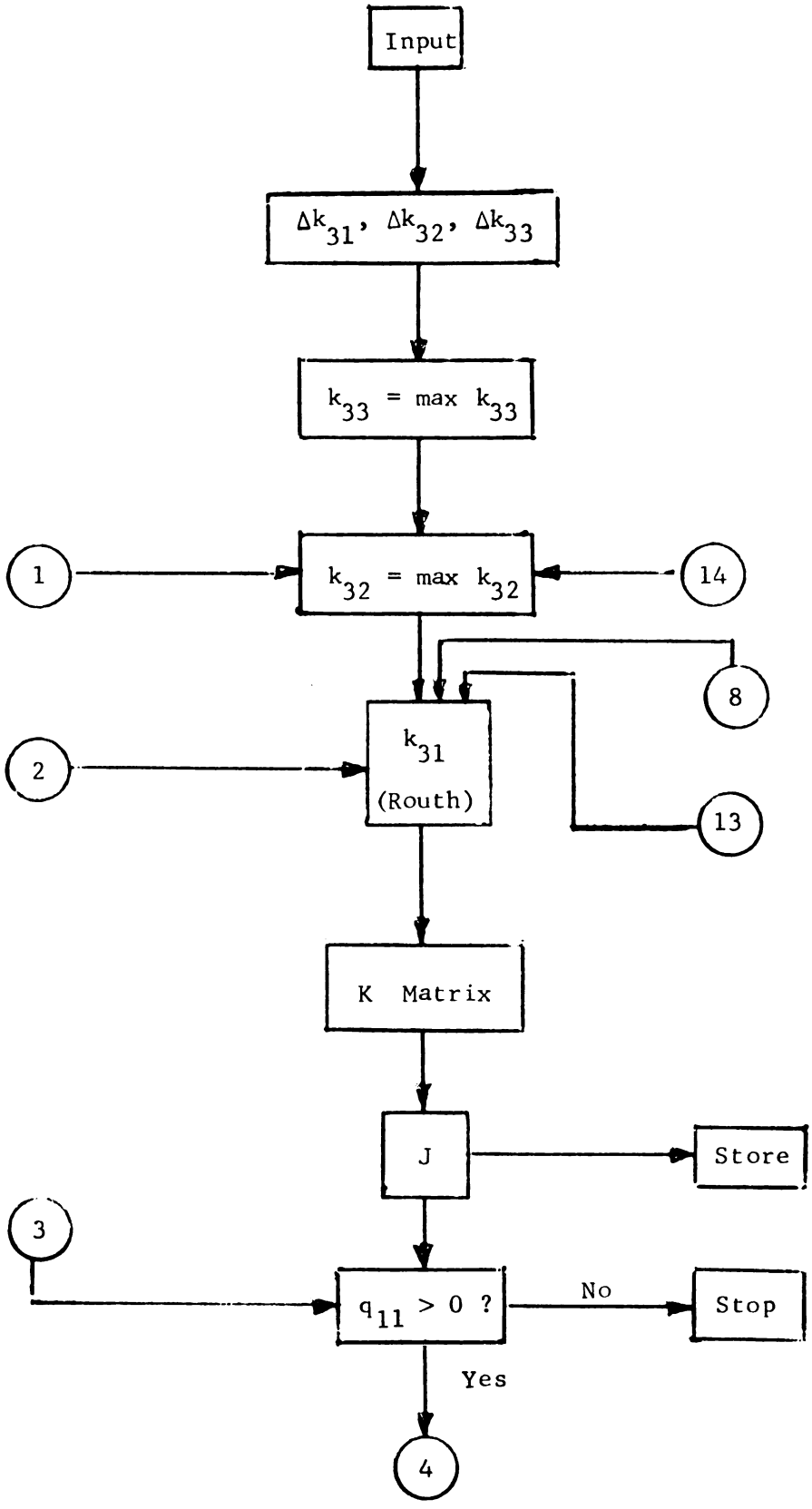


Figure 4-1(a)

Minimum Cost Program

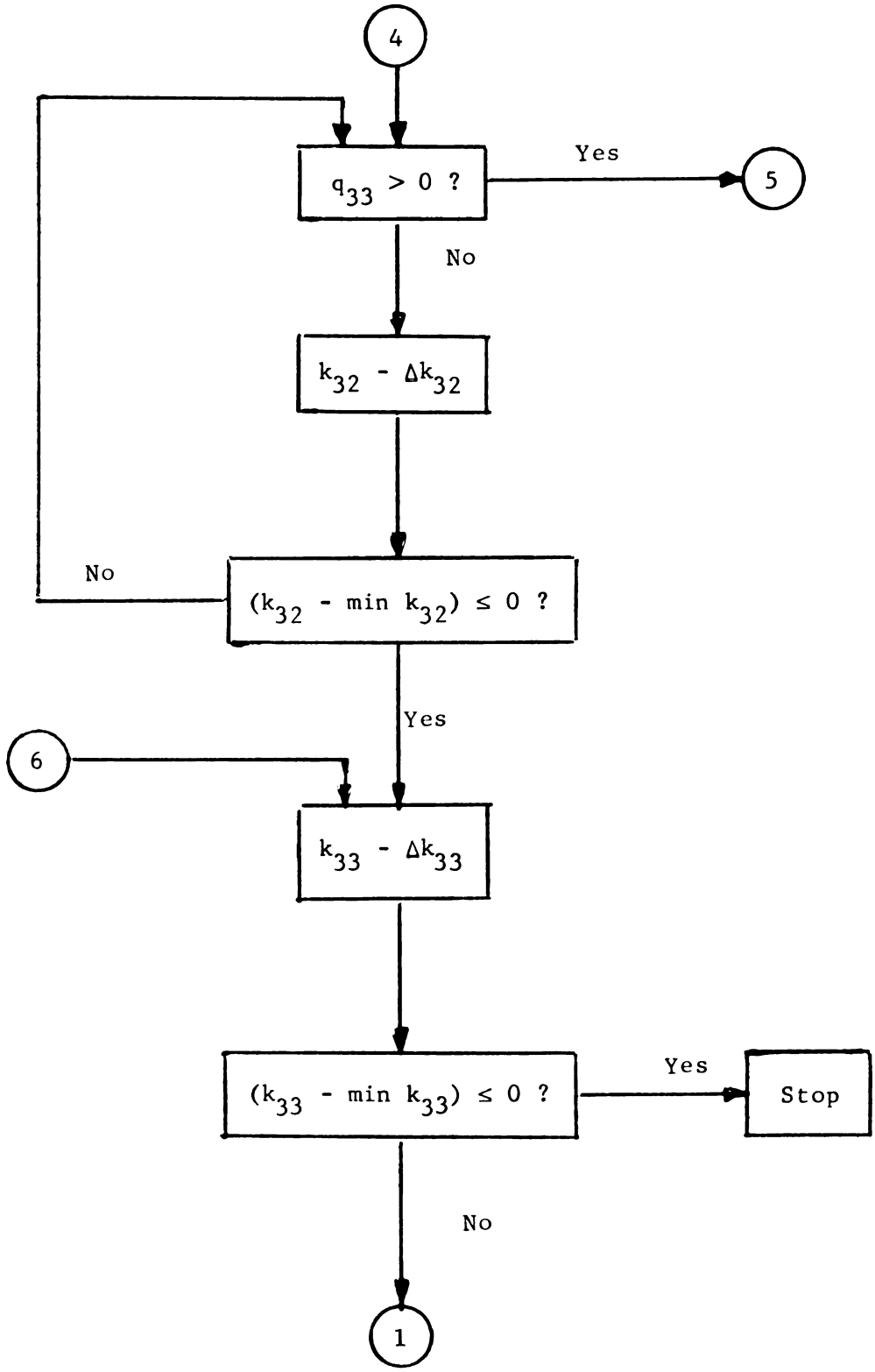


Figure 4-1(b)

Minimum Cost Program (con't.)

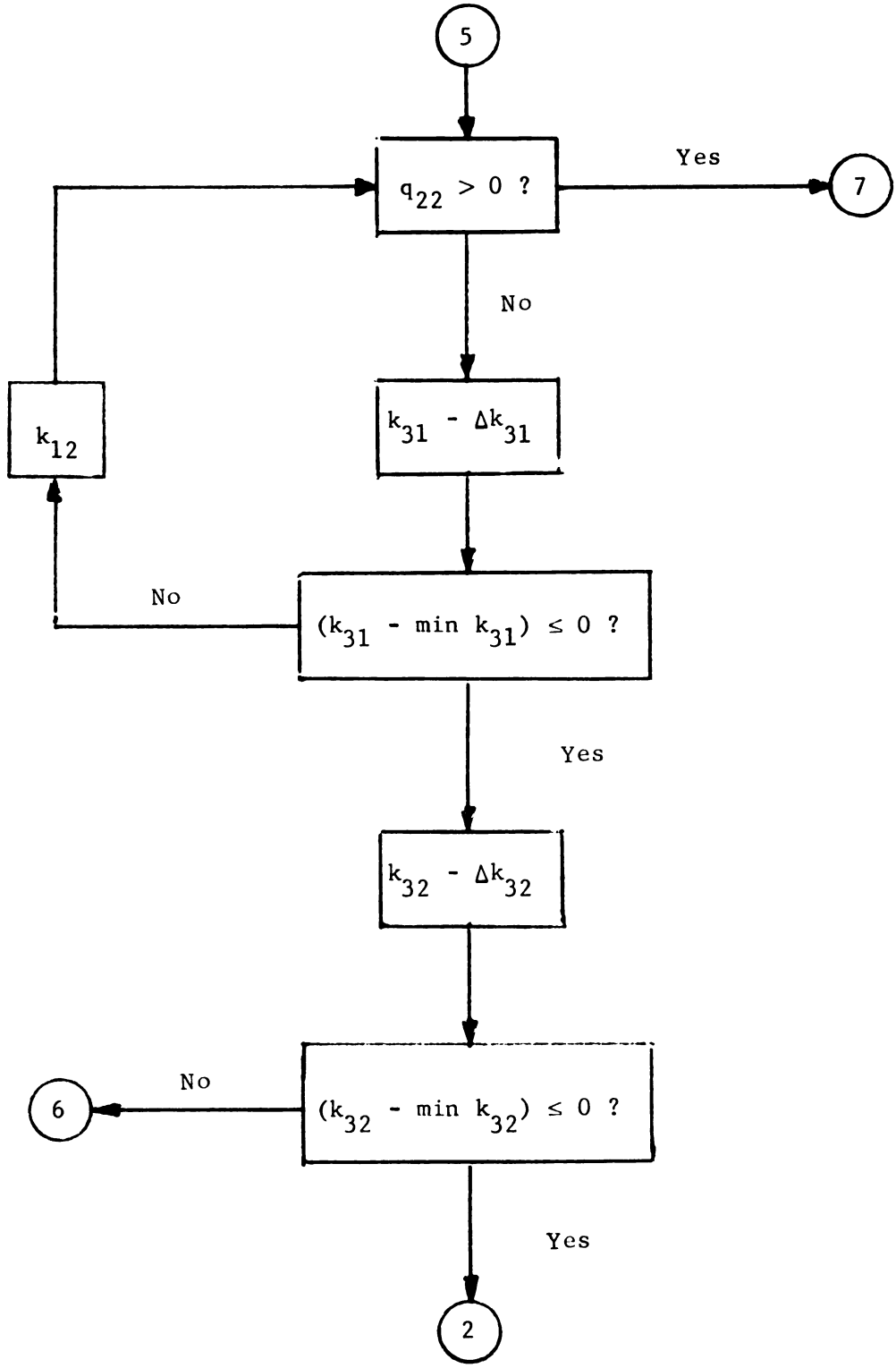


Figure 4-1(c)

Minimum Cost Program (con't.)

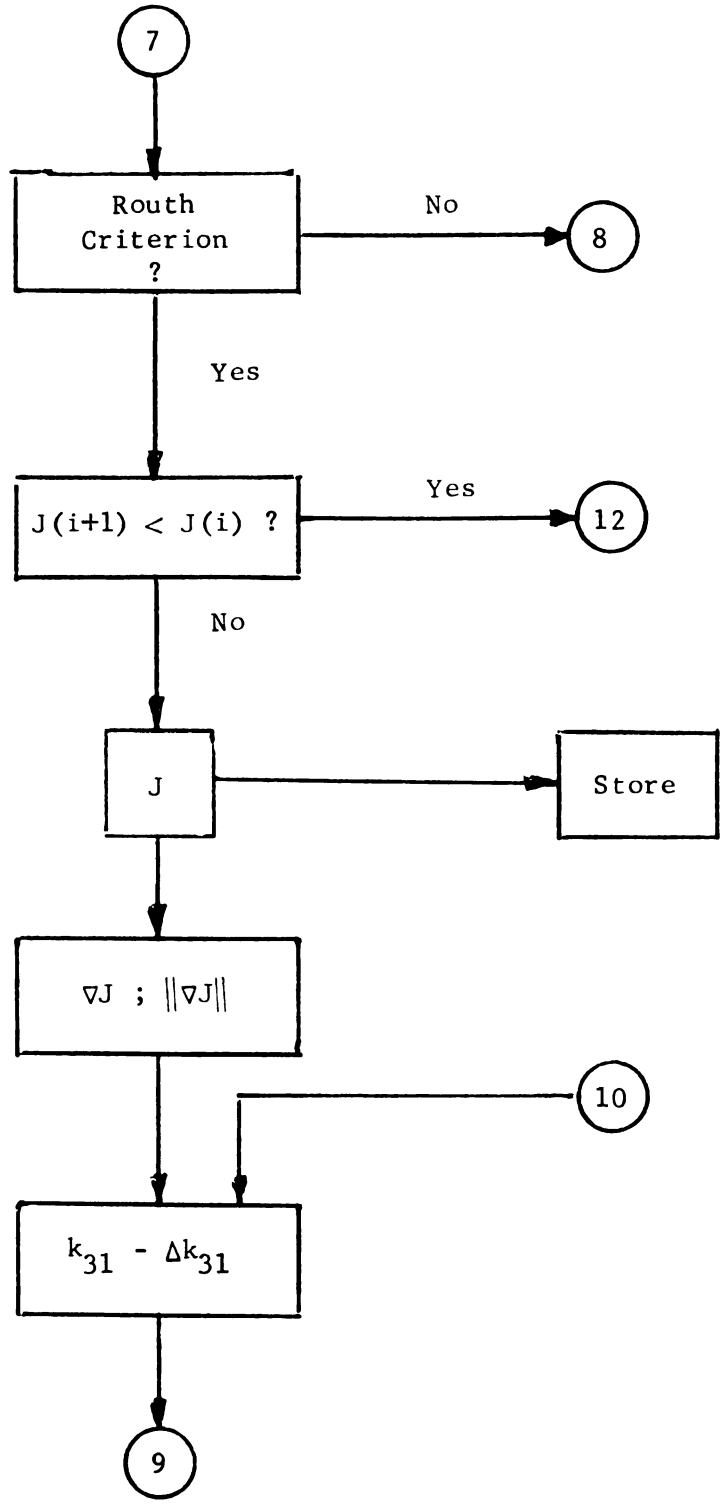


Figure 4-1(d)

Minimum Cost Program (con't.)

