2(mix)

[redacted]

FACILITY FORM 602		
N71-18062 (ACCESSION NUMBER)		(THRU)
152 (PAGES)		G3 (CODE)
CR-103032 (NASA CR OR TMX OR AD NUMBER)		10 (CATEGORY)

Rensselaer Polytechnic Institute

Troy, New York

Rensselaer Polytechnic Institute

Troy, New York 12181

Final Report - Vol. I

Contract No. NAS8-21131

Covering Period Nov. 4, 1969 - April 4, 1970

NATIONAL AERONAUTICS AND
SPACE ADMINISTRATION

Optimal Time Invariant Output Feedback Controllers

by Derek E. McBrinn

Submitted on behalf of

Rob Roy
Professor
Systems Engineering Division

NOTICE TO USERS

Portions of this document have been judged by the Clearinghouse to be of poor reproduction quality and not fully legible. However, in an effort to make as much information as possible available to the public, the Clearinghouse sells this document with the understanding that if the user is not satisfied, the document may be returned for refund.

If you return this document, please include this notice together with the IBM order card (label) to:

Clearinghouse
Attn: 152.12
Springfield, Va. 22151

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

Symbol	Meaning	Page First Used
A	System matrix	2
$\hat{A}$	Equivalent system matrix	36
B	Control matrix	2
C	Output/state matrix	8
$\hat{C}$	Equivalent output/state matrix	9
D	Output/control matrix	8
D	Arbitrary square matrix	14
D(t)	Riccati matrix	2
F	Terminal state cost matrix	2
G	Gradient matrix	15
GF	Gain function	41
I	Identity matrix	8
Ip	Permutation matrix	9
J	Cost functional	2
K	Matrix of feedback gains	3
K_0	Base gain matrix	47
$\hat{K}$	Perturbed gain matrix	47
M	Modal matrix	51
NC	Number of controls	2
NF	Number of feedback states	2
NS	Number of states	2
P	Equivalent cost matrix	37
Q	State cost matrix	2
$\hat{Q}$, $\hat{R}$, $\hat{W}$	Cost matrices used in Cassidy's SOC technique	7
R	Control cost matrix	2

Symbol	Meaning	Page First Used
T	Terminal time for optimization problem	2
V	Covariance matrix	39
W	Matrix of necessary conditions	44
t	Time	2
u	Scalar control	56
$\underline{u}(t)$	Control vector	2
$\underline{v}^T$, $\underline{v}_i^T$	Row eigenvector of a matrix	14
$\underline{w}$, $\underline{w}_i$	Column eigenvector of a matrix	14
$\underline{x}(t)$	State vector	2
$\hat{\underline{x}}(t)$	Subset of state vector	9
$\underline{y}(t)$	Output vector	8
$\underline{y}(t)$	Alternative state vector	40
$\underline{z}(t)$	Alternative state vector	9
α	Parameter of a matrix	14
Δ	Small perturbation	16
δ	Small perturbation	47
ϵ	Element of the set	2
$\phi(t)$	State transition matrix	36
Λ	Diagonal matrix of eigenvalues	51
λ, λ_i	Eigenvalue of a matrix	14
μ	Step size (convergence) parameter	16
∇	Gradient matrix	16
(t)	Function of time	2
$\cdot$ as in $\dot{\underline{x}}(t)$	Differentiation with respect to time	2
^{T} as in B^T	Transform of matrix or vector	2
$^{-1}$ as in R^{-1}	Inverse of matrix	2

Symbol	Meaning	Page First Used
" " as in "K"	Vector arrangement of a matrix by column ordering	15
$[0, T]$	Closed time interval from 0 to T	2
ij as in k_{ij} or $[\ \ ]_{ij}$	Element in row i, column j of the matrix	10
i as in x_i	i^{th} component of vector	15
q as in $\underline{b}_q$	q^{th} column of the matrix B	14
$\vert_T$	Evaluation at given point	33
$\Vert\ \ \Vert$	Euclidian norm	16
$Re[\ \ ]$	Real part	15
$E[\ \ ]$	Expected value	38
$\exp[\ \ ]$	Exponentiation	37
$tr[\ \ ]$	Trace of matrix	33

ACKNOWLEDGMENT

The author gratefully acknowledges the advice and guidance given him by his thesis committee in the preparation of this dissertation. A particular debt is due to his advisor, Dr. Rob Roy, strategist extraordinaire, for a special blend of encouragement, exhortation and counsel. Thanks also go to the author's colleagues at R.P.I. for their willingness to enter technical discussions at any time; Mr. Lutz Willner must be singled out for special mention in this respect.