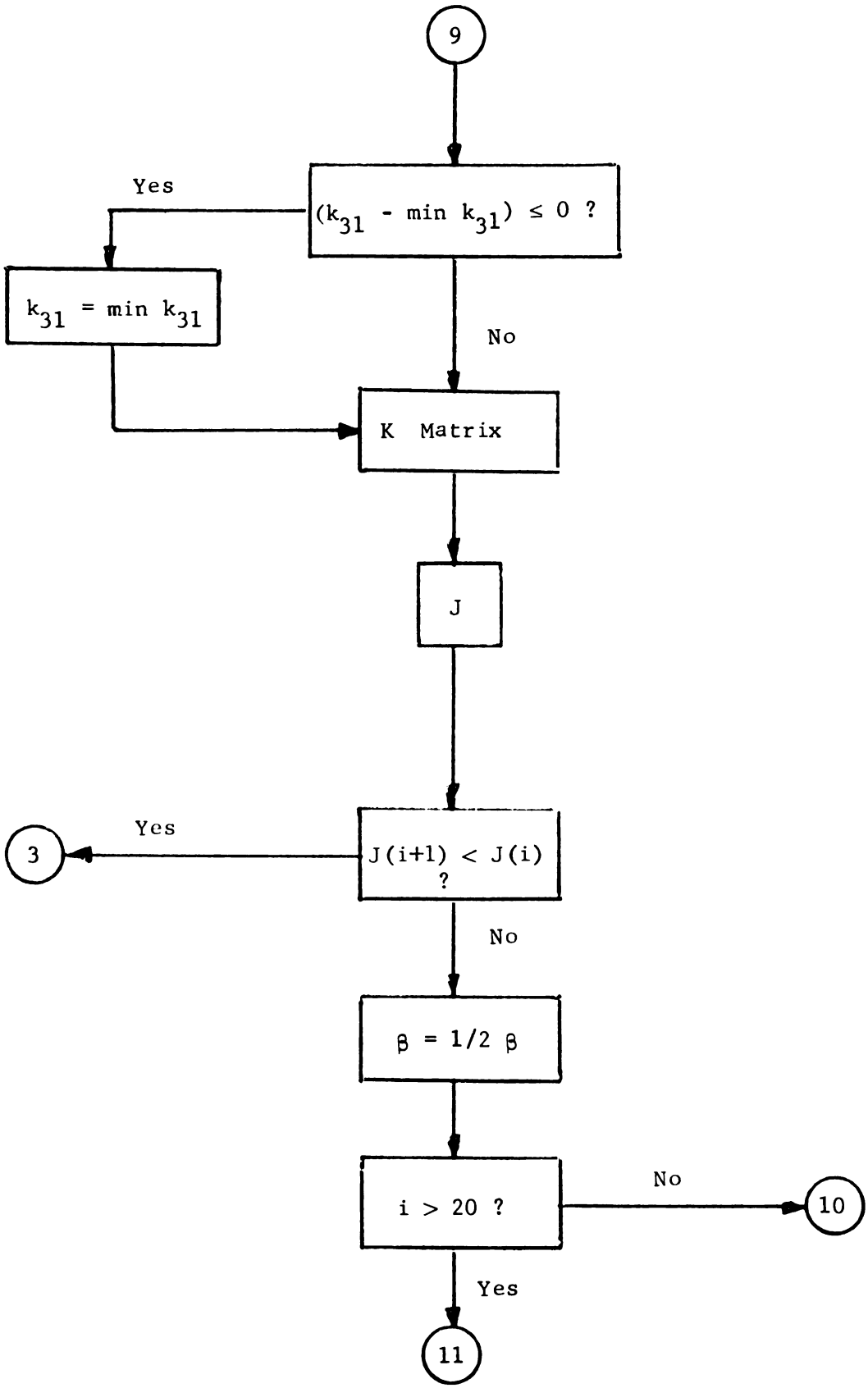


Figure 4-1(e)

Minimum Cost Program (con't.)

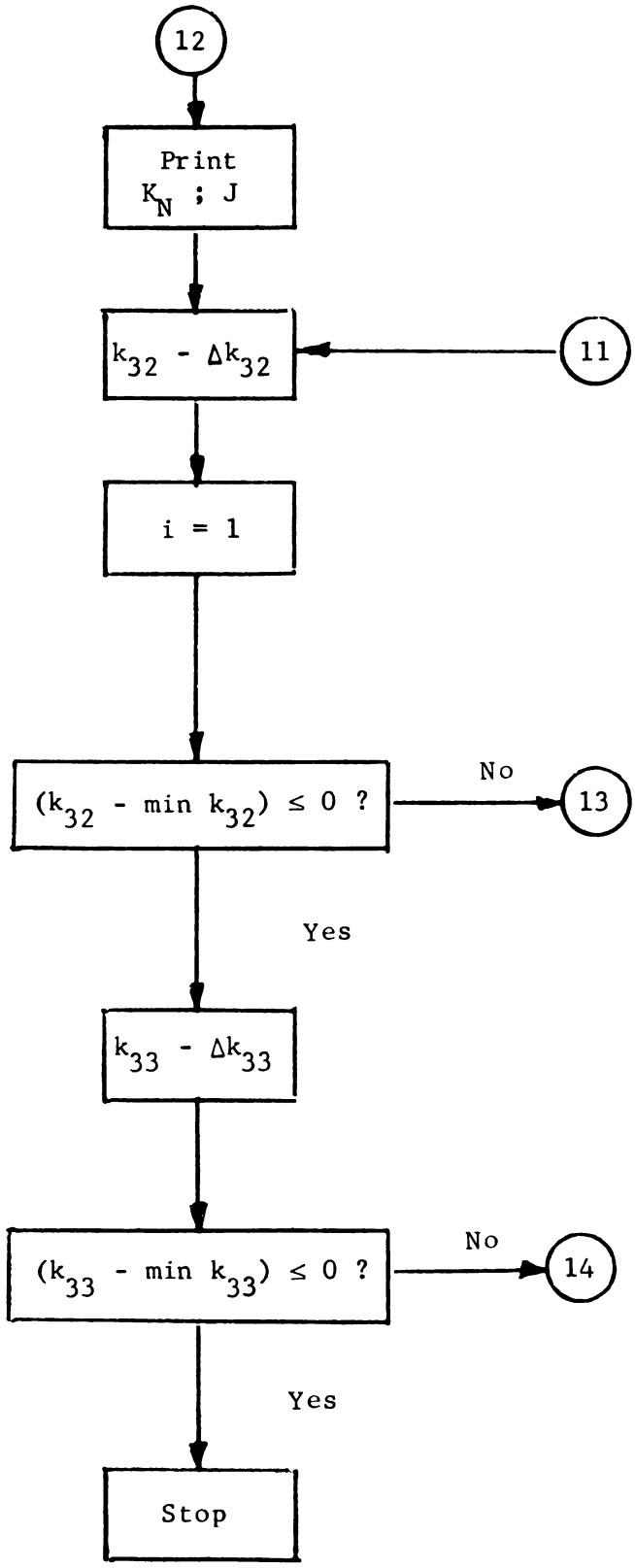


Figure 4-1(f)

Minimum Cost Program (con't.)

4.1.1 Incrementation.

The exhaustive search procedure requires incrementing each element k_{ni} of the K_N vector. In this program incrementation proceeds on the basis of

$$\Delta k_{ni} = \alpha_i[\max k_{ni} - \min k_{ni}]$$

Although α_i may be any value such that $0 < \alpha_i < 1$, satisfactory results are obtained by setting $\alpha_i = 0.1$. Since the Q_{ii} Criterion defines an open set, not all the values in the hypercube defined by $\max K_N$ and $\min K_N$ will be attained; but it is of interest to note that a lower bound is automatically obtained from

$$k_{ni} > k_{ni}(Q_{ii}) - \alpha_i[\max k_{ni} - \min k_{ni}]$$

where $k_{ni}(Q_{ii})$ is the minimum value of k_{ni} which satisfies the Q_{ii} and Routh Criteria.

4.1.2 Computation.

The computation is initiated by starting with the n-tuple $[\max k_{n1}, \max k_{n2}, \ldots\ldots, \max k_{nn}]$. A slight modification has been incorporated in this program which expedites the search: the search is begun with the (n-1)-tuple $[\max k_{32}, \max k_{33}]$ and an initial value for $\max k_{31}$ based on the Routh Criterion

$$\max k_{31} = \frac{(a_1 + b^2 k_{32})(a_2 + b^2 k_{33}) - a_o}{b^2} - 0.0001$$

where $\epsilon_1 = 0.0001$ is used to assure a strict inequality in the Routh Criterion.

An initial K matrix is computed using the algorithm described in Chapter III and used to determine an initial cost J. The Q_{ii} Criterion is then invoked

$$q_{11} = 2a_o k_{31} + b^2 k_{31}^2 > 0$$

$$q_{33} = -2k_{32} + 2a_2 k_{33} + b^2 k_{33}^2 > 0$$

$$q_{22} = -2k_{12} + 2a_1 k_{32} + b^2 k_{32}^2 > 0$$

$$= -2(a_o k_{33} + a_2 k_{31} + b^2 k_{31} k_{33}) + 2a_1 k_{32} + b^2 k_{32}^2 > 0$$

incrementing k_{33}, k_{32}, and k_{31} in that order until the criterion is satisfied. Next, the Routh Criterion is used to check for stability and the k_{ni} adjusted as necessary.

If both criteria are satisfied, ∇J and $\|\nabla J\|$ are computed. At this point the new estimate for K_N may be computed using either the gradient or the exhaustive search methods. In this program the choice was made to use a modified exhaustive search procedure where k_{32} and k_{33} are incremented with Δk_{32} and Δk_{33}, respectively, but k_{31} is incremented using

$$k_{31}(i+1) = k_{31}(i) - \beta \frac{\frac{\partial J}{\partial k_{31}}}{\|\nabla J\|}$$

This procedure further expedites the search; and for the example used, $\|\nabla J\| \neq 0$ in the region of interest.

The new cost J(i+1) is compared to J(i), and if the former is smaller, $K_N(i+1)$ and J(i+1) are stored, k_{32} incremented and the process repeated until $k_{32} < \min k_{32}$. This is followed by incrementing k_{33}, etc., until the whole region is scanned.

If $J(i+1)$ is greater than or equal to $J(i)$, then the step size β is decreased

$$\beta(i+1) = 1/2\ \beta(i)$$

until a lower cost is achieved or until a fixed number (20) of adjustments in β are exceeded.

After the entire region is scanned, a new contracted region containing within it the minimum J is identified, and the whole program repeated until the desired resolution in identifying minimum J is obtained.

4.2 C-Function Algorithm Program.

The hybrid computer program configuration consists of an IBM 1800 digital computer and an Applied Dynamics AD 4 analog computer. The IBM 1800 is used primarily to control the logic and for algebraic computations while the AD 4 is used for analog signal sensing and solving the differential equations. It is clear that numerical techniques, such as the Runge-Kutta, could be substituted with a resulting all-digital configuration, sacrificing computation speed for accuracy. The Fortran program is written for a general n^{th} order system with ℓ state variable constraints. The flow charts in Figures 4-2(a), (b) and (c) show the hybrid system and the program listing is contained in Appendix B.

4.2.1 Criteria Checks.

Every new estimate of the $\tilde{p}$ vector is checked to assure that the system $\dot{x} = Gx$ is asymptotically stable and the resulting

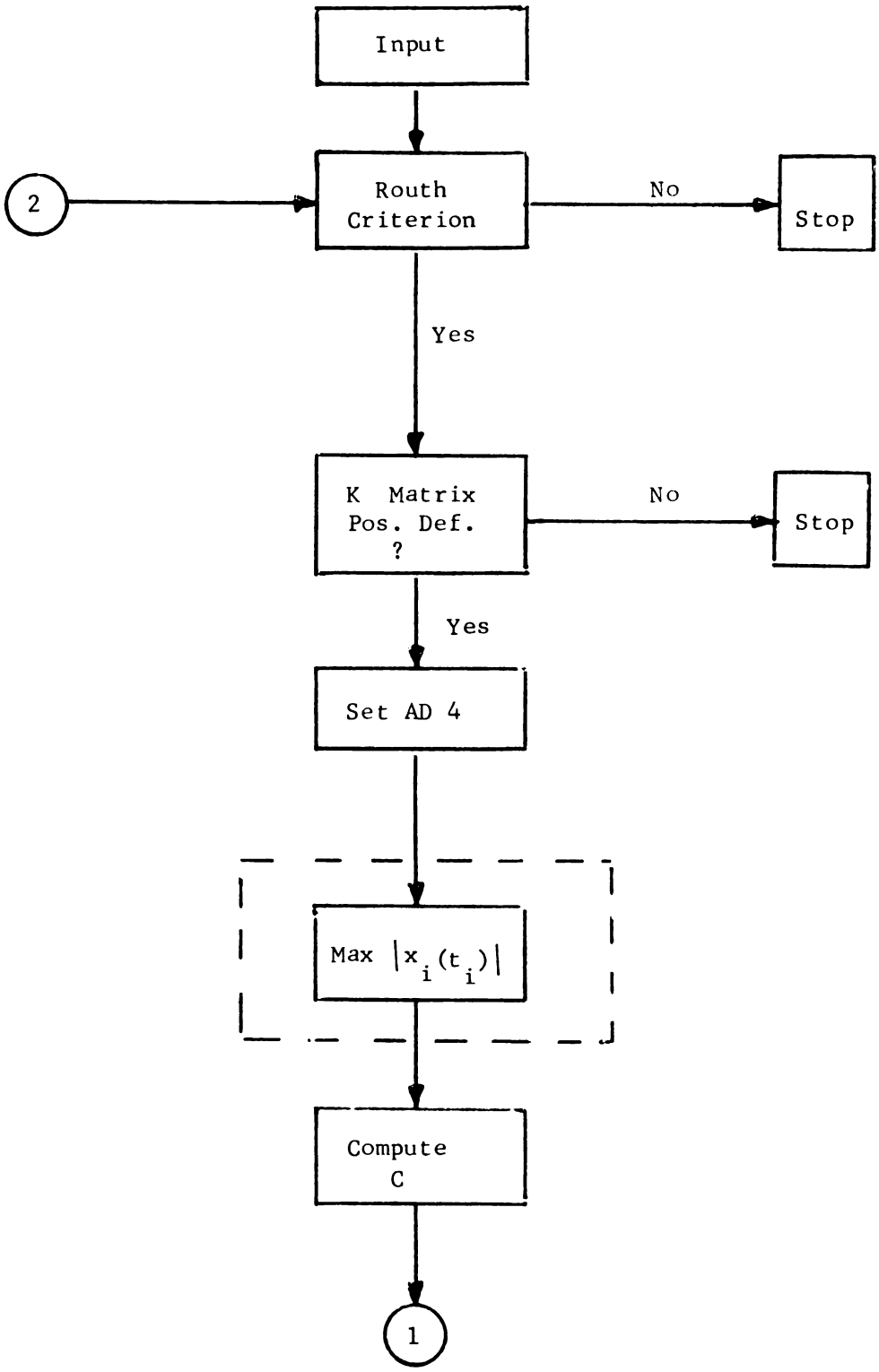


Figure 4-2(a)

C-Function Program

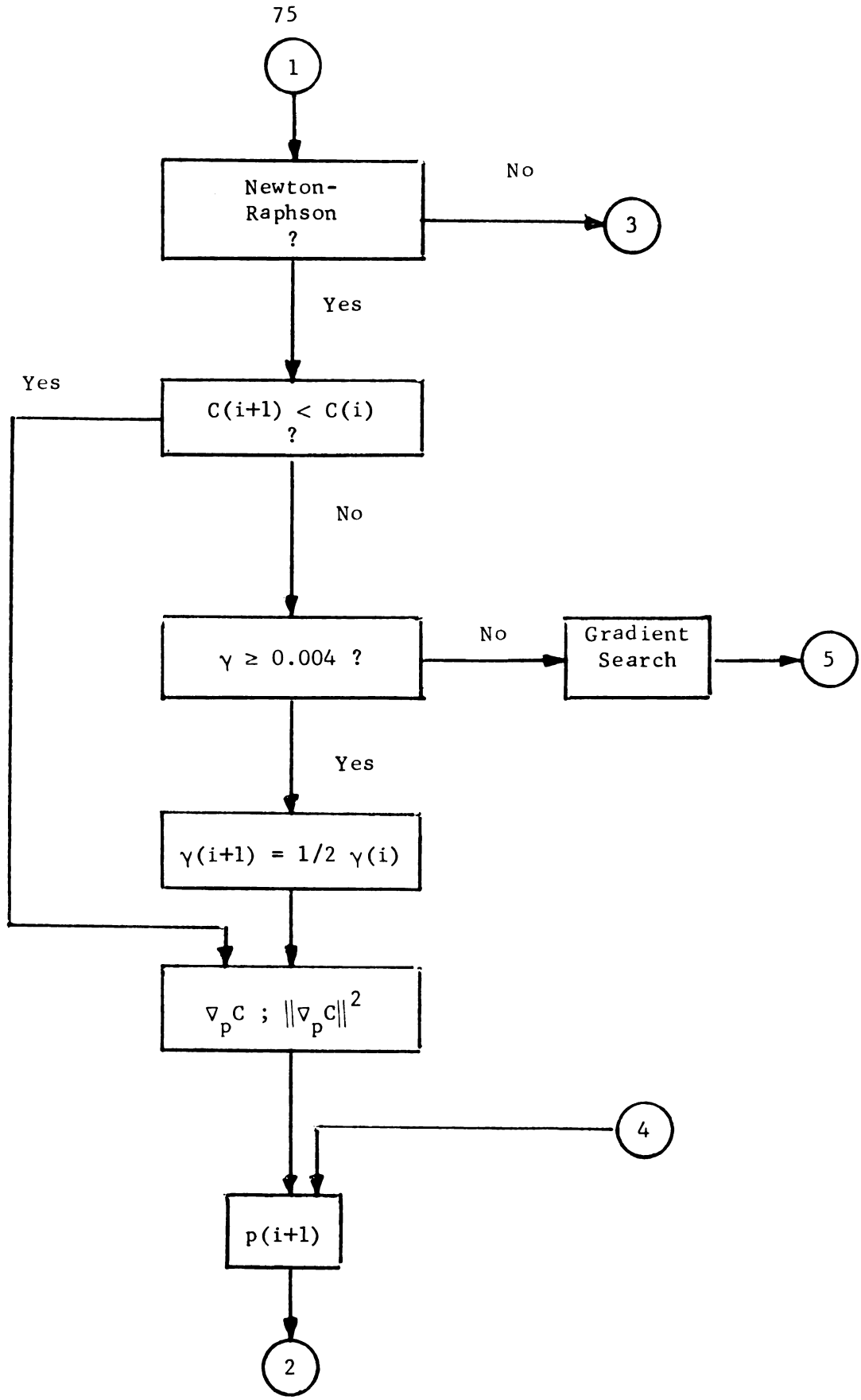


Figure 4-2(b)

C-Function Program (con't.)

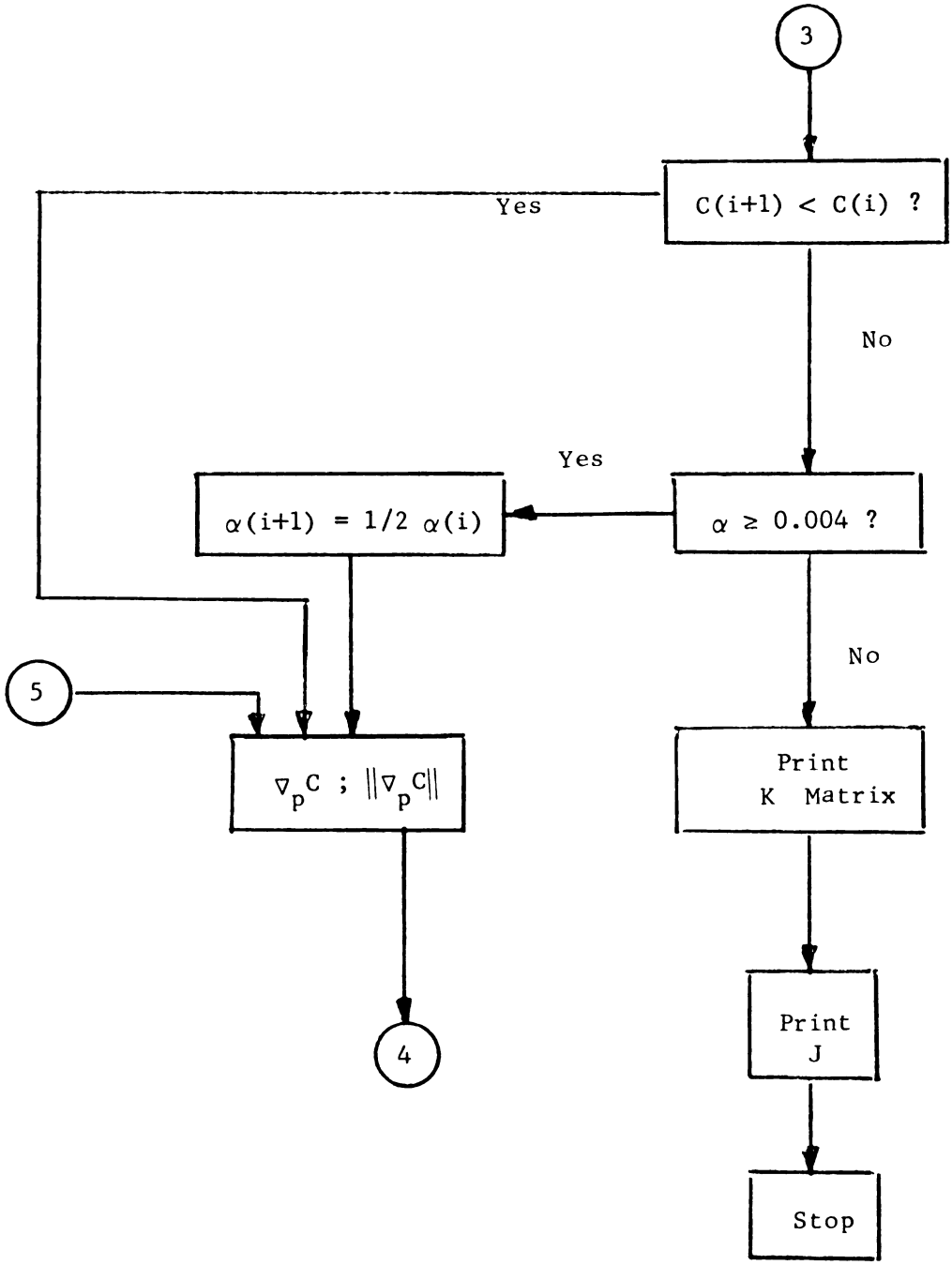


Figure 4-2(c)

C-Function Program (con't.)

K matrix is positive definite. The program determines the Routh array and checks each entry in the first column whether it is positive. The Hayes [HAY-1] algorithm is incorporated within the program to check the positive definiteness of the K matrix. Since violations of either criterion will be due usually to some pathological condition, the program stops in case of a violation and prints the violation.

4.2.2 Analog Computation

The analog computation, shown by the dashed-line box in Figure 4-2(a), has three basic operations requiring 2n+1 integrators: n integrators for the solution of $\dot{x} = Gx$; n integrators for the sensitivity analysis; and one integrator for signal timing. The analog connection diagram is shown for a third order system with a single state variable constraint in Figures 4-3(a), (b) and (c). It can, of course, be readily generalized to any n^{th} order system with ℓ state variable constraints.

The system solution operation is shown in Figure 4-3(a). The primary function is to solve the differential equation

$$\dot{x} = Gx \; ; \; x(t_o) = x_o$$

The servo-set potentiometers P_1, P_2 and P_3 are set with each successive iteration of the $\tilde{p}$ vector so that the final parameter solution is represented by these potentiometer settings. The analog signals representing x_1, x_2 and x_3 are transferred to the digital computer via analog trunk lines TA 16, TA 15 and TA 10, respectively.

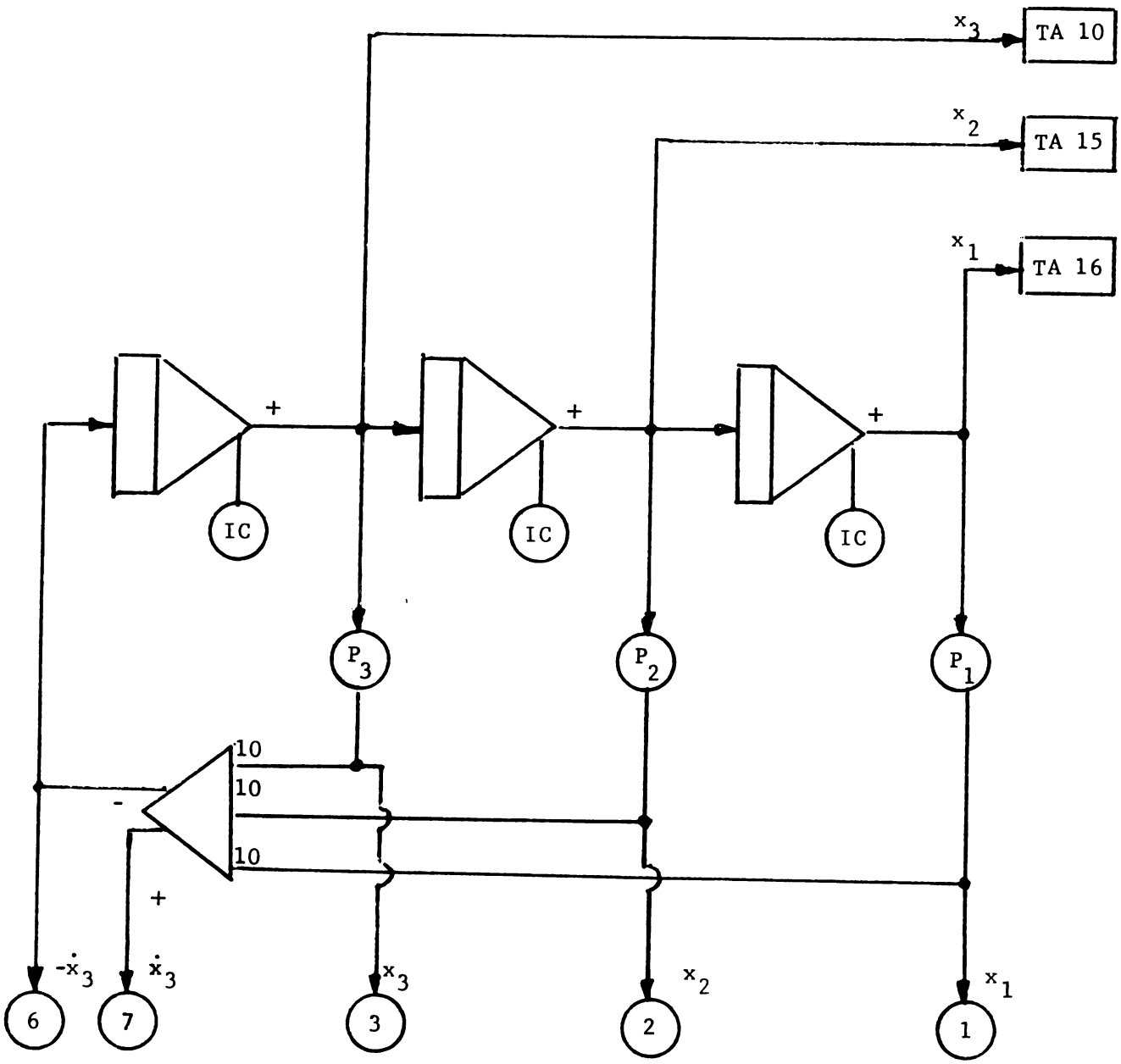


Figure 4-3(a)