The manuscript was typed by Miss Kathy Reid, and her skill and patience are much appreciated.

This research was performed at Rensselaer Polytechnic Institute under the sponsorship of NASA Contract No. NAS8-21131. Financial assistance as also received from an N.D.E.A. Title IV Fellowship and from a grant by Ling Temco Vought, Inc.

ABSTRACT

The problem of the determination of optimal time-invariant output feedback controllers for linear dynamic systems with quadratic cost functionals is considered. Two distinct cases arise, depending on whether optimization is over a finite or a semi-infinite time interval.

For the semi-infinite time interval problem a gain initialization technique is derived to complement existing optimization techniques. The gain initialization technique determines the feedback gains required to (locally) maximize the system stability. A computational algorithm for the technique is incorporated in a digital computer program, and is used to stabilize a seven state model of a Saturn V booster rocket.

For the finite time interval problem a technique is derived to (locally) minimize the expected value of the cost functional. The technique uses the concept of Initial State Averaging. A computational algorithm is provided and incorporated in a digital computer program. The technique is illustrated by three examples.

CHAPTER I

INTRODUCTION

I.1 Motivation

During the past decade much excellent work has been done in the field of Optimal Control Theory. Many books and a great number of technical papers have been published - the Bibliography of this report cites only a small portion of the total. Not all of this work has been purely theoretical. A substantial background has been developed in the important practical area of computation and it has for several years been feasible to determine optimal controllers for a wide variety of physical systems.

It would be expected that practical applications of this theory would have appeared in abundance. Optimization is the essence of all good engineering design, and practicing engineers should seize upon any techniques which might aid them in their work. Application of the new techniques has, however, been disappointingly slow. With a few notable exceptions the practical design of control systems has remained based on the transfer-function techniques developed prior to the 1960's.

The explanation of the above paradox is widely recognized. It lies in the nature of the word "optimal", which is meaningless without a criterion of optimality. By and large, control theorists have used criteria of optimality dependent only on the performance of the control system. Design engineers, on the other hand, interpret "optimal" as embracing both system performance and system cost. The difference may be best illustrated by a simple example.

Consider the problem of designing a speed control device for a cheap movie camera. The objective is to ensure that sixteen frames of film are exposed per second regardless of film tension, state of charge of the motor batteries, etc. The could be interpreted as comprising an example of the state-regulator problem in Optimal Control Theory, with the following method of analysis. The equations governing the film transportation are determined and linearized to the form

$$\dot{\underline{x}}(t) = A\,\underline{x}(t) + B\,\underline{u}(t) \qquad (I.1\text{-}1)$$

where $\underline{x}(t)$ is an NS-vector describing the deviation of the system from its desired state at time t

$\underline{u}(t)$ is an NC-vector defining the control inputs at time t

A and B are matrices whose elements depend on the characteristics of the system.

A cost functional is formulated having the form

$$J = \underline{x}^T(T)\;F\;\underline{x}(T) + \int_0^T \left[\underline{x}^T(\tau)\;Q\;\underline{x}(\tau) + \underline{u}^T(\tau)\;R\;\underline{u}(\tau)\right] d\tau \qquad (I.1\text{-}2)$$

where T is the duration of the scene to be filmed. The objective is to determine the control input $\underline{u}(t)$ for $t \in \left[0\,,\,T\right]$ which minimizes the cost functional J. The positive semi-definite matrices F and Q are chosen to penalize any given control system for the deviations it allows from the desired state. The positive definite matrix R is a penalty for improvident usage of control power (i.e., battery energy). The solution to the problem defined by (I.1-1) and (I.1-2) can be shown[2] to be

$$\underline{u}(t) = -R^{-1}\,B^T\,D(t)\,\underline{x}(t) \qquad (I.1\text{-}3)$$

where $D(t)$ satisfies the matrix Riccati equation

$$\dot{D}(t) = -D(t)\,A - A^T D(t) + D(t)\,BR^{-1}\,B^T D(t) - Q \qquad (I.1\text{-}4)$$

with

$$D(T) = F \tag{I.1-5}$$

Defining

$$K^T(t) = R^{-1} B^T D(t) \tag{I.1-6}$$

and re-writing (I.1-3) as

$$\underline{u}(t) = -K^T(t)\, \underline{x}(t) \tag{I.1-7}$$

we see that the optimal control input can be generated as a time-varying linear combination of all of the states of the system. We note in passing that the less-fastidious photographer, willing to edit a few frames from the beginning and end of each scene and settle for good steady state performance, would be rewarded by simplification of the controller to the form[2]

$$\underline{u}(t) = -K^T\, \underline{x}(t) \tag{I.1-8}$$

This time-invariant controller would be much simpler to mechanize than that described by (I.1-7).