Solution of $\dot{x} = Gx$

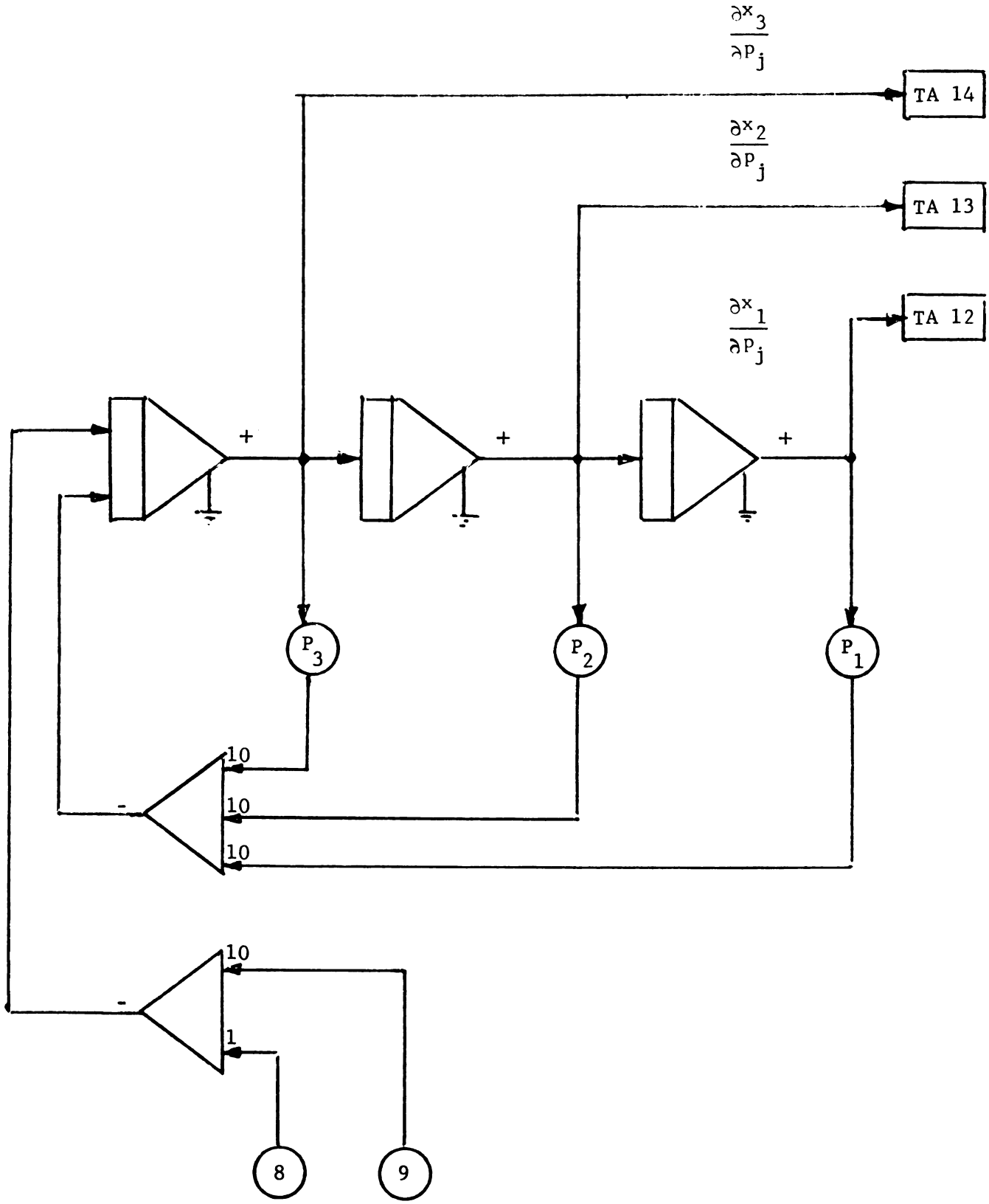


Figure 4-3(b)

Sensitivity Analysis

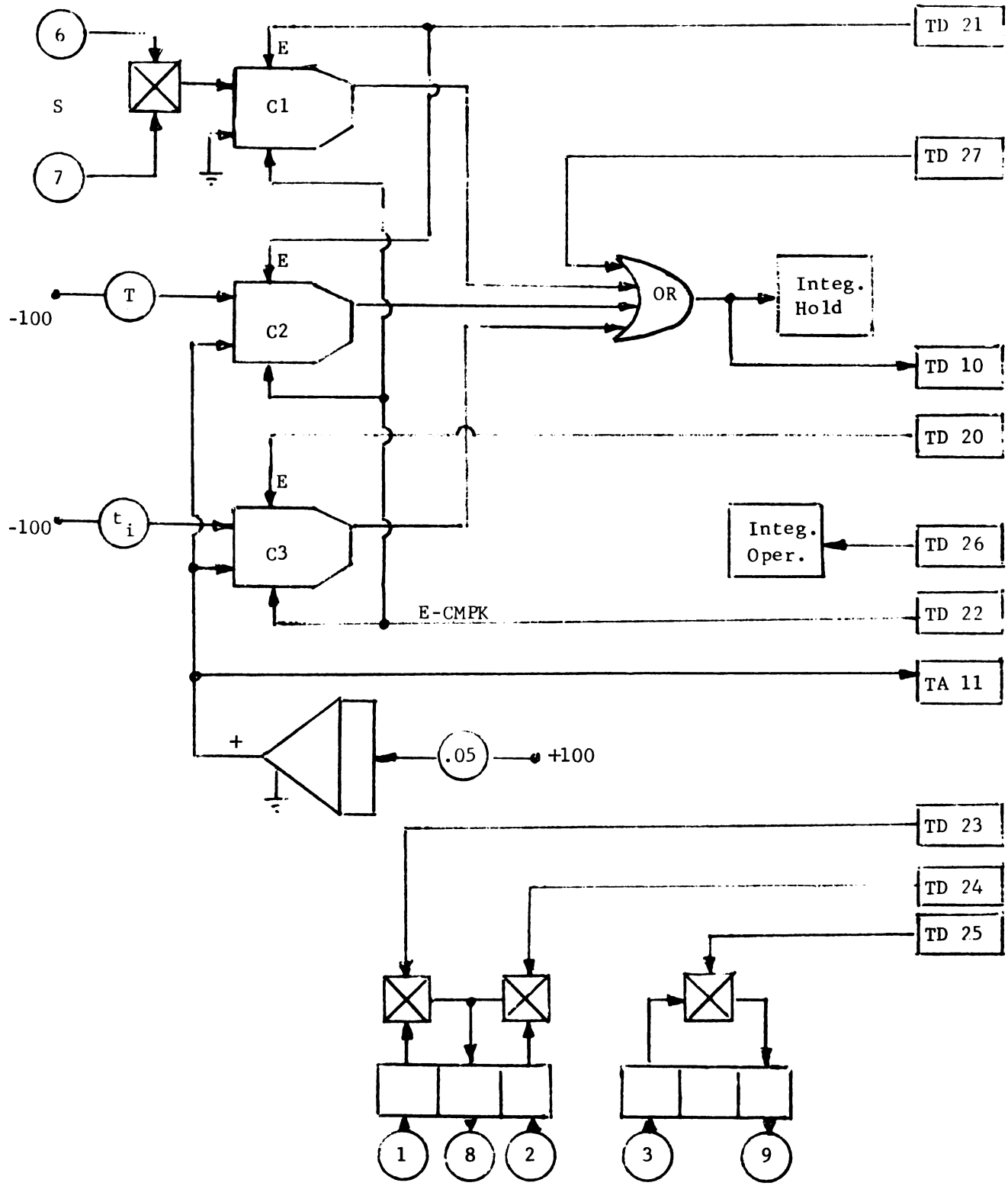


Figure 4-3(c)

Analog Logic Control

The sensitivity analysis required to obtain the S matrix, equation (34) in Chapter III, is performed by the three integrators in Figure 4-3(b). This configuration is essentially a duplicate of the system solution configuration with the exception that the system is forced, i.e.,

$$\begin{bmatrix} \frac{\partial x_1}{\partial P_j} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial x_n}{\partial P_j} \end{bmatrix} = -\epsilon^{Gt_i} \int_{t_o}^{t_i} \epsilon^{-G\tau} B x_j(\tau) d\tau$$

$$B = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} \quad ; \quad \begin{bmatrix} \frac{\partial x_1(t_o)}{\partial P_j} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial x_n(t_o)}{\partial P_j} \end{bmatrix} = \tilde{0}$$

The analog signals corresponding to $\frac{\partial x_1}{\partial P_j}$, $\frac{\partial x_2}{\partial P_j}$ and $\frac{\partial x_3}{\partial P_j}$ are transferred to the digital computer via analog trunk lines TS 12, TA 13, and TA 14, respectively. An additional amplifier was found useful when small signal levels were encountered, particularily for the constrained variable x_3. Compensation for this "scaling" was made in the digital program.

The signal sensing and timing functions were performed by comparators C1, C2 and C3, an OR gate and an integrator shown in Figure 4-3(c). The basic function is to determine the maximum

amplitude of each constrained state variable in the time interval $[t_o,T]$ and the corresponding time t_i. The final time T is pre-selected to be greater than the largest time constant in the system.

The first three peaks of each constrained state variable are observed by sensing the derivative of the variable as it goes through zero. In Figure 4-3(c), this function is performed by comparator C1 for x_3. The procedure is as follows: switch S is set so that with the given initial conditions the first peak of x_3 is sensed and its value transferred to the digital computer via analog trunk line TA 10; simultaneously, the time at which this occurs is observed by monitoring analog trunk line TA 11; these values are stored and switch S is changed to the opposite polarity signal and the procedure repeated twice, each time storing x_3 and t_i. The digital computer program then evaluates these signals and chooses the pair $[x_3,t_i]$ corresponding to the largest magnitude of x_3 which is then used in computing the C-Function and in the sensitivity analysis. Ordinarily, switching of switch S would be under the control of the digital program, but equipment limitations necessitated manual control.

Comparator C2 controls the total integrator running time which is pre-selected by potentiometer T. Both C1 and C2 are enabled simultaneously by a logic signal transferred from the digital computer via digital trunk line TD 21.

Potentiometer t_i is servo-set and comparator C3 then controls the integration interval of the sensitivity circuit. All three comparators and a logic signal from the digital computer via digital trunk line TD 27 feed into an OR gate so that any one

of these signals places all integrators into the "Hold" mode. Simultaneously, the digital program is signalled of a "Hold" condition by digital trunk line TD 10. Comparator C3 is enabled by digital trunk line TD 20 from the digital computer. All three comparators have a common initialization, E-CMPK, controlled by digital trunk line TD 22.

Digital trunk lines TD 23, TD 24, and TD 25 switch the forcing inputs required for the sensitivity analysis. Digital trunk line TD 26 controls the "operate" mode of all integrators, and a logic "0" on both TD 26 and TD 27 is used for the "IC" mode.

Summarizing, for each iteration on $\tilde{p}$ the system $\dot{x} = Gx$, Figure 4-3(a), is run once for each of the ℓ constrained variables to obtain the maximum amplitudes and the corresponding t_i's. Then for each t_i the sensitivity computation, Figure 4-3(b), is run n times. This corresponds to $\ell(1+n)$ total analog runs. In the case of the third order system with a single constrained state variable, a total of four analog runs are required for each iteration.

4.2.3 C Computation.

This digital computation is straightforward, using

$$C = \sum_{i \in \ell} [x_i(\max) - x_i(t_i)][x_i(t_i) - x_i(\min)]H(h_i) \ .$$

4.2.4 $\tilde{p}$ Estimation.

Both the Newton-Raphson and the gradient iteration schemes are incorporated in the digital program. Initially, the search for a parameter solution utilizes the modified Newton-Raphson

algorithm

$$\tilde{p}(l+1) = \tilde{p}(i) - \gamma \frac{\nabla_p C(i)}{\|\nabla_p C(i)\|^2}$$

The step size coefficient γ was found very useful in preventing oscillation, particularily near $C = 0$ because of the accuracy limitations imposed by the analog computer. In the program, $C(i+1)$ is compared to $C(i)$ and if it is smaller, the Newton-Raphson scheme with $\gamma = 1$ is continued; but if $C(i+1)$ is larger, the smaller $C(i)$ is stored while γ is adjusted by successive halving until either $C(i+1)$ becomes smaller than $C(i)$ or $\gamma < 0.004$. The latter limit was determined by experiment to be consistent with the accuracy limitations of the analog computer. At all times the smallest value of C and the corresponding $\tilde{p}$ vector are stored.