Now consider that speed control of the typical cheap movie camera is achieved by means of a simple flyball governor driving an on/off switch between the batteries and the motor. The flyball governor, invented by James Watt in 1788, was the first widely used automatic feedback controller.[5] In its simpler form it suffers from "hunting" about the set point. In its application to the movie camera it is driven by the single system state, motor speed. The single feedback gain can be considered as constant in the vicinity of the set point. It is clear that the flyball governor does not satisfy the requirements for an optimal controller as defined by (I,1-7).

Yet the flyball governor is indeed an optimal controller in the best engineering sense. It does an acceptable job at minimum cost. The mechanization of the "optimal" controller defined by (I.1-7) on the other hand, would be prohibitively expensive. It would require measurement and feedback of all system states. It would also require a digital computer to calculate and store the time-varying feedback gains for each scene to be filmed.

It should be clear from the above that the allowable degree of complexity of a control system is often constrained by considerations of cost. Weight, reliability and common engineering sense similarly often dictate simplicity. What the control system designer wants then is not that system which performs in the best possible manner. Rather he wishes the best possible performance for a given degree of complexity. This problem is considerably more difficult than that defined by (I.1-1) and (I.1-2). It must however, be solved if the benefits of optimal control are ever to be realized for the majority of potential applications.

This report considers a portion of the general problem of optimization within a given degree of control system complexity. Only linear systems with quadratic cost functionals are considered. Such combinations may be described by (I.1-1) and (I.1-2). The allowable degree of complexity of the controller is assumed to be time-invariant feedback of the system outputs only. The system outputs are those quantities which can be measured, and in most cases provide less than a full description of the system state at any time. The above limitations are those which are normally applied in classical control system analysis techniques. Thus this report seeks to optimize the classical controller, without increasing its complexity.

I.2 Historical Review

It is generally conceded that control system design prior to the Second World War was primarily an art. The techniques used were often empirical and thus confined to particular classes of problems. This situation was largely eliminated by the contributions of Nyquist,[6] Bode[7] and Evans[8] who established the theory and techniques of control system design which predominate in practical work even today. These techniques, however, are most suited to the design of relatively simple systems. They work best for systems having a single input and a single output related by a transfer function. Also these techniques are purely analytical and cannot be used to directly synthesize a control system.

The problem of synthesis was first tackled by Wiener[9], who considered the optimization of linear filters. This work was extended to control systems by Newton, Gould and Kaiser[10], who, however, still retained the transfer-function approach; their technique was to determine the values of system parameters required to minimize a cost functional for a specific perturbation of the system. It is significant that they made use of the Calculus of Variations in their approach to this problem. The extension to the classical Calculus of Variations provided by Pontryagin's Maximum Principle,[11] and the control problem framework provided by Kalman[12] finally brought Optimal Control Theory to maturity. This maturity, combined with the state-variable formulation presented by DeRusso, Roy and Close[1] and the increasing capability of digital computers, finally allowed the determination of optimal controllers for a wide variety of systems.

The early excitment at the disclosure of the ability to compute optimal controllers faded when it was realized that such controllers were very difficult to mechanize. The reasons for this difficulty are twofold. First, an optimal controller requires knowledge of the complete state of the system. It may not, however, be feasible to measure all system states. Secondly, for finite time duration problems an optimal controller requires time-varying feedback.

There are two fundamental approaches to elimination of the first difficulty. The approach taken by Kalman and Bucy[13] was to estimate the unmeasurable states from the (noise corrupted) system outputs or measurable states. This approach was extended by Luenberger[14] and again by Ash[15]. Estimation of the unmeasurable states allows the mechanization of a regular optimal controller. Note, however, that it addes the complexities of a state estimator to those of the optimal controller.

The second approach to the problem of unmeasurable states is to design a controller which uses only the available information. Such a controller will be "sub-optimal" in the mathematical sense of minimizing a cost functional such as that described by (I.1-2). It may well, however, be optimal in the engineering sense. This approach has been tried by a number of people. Newton, Gould and Kaiser[10] essentially advocated such controllers, but optimized them for specific system disturbances. Consideration of only a specific disturbance is equivalent to imposing a specific initial condition on the system to be controlled. Max-min techniques were developed in an attempt to eliminate the dependence of the solution on the chosen initial condition.[16,17] In the max-min procedure the maximum system cost with respect to a set of initial conditions is minimized with respect to the system feedback gains. It still remains to choose an appropriate set of initial conditions. Rekasius[18] circumvented this problem by considering the initial condition giving the maximum ratio of output feedback cost to allstate feedback cost but was still limited to design for this specific initial condition. Levine[19] determined the controller which was optimal in an average sense and computed time varying feedback gains for the case of the finite time-interval problem. For the semi-infinite time interval problem he required his feedback gains to be time-invariant. Cassidy[20] considered the minimization of a modified cost functional having the form

$$J = \frac{1}{2} \int_0^T (\underline{x}^T Q\underline{x} + \underline{x}^T \hat{Q}\underline{x} + \underline{x}^T W\underline{u} + \underline{x}^T \hat{W}\underline{y} + \underline{u}^T R\underline{u})\, d\tau \qquad \text{(I.2-1)}$$

where the matrices $\hat{Q}$ and $\hat{W}$ were chosen to ensure that the feedback gains for the unavailable states were zero. The resultant feedback gains were again time-varying for a finite time interval, time-invariant for the semi-infinite time interval.

It is considered that the approaches used by Levine and Cassidy are the most satisfactory of those considered, since they are both completely independent of initial conditions. They both, however, result in time-varying gains for the finite time-interval case. For the semi-infinite time interval both give constant feedback gains, but use computational algorithms which require initialization by a set of stabilizing feedback gains.

The problem of determining stabilizing output-feedback gains for complex systems has not, to the author's knowledge, been satisfactorily solved. Koenigsberg[21] has considered local stability maxima, but his computational algorithm is felt to be inefficient. Jameson[22] and Davison[23] have shown that as many system roots as there are feedback states can be arbitrarily determined, but the remaining roots are then unconstrained.

A solution to the problem of determining optimal constant or piecewise constant feedback gains for a finite time interval has been derived by

Kleinman, Fortmann and Athans.[25] As with Levine,[19] they considered optimality in an average sense, but confined attention to the allstate feedback case.

I.3 Scope and Contribution of This Work

This report considers the determination of optimal time-invariant output feedback control gains for systems which can be described by the equation

$$\dot{\underline{x}}(t) = A\,\underline{x}(t) + B\,\underline{u}(t) \qquad \text{(I.3-1)}$$

where $\underline{x}(t)$ is an NS-vector describing the state of the system at time t

$\underline{u}(t)$ is an NC-vector describing the control inputs at time t

A,B are time invariant matrices of appropriate order.

It is assumed, without loss of generality, that the system outputs are the first NF states. If this is not the case then a new set of state variables may be obtained as follows to satisfy the assumption. Suppose that the system described by (I.3-1) has NF independent outputs $\underline{y}(t)$ given by

$$\underline{y}(t) = C\,\underline{x}(t) + D\,\underline{u}(t) \qquad \text{(I.3-2)}$$

We shall require that

$$\underline{u}(t) = -\,K^T\,\underline{y}(t) \qquad \text{(I.3-3)}$$

where K is the time-invariant matrix of feedback gains. Thus

$$\left[I + DK^T\right]\,\underline{y}(t) = C\,\underline{x}(t) \qquad \text{(I.3-4)}$$

where I is the identity matrix of order NF. Assuming $\left[I + DK^T\right]$ to be non-singular gives

$$\underline{y}(t) = \hat{C}\,\underline{x}(t) \tag{I.3-5}$$

where

$$\hat{C} = \left[I + DK^T\right]^{-1} C \tag{I.3-6}$$

We note that in most cases D will be zero, leaving $\hat{C}$ equal to C. We now form the vector $\hat{\underline{x}}(t)$ comprising any $(NS - NF)$ of the $\underline{x}(t)$ state variables which are independent of the $\underline{y}(t)$ variables. We note that such variables must exist since $\underline{x}(t)$ spans NS space while $\underline{y}(t)$ only spans NF space. Appending $\hat{\underline{x}}(t)$ to $\underline{y}(t)$ gives

$$\underline{z}(t) = \begin{bmatrix} \underline{y}(t) \\ \hat{\underline{x}}(t) \end{bmatrix} = \begin{bmatrix} \hat{C} \\ Ip \end{bmatrix} \underline{x}(t) \tag{I.3-7}$$

where Ip is a permutation matrix describing how $\hat{\underline{x}}(t)$ was chosen from $\underline{x}(t)$.