To insure safeguards in the event $\nabla_p C \to 0$, the search automatically switches into the gradient procedure

$$\tilde{p}(i+1) = \tilde{p}(i) \pm \alpha \frac{\nabla_p C(i)}{\|\nabla_p C(i)\|}$$

every time γ becomes less than 0.004. The step size coefficient α is determined by successive halving as in the case of γ. When α becomes less than 0.004, the K matrix is computed for parameter vector $\tilde{p}$ corresponding to the solution closest to $C = 0$. The vector K_N obtained from the bottom row of the K matrix yields the desired feedback coefficients.

In the event that $\nabla_p C \to 0$ faster than $C \to 0$, then it is possible that the procedure will not continue to the solution

at $C = 0$. In this case, after a finite number of iterations, the program automatically switches to the gradient method; and the extremum is located in terms of the $\tilde{p}_e$ vector. The user can then choose a new starting $\tilde{p}$ vector sufficiently far away from $\tilde{p}_e$ and in the direction of $C = 0$ and continue with a new search using the C-Function Algorithm.

CHAPTER V

EXAMPLES

A problem of current interest is that of automatic spacing of vehicles in a single lane, i.e., longitudinal control of a vehicle [ATH-3], [FEN-1], [MIK-1]. Assuming a given desired spacing between two vehicles, the objective is to design an optimal control that would accelerate or decelerate the following vehicle to maintain the desired headway distance and velocity relative to the lead vehicle. The vehicles chosen for this example are assumed to be conventional automobiles.

Blackwell [BLA-1] has shown that the longitudional dynamics of a typical automobile may be quite accurately approximated by a simple transfer function of the form

$$\frac{V(s)}{e(s)} = \frac{k/T}{s + a} \tag{1}$$

where s is the usual Laplace complex variable

V is the vehicle velocity

e is the throttle position

$k = \dfrac{k_1}{k_2 + k_3}$

k_1 is the ratio of rear wheel traction force to throttle position

k_2 is the ratio of rear wheel traction force to vehicle speed

k_3 is the ratio of road load to vehicle speed

T = time constant $= \dfrac{M_e}{k_2 + k_3}$

$a = 1/T$

M_e = effective mass of the vehicle

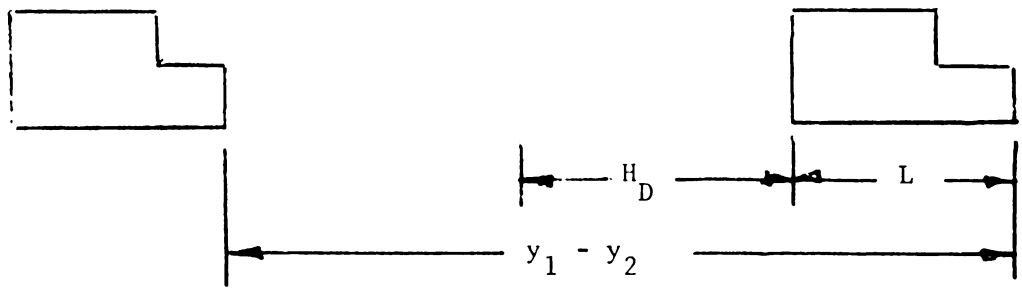
Actually, equation (1) represents the small-signal transfer function since k_1, k_2, k_3 and M_e are not constants over the whole operating range of the vehicle. For example, at 40 mph k may have a value of 240 sec^{-1} and at 60 mph k may be reduced to 165 sec^{-1}. For this example nominal values were chosen

$$k = 200 \text{ sec}^{-1}$$

$$T = 20 \text{ sec}$$

and

$$V(s) = \frac{10}{s + 0.05} \tag{2}$$



y_2 $\qquad$ y_1

$\dot{y}_2$ $\qquad$ $\dot{y}_1$

$\ddot{y}_2$ $\qquad$ $\ddot{y}_1$

Figure 5-1

Vehicle Longitudinol Spacing Problem

Figure 5-1 is a schematic representation of the problem where y, $\dot{y}$, and $\ddot{y}$ are the position, velocity and acceleration,

respectively, of each vehicle. The distance H_D is the desired spacing between the two vehicles. There is justification based on traffic studies [TRE-1] that drivers tend to choose a headway corresponding to the relation

$$H_D = c_o + 2\tau\dot{y}_1 \tag{3}$$

where c_o is a constant and τ is the reaction time of the driver. The actual choice of H_D for an automatic system would maximize traffic flow consistent with the driver "psychological" comfort.

Assuming that the lead vehicle is moving at a constant velocity, equation (2) is transformed into the regulator form in state space by defining the state variables

$$x_1 = y_1 - y_2 - H_D - L$$

$$\dot{x}_1 = y_1 - y_2 = x_2$$

$$\dot{x}_2 = y_1 - y_2 = -y_2 = x_3$$

$$\dot{x}_3 = -y_2 = -ax_3 - b\dot{e}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -0.05 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -10 \end{bmatrix} \dot{e} \tag{4}$$

Equation (4) is in the form compatible with the algorithms described in Chapters III and IV such that constraints on the vehicle acceleration can be invoked:

$$10.0 = x_3(\max) \geq x_3(t_3) \geq x_3(\min) = -10.0$$

The autonomous form of equation (4) becomes

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -p_1 & -p_2 & -p_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

where $p_1 = 100k_{31}$

$p_2 = 100k_{32}$

$p_3 = 0.05 + 100k_{33}$

5.1 Example 1.

Assuming initial conditions

$$\begin{bmatrix} x_1(t_o) \\ x_2(t_o) \\ x_3(t_o) \end{bmatrix} = \begin{bmatrix} 450 \\ -60 \\ 0 \end{bmatrix}$$

minimum values of k_{31}, k_{32}, and k_{33} determined by the Routh Criterion

$$\min k_{31} = 0.0001$$

$$\min k_{32} = 0.0006$$

$$\min k_{33} = -0.0004$$

and maximum values based on a preliminary exploration using the Minimum Cost Algorithm and the C-Function Algorithm

$$\max k_{31} = 0.0226$$

$$\max k_{32} = 0.0715$$

$$\max k_{33} = 0.0488$$

Table I summarizes the results of the search for an absolute minimum of J using the Minimum Cost Algorithm. It is to be noted that after the fourth contraction of the search region, the neighborhood of the absolute minimum was located, with the following lower bounds:

$$k_{31} \geq 0.0001$$

$$k_{32} > 0.0029 - 0.1(0.0029 - 0.0006) = 0.00267$$

$$k_{33} > 0.0083 - 0.1(0.0083 + 0.0004) = 0.00743$$

and parameter values

$$p_1 = 0.01$$

$$p_2 = 0.29$$

$$p_3 = 0.88$$

TABLE I

Search for Absolute Minimum

Step	k_{31}	k_{32}	k_{33}	J
Initial	0.0226	0.0715	0.0488	13,956.9
1	0.0001	0.0170	0.0192	72.1
2	0.0001	0.0088	0.0133	26.9
3	0.0001	0.0046	0.0105	10.6
4	0.0001	0.0029	0.0083	5.2
5	0.0001	0.0029	0.0083	5.2

The parameter values were then used as the input to the C-Function program to check for constraint violations. Since

$$\max |x_3(t_3)| = 10.75$$

as determined by the analog run, the constraint on acceleration was violated and the search with the C-Function Algorithm provided new parameter solutions as shown in Figure 5-2 and Table II. Accuracy limitations of the analog computer resulted in the closest solution

$$\max |x_3(t_3)| = 9.75$$

K Matrix

$$\begin{bmatrix} 0.000039976 & 0.00012159 & 0.00013827 \\ 0.00012159 & 0.0037216 & 0.0028910 \\ 0.00013827 & 0.0028910 & 0.0082936 \end{bmatrix}$$

$$J = 7.46$$

Q Matrix

$$\begin{bmatrix} 0.0000019 & 0 & 0 \\ 0 & 0.00059 & 0 \\ 0 & 0 & 0.0019 \end{bmatrix}$$

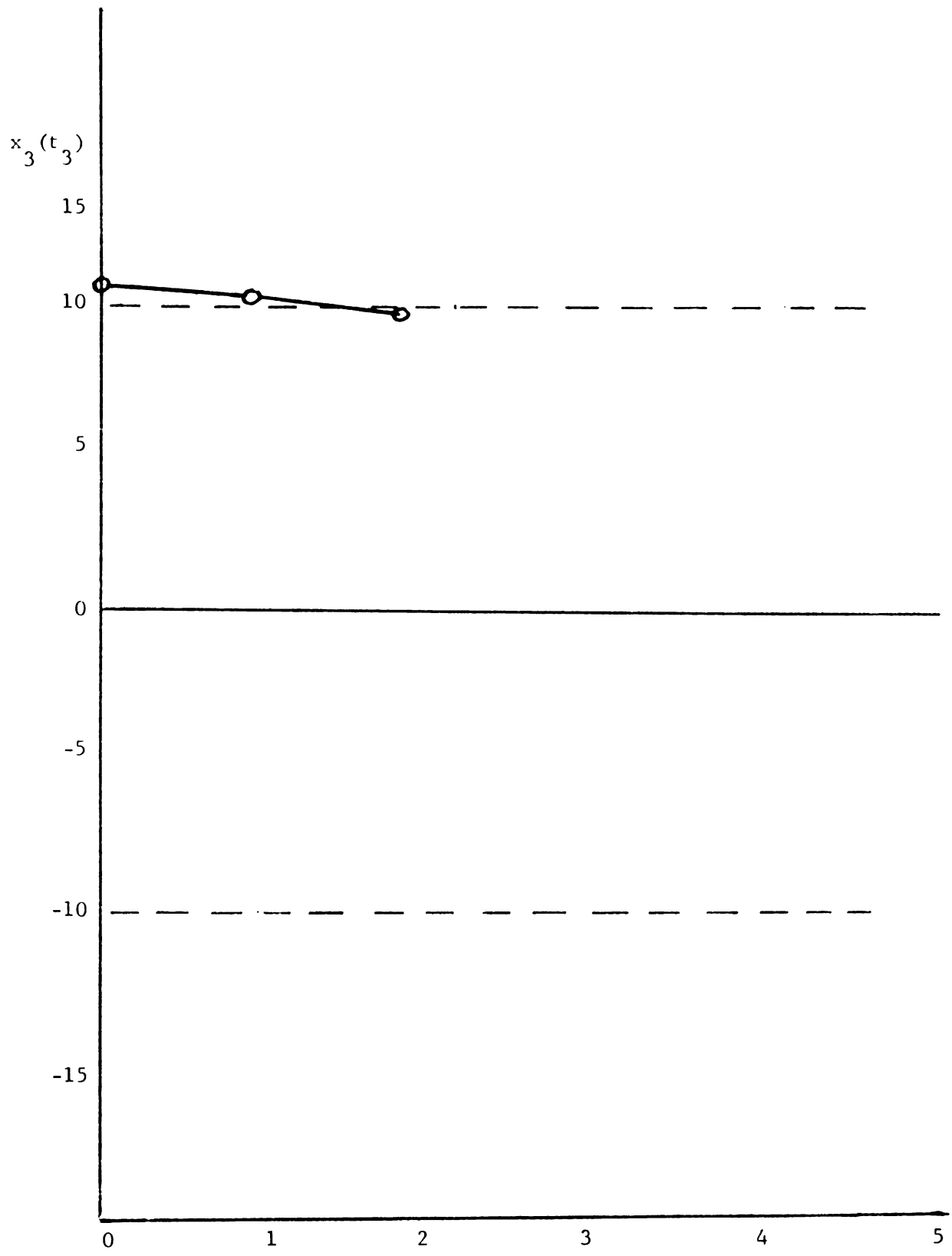


Figure 5-2

C-Function Iteration - Example 1

TABLE II

Results of C-Function Search

	Iteration		
	1	2	3
p_1	0.01	0.0136	0.0138
p_2	0.29	0.2891	0.2891
p_3	0.88	0.8794	0.8794
$x_3(t_3)$	10.75	10.25	9.75
t_3	2.81	2.40	2.41
C	-0.156	-0.051	0.049
$\frac{\partial x_3}{\partial p_1}$	-4.75	-4.03	-4.83
$\frac{\partial x_3}{\partial p_2}$	1.08	0.78	0.23
$\frac{\partial x_3}{\partial p_3}$	8.03	-0.08	-0.08
$\frac{\partial C}{\partial p_1}$	9.84	8.25	9.41
$\frac{\partial C}{\partial p_2}$	-2.31	-1.59	-0.44
$\frac{\partial C}{\partial p_3}$	-1.73	0.0154	-0.0146

The actual steps required within the C-Function Algorithm to obtain the results in Figure 5-2 are graphed in Figure 5-3. The circled points represent the successive acceptable estimates while the rest of the points demonstrate adjustments required in the γ and α step size coefficients to obtain rapid damping

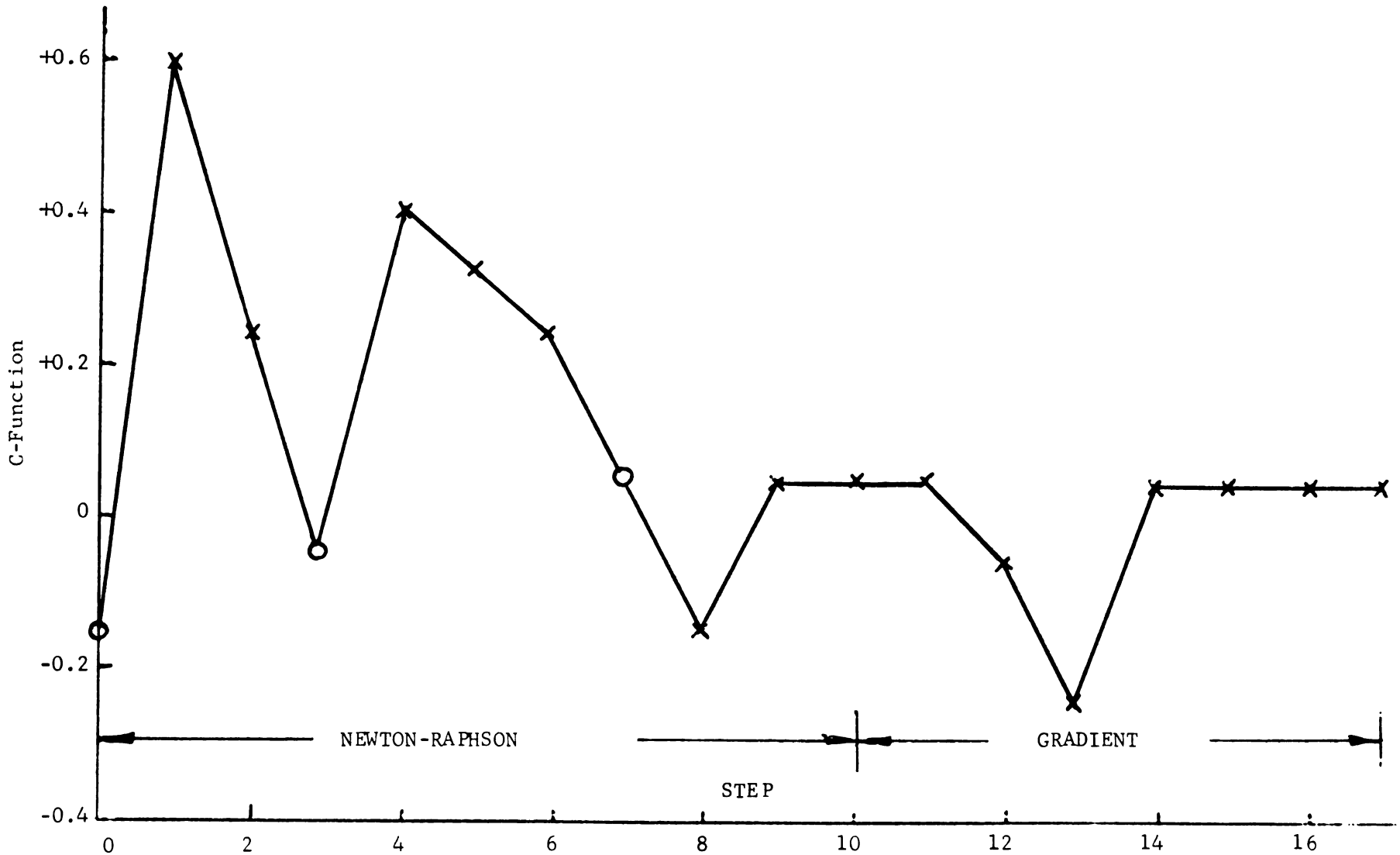


Figure 5-3

Step-Size Histroy of C-Function Solution - Example 1

of the oscillation as $C \rightarrow 0$.

Since a solution at $C = 0$ does not necessarily imply the final solution, a neighborhood was chosen about the point

$$k_{31} = 0.00013827$$

$$k_{32} = 0.0028910$$

$$k_{33} = 0.0082936$$

such that

$$\max K_N = [0.00013827, 0.0147, 0.0192]$$

$$\min K_N = [0.0001, 0.0006, -0.004]$$

With this input to the Minimum Cost Algorithm a new minimum was located at

$$k_{31} = 0.0001$$

$$k_{32} = 0.0020$$

$$k_{33} = 0.0133$$

$$J = 2.5$$

This K_N vector yielded the following parameter values:

$$p_1 = 0.01$$

$$p_2 = 0.20$$

$$p_3 = 1.38$$

where were used as input to the C-Function Algorithm to check constraints. The results were as follows:

$$\max |x_3(t_3)| = 5.25 \text{ ft/sec}^2$$

$$t_3 = 3.38 \text{ sec}$$

K Matrix

$$\begin{bmatrix} 0.00002 & 0.000138 & 0.0001 \\ 0.000138 & 0.002622 & 0.0020 \\ 0.0001 & 0.0020 & 0.0133 \end{bmatrix}$$

Q Matrix

$$\begin{bmatrix} 0.000001 & 0 & 0 \\ 0 & 0.000124 & 0 \\ 0 & 0 & 0.01512 \end{bmatrix}$$

Since the constraints are satisfied, this is the desired solution within the limits of resolution in the search region.

5.2 Example 2.

The possibility of discontinuity in p-space was discussed in Chapter III. This example is typical of the results obtained when the largest amplitude of $x_3(t)$ shifted from the first peak to the second. Figure 5-4 demonstrates the behavior of the C-Function and $x_3(t_3)$ under the following conditions:

$$5.0 \geq x_3(t) \geq -5.0$$

$$\begin{bmatrix} x_1(t_o) \\ x_2(t_o) \\ x_3(t_o) \end{bmatrix} = \begin{bmatrix} 450 \\ -60 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} p_1(0) \\ p_2(0) \\ p_3(0) \end{bmatrix} = \begin{bmatrix} 9.99\underline{6} \\ 18.26 \\ 12.68 \end{bmatrix}$$

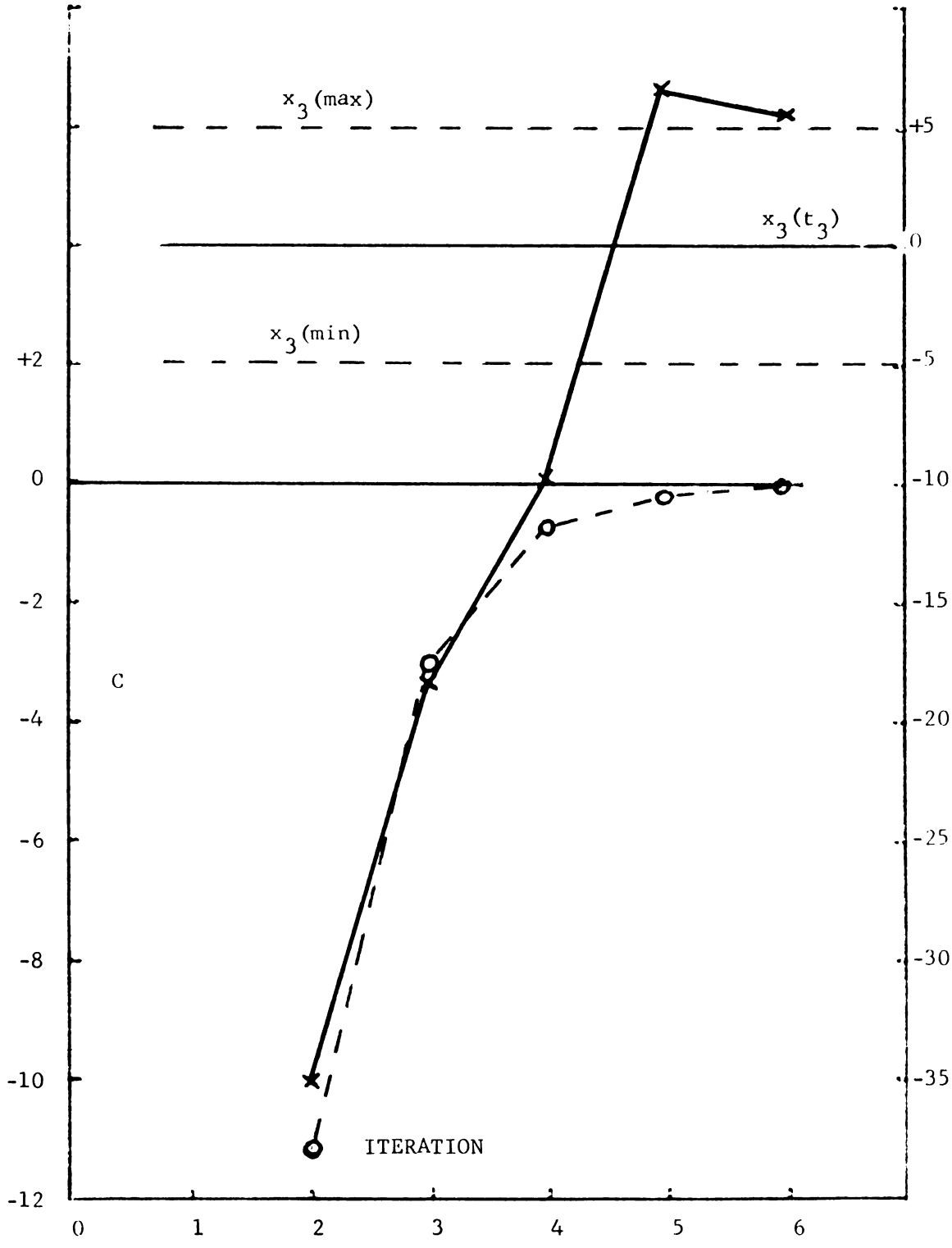


Figure 5-4

Solution Stability under Peak Switching - Example 2

for the system shown in equation (4). The initial values and the first iteration values were extremely large and are not shown in the figure. There was no oscillation of the C-Function as the dominant peaks changed at the fifth iteration, and final convergence, within the limits of analog accuracy, was obtained on the sixth iteration. Table III summarizes the values at each step in the computation.

TABLE III

C-Function Behavior Under Peak Switching

	Iteration				
	2	3	4	5	6
p_1	3.577	2.997	2.713	2.5913	2.2616
p_2	17.60	17.663	17.694	17.708	17.805
p_3	12.18	12.218	12.23	12.229	12.235
$x_3(t_3)$	-35.75	-18.25	-9.75	6.75	5.75
t_3	0.18	0.17	0.16	2.56	1.83
C	-11.25	-3.08	-0.70	-0.21	-0.08
$\frac{\partial x_3}{\partial p_1}$	-2.97	-2.93	-2.93	0.43	0.18
$\frac{\partial x_3}{\partial p_2}$	0.32	0.32	0.32	-0.13	-0.13
$\frac{\partial x_3}{\partial p_3}$	1.97	0.92	0.48	-0.08	-0.08
$\frac{\partial C}{\partial p_1}$	-21.27	-10.68	-5.70	-0.57	-0.20
$\frac{\partial C}{\partial p_2}$	2.32	1.19	0.63	0.17	0.14
$\frac{\partial C}{\partial p_3}$	1.41	0.34	0.09	0.01	0.009

K Matrix

$$\begin{bmatrix} 0.051148 & 0.25153 & 0.022616 \\ 0.35153 & 2.82691 & 0.17805 \\ 0.022616 & 0.17805 & 0.12185 \end{bmatrix}$$

$J = 23,241.5$

Q Matrix

$$\begin{bmatrix} 0.051148 & 0 & 0 \\ 0 & 2.46712 & 0 \\ 0 & 0 & 1.14083 \end{bmatrix}$$

CHAPTER VI

SUMMARY AND CONCLUSIONS

An algorithm has been developed resulting in a computational procedure which determines the optimal scalar control for a linear, time-invariant regulator with state variable constraints and a quadratic performance index. The basic contribution which makes possible a practicable engineering solution to this problem is the development of a simple algebraic algorithm for the determination of the gain matrix K. It has been shown that the K-matrix can be obtained from an n-parameter optimization of the closed loop system. The need to solve the matrix Riccati equation is eliminated, and, therefore, there are no computational stability difficulties which impose a practical limit on the order of the system.

The K-matrix algorithm also provides a criterion for the positive definiteness of a diagonal Q weighting matrix which is used as part of the search procedure to obtain a minimum cost solution. An efficient digital computer program allows only the positive definite constant K matrices which represent stable control laws and define positive definite Q matrices to be considered in the minimization of the quadratic performance index

$$J = 1/2x^{T}(t_{o})Kx(t_{o}).$$

The result of this minimization is then evaluated for state variable constraint violations with the aid of a constraint function specifically developed for this purpose. This constraint function is an integral part of an iterative procedure that locates the K matrix nearest the minimum cost solution and which also satisfies the constraints. The new value of K is then the basis for the next estimate in the cost minimization program. Successive iterations produce the desired K matrix.

An example of the application of this algorithm to a third order system is presented, showing the results obtained in defining the optimal feedback coefficients for vehicle spacing control with acceleration constraints.

Since the K matrix contains all the necessary engineering design information, the procedure described in this thesis should have greater application than any of the currently available methods.

There are obvious extensions to this work which should be considered. It would be desirable to investigate the possibility of developing a corresponding K matrix algorithm for the general case of a vector control u(t). Additional research needs to be done on the sensitivity of the cost function J to state variable initial conditions. For example, it would be desirable to obtain a solution for minimum J such that J has minimum sensitivity to the initial conditions. Another possible extension of these results is to the work of Kleinman, Fortmann, Athans [KLE-2] where a time varying matrix K(t), representing a finite time cost interval, is approximated by a series of piecewise-constant K matrices. Finally, it would be of practical interest to

investigate the computational limits imposed by the errors inherent in an analog computer. For example, in the course of this research it was found that potentiometer setting tolerances imposed a limit on the accuracy with which the constraint function could be obtained in the vicinity of $C = 0$. Results of such work would show the conditions under which it would be advisable to consider an all-digital configuration.

REFERENCES

REFERENCES

[ATH-1] : Athans, M., and Falb, P., "Optimal Control: An Introduction to the Theory and Its Applications", McGraw-Hill Book Company, 1966.

[ATH-2] : Athans, M., and Levine, W.S., "On the Numerical Solution of the Matrix Riccati Differential Equation using a Runge-Kutta Scheme", Report ESL-R-276, Electronic Systems Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., July, 1966.

[ATH-3] : Athans, M., Levine, W.S., Levis, A.H., "On the Optimal and Suboptimal Position and Velocity Control of a String of High-Speed Moving Trains", Report ESL-R-291, Electronic Systems Laboratory, M.I.T., Cambridge, Mass., November, 1966.

[BEK-1] : Bekey, G.A., Karplus, W.J., "Hybrid Computation", John Wiley and Sons, Inc., New York, 1968.

[BEL-1] : Bellman, R.E., "Stability Theory of Differential Equations", McGraw-Hill Book Company, Inc., New York, 1953.

[BER-1] : Berkovitz, L.D., "On Control Problems with Bounded State Variables", J. Math. Anal. Appl., vol. 5, 1962.

[BLA-1] : Blackburn, T.R., "Solution of the Algebraic Matrix Riccati Equation via Newton-Raphson Iteration", pp. 940-945, JACC, 1968.

[BLA-2] : Blackwell, L.M., "A Study of the Adaptivility of the Automobile to Automatic Control", Report EES-276A-3, Communications and Control Systems Laboratory, Ohio State University, Columbus, Ohio, February, 1967.

[CES-1] : Cesari, L., "Asymptotic Behavior and Stability Problems in Ordinary Differential Equations", Berlin, Springer, 1963.

[CHE-1] : Chen, R.T.N., Shen, D.W.C., "Sensitivity Analysis and Design of Multivariate Regulators Using a Quadratic Performance Criterion", pp. 229-237, JACC, 1968.

[FEN-1] : Fenton, R.E., Cosgriff, R.L., Olson, K.W., Blackwell, L.M., "One Approach to Highway Automation", Proceedings of the IEEE, Vol. 56, No. 4, pp. 556-566, April, 1968.

[HAY-1] : Hayes, R.M., Jr., "An Algorithm for the Determination of the Definiteness of a Real Square Matrix", IEEE Transactions on Automatic Control, vol. 13, no. 2, pp. 219-220, April, 1968.

[KAL-1] : Kalman, R.E., "Mathematical Description of Linear Dynamical Systems", J. SIAM on Control, ser. A, vol. 1, pp. 152-192, 1963.

[KAL-2] : Kalman, R.E., "When is a Linear Control System Optimal?", pp. 1-15, JACC, 1963.

[KAL-3] : Kalman, R.E., Bertram, J.E., "Control System Analysis and Design Via the Second Method of Lyapunov", J. Basic Eng., vol. 82, pp. 371-393, 1960.

[KEL-1] : Kelley, H.J., "Method of Gradients", G. Leitman, ed., Optimization Techniques, Chapter 6, Academic Press, New York, 1962.

[KLE-1] : Kleinman, D.L., "On the Iterative Technique for Riccati Equation Computations", IEEE Transactions on Automatic Control, vol. 13, no. 1, p. 114, February, 1968.

[KLE-2] : Kleinman, D.L., Fortmann, T., Athans, M., "On the Design of Linear Systems with Piecewise-Constant Feedback Gains", IEEE Transactions on Automatic Control, vol. 13, no. 4, pp. 354-361, August, 1968.

[KOE-1] : Koenig, H.E., Tokad, Y., Kesavan, H.K., "Analysis of Discrete Physical Systems", McGraw-Hill Book Company, New York, 1967.

[KRE-1] : Kreindler, E., Sarachik, P.E., "On the Concepts of Controllability and Observability of Linear Systems", IEEE Transactions on Automatic Control, vol. 9, no. 2, pp. 129-136, April, 1964.

[LET-1] : Letov, A.M., "Analytic Controller Design II", Automation and Remote Control, vol. 21, pp. 561-568, May, 1960.

[McG-1] : McGill, R., "Optimal Control, Inequality State Constraints and the Generalized Newton-Raphson Algorithm", J. SIAM on Control, vol. 3, pp. 291-298, 1965.

[MIK-1] : Mika, H.S., "Optimal Control for Automatic Vehicle Longitudinal Guidance", Report 1206-1, Ford Motor Company, September, 1969.

[MOR-1] : Morgan, B.S., Jr., "Sensitivity Analysis and Synthesis of Multivariable Systems", IEEE Transactions on Automatic Control, vol. II, no. 3, pp. 506-512, July, 1966.

[PER-1] : Perkins, W.R., Cruz, J.B., Jr., "Engineering of Dynamic Systems", John Wiley and Sons, Inc., New York, 1969.

[PON-1] : Pontryagin, L.S., Boltyanskii, V.G., Gamkrelidze, R.V., Mishchenko, E.F., "The Mathematical Theory of Optimal Processes", Interscience Publishers, New York, 1962.

[REK-1] : Rekasins, S.V., Hsia, T.C., "On an Inverse Problem in Optimal Control", IEEE Transactions on Automatic Control, vol. 9, no. 4, pp. 370-375, October, 1964.

[SAG-1] : Sage, A.P., "Optimum Systems Control", Prentice-Hall, Inc., 1968.

[TOM-1] : Tomovic, R., "Sensitivity Analysis of Dynamic Systems", McGraw-Hill Book Company, Inc., New York, 1963.

[TRE-1] : Treiterer, J., "Improvement of Traffic Flow and Safety by Longitudional Control", Transportation Research, vol. 1, pp. 231-251, Pergamon Press, Great Britain, 1967.

[TUR-1] : Turnbull, H.W., "Theory of Equations", Interscience Publishers, New York, 1957.

[TYL-1] : Tyler, J.S., Tuteur, F.B., "The Use of a Quadratic Performance Index to Design Multivariable Control Systems", IEEE Transactions on Automatic Control, vol. II, no. 1, pp. 84-92, January, 1966.

[VAL-1] : Valentine, F.A., "The Problem of Lagrange with Differential Inequalities as Added Side Conditions", Contributions to the Calculus of Variations 1933-37, University of Chicago Press.

[Van-1] : Van Ness, J.E., Boyle, J.M., Imad, F.P., "Sensitivities of Large Multiple-Loop Control Systems", IEEE Transactions on Automatic Control, vol. 10, no. 3, pp. 308-315, July, 1965.

[WAN-1] : Wang, S.J., "A Gradient Computational Technique for a Class of Optimal Control Problems Subject to Inequality Constraints", Ph.D. Dissertation, Michigan State University, 1969.

[WON-1] : Wonham, W.M., Johnson, C.D., "Optimal Bang-Bang Control with Quadratic Performance Index", pp. 101-112, JACC, 1963.

APPENDIX A

SAMPLE MINIMUM COST ALGORITHM PROGRAM

```
// JOB
// FOR MINK
*LIST ALL
*NONPROCESS PROGRAM
*DEF WORD INTEGERS
*IOCS (CARD,1443 PRINTER,DISK)
      REAL K(3,3)
      DIMENSION A(3),XO(3)
      READ(2,500)L,D,(A(I),I=1,3),B
  500 FORMAT(I2,5F10.3)
      READ(2,501)K(3,1),K(3,2),K(3,3)
      READ(2,501)XO(1),XO(2),XO(3)
  501 FORMAT(3F10.4)
      READ(2,506)RK3MX,RK2MX,RK1MX,RK3MN,RK2MN,RK1MN
  506 FORMAT(6F10.4)
      M=1
      J=1
      C=1.0
      DK1=0.1*(RK1MX-RK1MN)
      DK2=0.1*(RK2MX-RK2MN)
      DK3=0.1*(RK3MX-RK3MN)
      K(3,3)=RK3MX
   99 K(3,2)=RK2MX
   94 K(3,1)=((A(2)+(B**2)*K(3,2))*(A(3)+(B**2)*K(3,3))-A(2))/B**2-0.0001
      IF(K(3,1)-RK1MX)71,71,72
   72 K(3,1)=RK1MX
   71 K(1,1)=A(1)*K(3,2)+A(2)*K(3,1)+(B**2)*K(3,2)*K(3,1)
      K(1,2)=A(1)*K(3,3)+A(3)*K(3,1)+(B**2)*K(3,1)*K(3,3)
      K(2,2)=-K(3,1)+A(2)*K(3,3)+A(3)*K(3,2)+(B**2)*K(3,2)*K(3,3)
      VO=K(1,1)*(XO(1)**2)+K(2,2)*(XO(2)**2)+K(3,3)*(XO(3)**2)+2.0*
     1(K(3,1)*XO(1)*XO(3)+K(3,2)*XO(2)*XO(3)+K(1,2)*XO(1)*XO(2))
```

```
      VN=0.5*VN
    1 Q1=2.0*A(1)*K(3,1)+(B**2)*K(3,1)**2
      IF(Q1)73,73,7
   73 WRITE(3,508)
  508 FORMAT(13H CHANGE RK1MX)
      GO TO 80
    7 Q3=-2.0*K(3,2)+2.0*A(3)*K(3,3)+(B**2)*K(3,3)**2
      IF(Q3)75,75,6
   75 K(3,2)=K(3,2)-DK2
      IF(K(3,2)-RK2MN)63,7,7
   63 K(3,3)=K(3,3)-DK3
      IF(K(3,3)-RK3MN)80,40,40
    6 Q2=-2.0*K(1,2)+2.0*A(2)*K(3,2)+(B**2)*K(3,2)**2
      IF(Q2)74,74,8
   74 K(3,1)=K(3,1)-DK1
      IF(K(3,1)-RK1MN)16,17,17
   17 K(1,2)=A(1)*K(3,3)+A(3)*K(3,1)+(B**2)*K(3,1)*K(3,3)
      GO TO 6
   16 K(3,2)=K(3,2)-DK2
      IF(K(3,2)-RK2MN)63,54,54
    8 P3=A(3)+(B**2)*K(3,3)
      IF(P3)76,76,9
   76 WRITE(3,509)
  509 FORMAT(13H CHANGE RK3MN)
      GO TO 80
    9 P2=A(2)+(B**2)*K(3,2)
   18 P1=A(1)+(B**2)*K(3,1)
      R3=(P2*P3-P1)/P3
      IF(R3)54,54,10
   10 IF(P1)77,77,82
   77 WRITE(3,510)
  510 FORMAT(13H CHANGE RK1MN)
      GO TO 80
   82 IF(L-2)5,85,85
    5 VN=K(1,1)*(XO(1)**2)+K(2,2)*(XO(2)**2)+K(3,3)*(XO(3)**2)+2.0*
     1(K(3,1)*XO(1)*XO(3)+K(3,2)*XO(2)*XO(3)+K(1,2)*XO(1)*XO(2))
      VN=0.5*VN
   15 PAR1=(A(2)+(B**2)*K(3,2))*(XO(1)**2)-XO(2)**2+2.0*XO(1)*XO(3)
     1+2.0*A(3)*XO(1)*XO(2)
   35 PAR2=(A(1)+(B**2)*K(3,1))*(XO(1)**2)+(A(3)+(B**2)*K(3,3))*(XO(2)**
     12)+2.0*XO(2)*XO(3)
      PAR3=(A(2)+(B**2)*K(3,2))*(XO(2)**2)+XO(3)**2+2.0*(A(1)+(B**2)*
     1K(3,1))*XO(1)*XO(2)
      SDEL=SQRT(PAR1**2+PAR2**2+PAR3**2)
   36 COR1=C*PAR1/SDEL
      COR2=C*PAR2/SDEL
      COR3=C*PAR3/SDEL
      IF(ABS(COR1)-ABS(K(3,1)))36,32,32
   36 IF(ABS(COR2)-ABS(K(3,2)))37,32,32
   37 IF(ABS(COR3)-ABS(K(3,3)))3,32,32
   32 C=0.5*C
      GO TO 38
    3 K(3,1)=K(3,1)-C*PAR1/SDEL
      IF(K(3,1)-RK1MN)78,70,70
   78 K(3,1)=RK1MN
   70 K(1,1)=A(1)*K(3,2)+A(2)*K(3,1)+(B**2)*K(3,2)*K(3,1)
      K(1,2)=A(1)*K(3,3)+A(3)*K(3,1)+(B**2)*K(3,1)*K(3,3)
      K(2,2)=-K(3,1)+A(2)*K(3,3)+A(3)*K(3,2)+(B**2)*K(3,2)*K(3,3)
      VN=VN
```

```
      VN=K(1,1)*(X0(1)**2)+K(2,2)*(X0(2)**2)+K(3,3)*(X0(3)**2)+2.0*
     1(K(3,1)*X0(1)*X0(3)+K(3,2)*X0(2)*X0(3)+K(1,2)*X0(1)*X0(2))
      VN=0.5*VN
      IF(VN-V0)81,79,79
   79 K(3,1)=K(3,1)+C*PAR1/SDEL
    2 C=0.5*C
      J=J+1
      IF(J-20)3,3,100
   81 L=L+1
      GO TO 1
   85 WRITE(3,502)K(3,1),K(3,2),K(3,3),VN
  502 FORMAT(3F10.4,F20.1)
   11 IF(ABS(VN-V0)-1.0001)12,12,14
   14 J=1
      N=N+1
      IF(N-10)15,15,61
   61 WRITE(3,507)
  507 FORMAT(2H N)
   12 K(3,2)=K(3,2)-DK2
      C=1.0
   68 J=1
      M=1
      L=1
      IF(K(3,2)-RK2MN)53,54,54
   53 K(3,3)=K(3,3)-DK3
   65 IF(K(3,3)-RK3MN)80,40,40
  100 WRITE(3,503)
  503 FORMAT(7H J LOOP)
      GO TO 12
   80 CALL EXIT
      END
```

APPENDIX B

C-FUNCTION ALGORITHM PROGRAM

```
R04 READY READER
TASK 1800 TSX V3M4.4

// JOB
// FOR SIMA
*LIST ALL
*STOREPROCESS PROGRAM
*ONE WORD INTEGERS
*IOCS (CARD,1443 PRINTER,DISKT
      DIMENSION R(6,6),RK(5,5),S(5,5),TC(5,1),TR(1,5),PAR(5),F(5),A(5),
     1X0(5),XMAX(5),XMIN(5),PT(5),X(5),T(5),F(5),L(5),JP(5),PP(5),AR(5),
     2TA(3),PPP(5),PCOR(5),ESTP(5)
      CALL PYENT
      CALL HY1
      READ (2,100)N,TF,(A(I),I=1,N),B
      READ (2,200)(P(I),I=1,N)
      READ (2,200)(X0(I),I=1,5),SCALE
      READ (2,400)(XMAX(I),I=1,N),(XMIN(I),I=1,N)
  100 FORMAT(I2,7F10.3)
  200 FORMAT(6F10.4)
  400 FORMAT(10F8.2)
      DO 1 I=1,N
      IF (P(I))1,2,1
    1 CONTINUE
      AN=N
      BN=AN/2.0
      NB=BN
      NN=2*NB
      C=1.0
      RK=1.0
      KPK=1
      SUMJ=1000.0
   72 IF(N-NO)3,4,3
    3 J=(J+1)/2
```

```
      GO TO 10
    4 J=N/2+1
   10 R(1,1)=1.0
      DO 201 I=20,21
  201 CALL HYDO(I,0)
      CALL HYDO(22,1)
      CALL HYDO(26,0)
      CALL HYDO(27,0)
      CALL HYDLY(60)
      CALL HYDO(22,0)
      PAUSE
      R(2,1)=-P(N)
      K=1
      DO 5 I=2,J
      KK=N-K
      R(1,I)=-P(KK)
    5 K=K+2
      K=2
      IF (N-NO)11,12,11
   12 J=J-1
   11 DO 6 I=2,J
      KK=N-K
   13 R(2,I)=-P(KK)
    6 K=K+2
      NN=N-1
      IF(R(2,1))91,91,7
    7 DO 8 I=3,N
      DO 8 K=1,J
      R(I,K)=(R(I-1,1)*R(I-2,K+1)-R(I-2,1)*R(I-1,K+1))/R(I-1,1)
      IF(R(I,1))92,92,8
    8 CONTINUE
      R(N+1,1)=R(N-1,2)
      IF(R(N-1,2))93,93,9
    9 DO 14 I=1,N
      RK(N,I)=-(A(I)+P(I))/(B**2)
   14 RK(I,N)=RK(N,I)
      DO 15 I=1,NN
      IF(I-1)18,17,18
   17 DO 16 K=2,N
      RK(I,K-1)=A(I)*RK(N,K)+A(K)*RK(I,N)+RK(I,N)*RK(N,K)*(B**2)
   16 RK(K-1,I)=RK(I,K-1)
      GO TO 15
   18 DO 19 K=3,N
      RK(I,K-1)=-RK(I-1,K)+A(I)*RK(N,K)+A(K)*RK(I,N)+RK(I,N)*RK(N,K)*
     1(B**2)
   19 RK(K-1,I)=RK(I,K-1)
   15 CONTINUE
      DO 34 I=1,N
      DO 34 J=1,N
    4 S(I,J)=RK(I,J)
      DO 33 M=1,N
      SMAX=S(M,M)
      DO 24 I=M,N
      IF(ABS(S(I,I))-ABS(SMAX))24,24,25
   25 SMAX=S(I,I)
      L=I
   24 CONTINUE
      IF(L-M)26,26,27
   27 DO 28 I=M,N
```

```
      TC(I,L)=S(I,M)
      S(I,M)=S(I,L)
   28 S(I,L)=TC(I,L)
      DO 29 I=M,N
      TR(L,I)=S(M,I)
      S(M,I)=S(L,I)
   29 S(L,I)=TR(L,I)
   26 DO 30 I=M,N
   30 S(M,I)=S(M,I)/ABS(SMAX)
      MM=M+1
      DO 31 I=MM,N
      DO 31 J=M,I
   31 S(I,J)=S(I,J)-S(I,M)*S(M,M)*S(M,J)
      DO 32 I=M,NN
      III=I+1
      DO 32 J=III,M
   32 S(I,J)=S(J,I)
      IF(S(M,M))94,94,33
   33 CONTINUE
      DO 82 I=1,N
      IF(ABS(P(I))-10.0)22,23,23
   23 WRITE(3,500)I
  500 FORMAT(I4)
      PT(I)=0.1*(ABS(P(I))-10.0)
      GO TO 82
   22 PT(I)=0.1*ABS(P(I))
   82 CONTINUE
      WRITE(3,800) PT
      DO 126 K=1,N
  126 JP(K)=10000.0*PT(K)
      CALL PUTS(1,216,JP(1))
      CALL PUTS(1,227,JP(1))
      CALL PUTS(1,205,JP(2))
      CALL PUTS(1,217,JP(2))
      CALL PUTS(1,206,JP(3))
      CALL PUTS(1,225,JP(3))
      DO 40 I=20,21
   40 CALL HYDU(I,0)
      CALL HYDU(27,0)
      CALL HYDU(22,1)
      CALL HYDU(26,0)
      CALL HYDLY(60)
      CALL HYDU(22,0)
      CALL HYDU(21,1)
      CALL HYDU(26,1)
   41 CALL HYDI(10,K)
      IF(K)41,41,42
   42 DO 60 J=1,N
      X(J)=0.0
      IF(XMAX(J))202,203,202
  203 IF(XMIN(J))202,60,202
  202 WRITE(3,500)J
      WRITE(3,1004)
 1004 FORMAT(24H CONNECT TR10 TO XOUT(J))
      PAUSE
      CALL HYMK
      CALL HYAIR(10,2,A(1),TA(1))
      CALL HYUMK
      IF(TT-TA(1))43,43,411
```

```
  411 DO 412 I=2,3
      CALL HYDO(27,1)
      CALL HYDO(21,0)
      WRITE(3,1000)
 1000 FORMAT(20H CHANGE C10 POLARITY)
      PAUSE
      CALL HYDO (27,0)
      CALL HYDLY(60)
      CALL HYDO(27,1)
      CALL HYDO(21,1)
      CALL HYDO(27,0)
  413 CALL HYDI(10,K)
      IF(K)413,413,414
  414 CALL HYMK
      CALL HYAIR(10,2,AM(I),TA(I))
      CALL HYUMK
  412 CONTINUE
      DO 415 I=2,3
      IF(ABS(AM(1))-ABS(AM(I)))417,416,416
  416 X(J)=AM(1)
      T(J)=TA(1)
      GO TO 415
  417 X(J)=AM(I)
      AM(1)=AM(I)
      T(J)=TA(I)
      TA(1)=TA(I)
  415 CONTINUE
      WRITE(3,900)X(J)
   43 IF(ABS(X(J))-ABS(XO(J)))60,45,60
   45 X(J)=XO(J)
   60 CONTINUE
   44 DO 51 J=1,N
      IF(XMAX(J))46,47,46
   47 IF(XMIN(J))46,48,46
   48 F(J)=0.0
      GO TO 51
   46 H(J)=(XMAX(J)-X(J))*(X(J)-XMIN(J))
      IF(H(J))49,50,50
   49 F(J)=1.0
      GO TO 51
   50 F(J)=0.0
   51 CONTINUE
      SUMF=0.0
      DO 52 J=1,N
   52 SUMF=SUMF+F(J)
      J=1
  101 IF(SUMF)53,54,53
   53 IF(F(J))73,74,73
   54 IF(X(J)-XO(J))55,56,55
   56 F(J)=0.0
   74 DO 57 KK=1,N
   57 S(J,KK)=0.0
      J=J+1
      IF(J-N)101,101,103
   55 IF(XMAX(J))104,105,104
  105 IF(XMIN(J))104,56,104
  104 F(J)=1.0
   73 WRITE(3,900)T(J)
  555 IF(KK-3)715,715,103
```

```
715   KK=100.0*T(J)
      CALL POTS(1,230,KK)
      DO 130 I=20,21
130   CALL HYDO(I,0)
      CALL HYDO(22,1)
      DO 58 M=23,25
 58   CALL HYDO(M,0)
      II=J+11
      DO 59 M=23,25
      CALL HYDO(22,1)
      CALL HYDO(M,1)
      CALL HYDO(26,0)
      CALL HYDO(27,0)
      CALL HYDLY(60)
      CALL HYDO(22,0)
      CALL HYDO(20,1)
      CALL HYDO(26,1)
 62   CALL HYDI(10,K)
      IF(K)62,62,63
 63   JJ=M-22
      CALL HYMK
      CALL HYAIR(II,1,S(J,JJ))
      CALL HYUMK
      WRITE(3,900)S(J,JJ)
 59   CALL HYDO (M,0)
      J=J+1
      IF(J-N)101,101,103
103   SUMJ1=SUMJ2
      SUMJ2=SUMJ
      SUMJ=0.0
      DO 64 J=1,N
 75   COST=H(J)*F(J)
 64   SUMJ=SUMJ+COST
      WRITE(3,900)SUMJ
553   IF(KKK-2)708,708,702
708   IF(ABS(SUMJ2)-ABS(SUMJ))701,702,702
701   ESTS=SUMJ2
      DO 703 I=1,N
703   ESTP(I)=PPP(I)
      IF(KKK-2)711,707,553
711   DK=0.5*DK
716   DO 704 I=1,N
704   P(I)=ESTP(I)+DK*PCOR(I)
      KKK=2
      GO TO 405
707   IF(ABS(ESTS)-ABS(SUMJ))709,709,710
710   KKK=1
      GO TO 702
709   DK=0.5*DK
      IF(DK-0.004)712,712,716
712   DO 714 I=1,N
714   P(I)=ESTP(I)
      KKK=3
      GO TO 405
702   IF(KKK-2)121,160,122
160   IF(ABS(SUMJ)-ABS(ESTS))161,709,709
161   ESTS=SUMJ
      DO 162 I=1,N
162   ESTP(I)=P(I)
```

```
      GO TO 121
  124 IF(ABS(SUMJ)-ABS(FSTS))124,123,123
  123 KKK=5
      FSTS=SUMJ
      DO 163 I=1,N
  163 FSTP(I)=P(I)
      GO TO 715
  125 C=0.5*C
      IF(C-0.004)403,126,126
  126 DO 127 I=1,N
      PCOR(I)=C*PCOR(I)
      IF(SUMJ)128,403,129
  128 P(I)=FSTP(I)-PCOR(I)
      GO TO 127
  129 P(I)=FSTP(I)+PCOR(I)
  127 CONTINUE
      GO TO 405
  121 SDEL=0.0
      DO 65 K=1,N
      SPAR=0.0
      DO 66 J=1,N
   78 IF(K-2)108,108,109
  109 S(J,K)=0.1*S(J,K)
  108 PARJ=(XMAX(J)+XMIN(J)-2.0*X(J))*F(J)*S(J,K)
   66 SPAR=SPAR+PARJ
      PAR(K)=SPAR
      WRITE(3,900)PAR(K)
      PAR2= SPAR **2
   65 SDEL=SDEL+PAR2
      DO 422 I=1,N
      PP(I)=PPP(I)
      PPP(I)=P(I)
   23 IF(KKK-2)802,802,23
  802 PCOR(I)=SUMJ*PAR(I)/SDEL
      GO TO 67
   20 IF(DK-1.0)21,802,802
   21 PCOR(I)=C*PAR(I)/SQRT(SDEL)
      KKK=4
      IF(SUMJ)69,403,67
   67 P(I)=P(I)+PCOR(I)
      GO TO 603
   69 P(I)=P(I)-PCOR(I)
  603 WRITE(3,900)PCOR(I)
  422 CONTINUE
  405 WRITE(3,600)(P(I),I=1,N)
      CI=45.0*P(1)-6.0*P(2)
      IF(CI)131,131,132
  131 IC=-1
      GO TO 133
  132 IC=1
  133 WRITE(3,500)IC
  600 FORMAT(3E20.4)
      PAUSE
      GO TO 72
  403 IF(ABS(FSTS)-ABS(SUMJ))61,61,70
   61 WRITE(3,900)FSTS
      GO TO 71
   70 WRITE(3,900)SUMJ
   71 WRITE(3,600)(X(I),I=1,N)
```

```
      WRITE(3,600)(X(I),I=1,N)
      WRITE(3,600)((RK(I,J),J=1,N),I=1,N)
      RK=1.0
      KKK=1
      XKXO=0.0
      DO 425 I=1,N
      DO 425 J=1,N
      VALUE=RK(I,J)*X0(I)*X0(J)*SCALE**2
  425 XKXO=XKXO+VALUE
      XKXO=0.5*XKXO
      WRITE(3,1001)
 1001 FORMAT(26H CHANGE C10 ENABLE TO TR20)
      PAUSE
      DO 426 I=20,21
  426 CALL HYDO(I,0)
      CALL HYDO(22,1)
      CALL HYDO(26,0)
      CALL HYDO(27,0)
      CALL HYDLY(60)
      CALL HYDO(22,0)
      CALL HYDO(21,1)
      CALL HYDO(26,1)
  427 CALL HYDI(10,K)
      IF(K)427,427,428
  428 CALL HYMK
      CALL HYAIR(10,1,X(3))
      CALL HYAIR(15,2,X(2),X(3))
      CALL HYUMK
      XKXT=0.0
      DO 429 I=1,N
      DO 429 J=1,N
      FINAL=RK(I,J)*X(I)*X(J)*SCALE**2
  429 XKXT=XKXT+FINAL
      XKXT=0.5*XKXT
      COSTJ=XKXO-XKXT
      WRITE(3,1002)
 1002 FORMAT(6H COSTJ)
      WRITE(3,600)XKXO,XKXT,COSTJ
      WRITE(3,1003)
 1003 FORMAT(26H CHANGE C10 ENABLE TO TR21)
      PAUSE
    2 WRITE(3,800)(P(I),I=1,N)
  800 FORMAT(5E20.4)
      GO TO 80
   91 WRITE(3,900)R(2,1)
  900 FORMAT(E20.4)
      GO TO 80
   92 WRITE(3,150)I,P(I,1)
  150 FORMAT(I2,10X,E20.4)
      GO TO 80
   93 WRITE(3,250)N,P(N+1,1)
  250 FORMAT(I2,10X,E20.4)
      GO TO 80
   94 WRITE(3,350)M,S(M,M)
  350 FORMAT(I2,50X,E20.4)
   80 CALL EXIT
      END
```