It follows that

$$\dot{\underline{z}}(t) = \begin{bmatrix} \hat{C} \\ Ip \end{bmatrix} \dot{\underline{x}}(t) \tag{I.3-8}$$

i.e.,

$$\dot{\underline{z}}(t) = \begin{bmatrix} C \\ Ip \end{bmatrix} A\,\underline{x}(t) + \begin{bmatrix} C \\ Ip \end{bmatrix} B\,\underline{u}(t) \tag{I.3-9}$$

Thus, from (I.3-7)

$$\dot{\underline{z}}(t) = \begin{bmatrix} C \\ Ip \end{bmatrix} A \begin{bmatrix} C \\ Ip \end{bmatrix}^{-1} \underline{z}(t) + \begin{bmatrix} C \\ Ip \end{bmatrix} B\,\underline{u}(t) \tag{I.3-10}$$

and $\underline{z}(t)$ is the desired state-vector having its first NF states as the system outputs.

Thus the control inputs $\underline{u}(t)$ for the system described by (I.3-1) are to be optimized over the set

$$\underline{u}(t) = - K^T \underline{x}(t) \tag{I.3-11}$$

where K, the time invariant matrix of feedback gains, is constrained to the form

$$K = \left.\begin{bmatrix} \overbrace{\; k_{ij} \;}^{NC} \\ \hdashline 0 \end{bmatrix}\right\} NS \qquad \begin{matrix} \} \; NF \\ \\ \end{matrix} \tag{I.3-12}$$

The cost functional is assumed to be of the form

$$J = \underline{x}^T(T)\ F\ \underline{x}(T) + \int_0^T \left[\underline{x}^T(\tau)\ Q\underline{x}(\tau) + \underline{u}^T(\tau)\ R\ \underline{u}(\tau)\right] d\tau \tag{I.3-13}$$

where F and Q are positive semi-definite

R is positive definite

and optimization is over the time interval $[0\ ,\ T]$.

Two separate cases are investigated. For the case where the terminal time T is infinite it is considered that the techniques presented by either Levine[19] or Cassidy[20] are adequate. Both these techniques, however, require initialization by stabilizing feedback gains. In practice, indeed the author has found that feedback gains giving marginal stability may be inadequate, due to numerical considerations. Chapter II of this report therefore presents a computational algorithm designed to find feedback gains of the form given by (I.3-12) such that the system stability is driven to a local maximum. Note that maximum stability is here defined to mean that the least stable of the system roots is made as stable as possible. The algorithm presented is felt to be superior to that given by Koenigsberg[21] in that it is computationally faster.

For the case where the terminal time T is finite, an approach is taken similar to that used by Levine[19], except that the feedback gains are required to be time-invariant. The resulting theory is presented in Chapter III, together with a computational algorithm to allow the determination of the optimal gains.

It is felt that this report will help to fill some of the gaps in the existing ability to determine practical feedback controllers which are optimal in the engineering sense.

CHAPTER II

THE GAIN INITIALIZATION PROBLEM

II.1 Introduction

The problem of determination of the output feedback gains required to stabilize a dynamic system is an interesting one in its own right. In this report, however, we are concerned with the determination of such gains as a pre-requisite to the application of optimization techniques.

Both Levine[19] and Cassidy[20] have considered the determination of optimal output feedback controllers for linear systems. Both have derived iterative techniques resulting in time invariant feedback gains when the system is optimized over the semi-infinite time interval $[0, \infty]$. Each technique, however, requires initialization with a set of stabilizing feedback gains in order to ensure convergence of the computational algorithm.

Based on the author's practical experience[26] with Cassidy's technique it appears that an extra requirement on initial system stability may be added when the optimization algorithms are mechanized on a digital computer. It was found that with low order systems (five states or less) marginal stability was sufficient to ensure convergence of the optimization algorithm. When a twenty-one state model of the Saturn V booster rocket was considered, however, initialization by feedback gains giving marginal stability was insufficient to cause convergence. This was judged to be due to numerical truncation resulting from the finite word length in the digital computer. The problem was solved by computing a new set of feedback gains giving better than marginal stability. Re-initialization of the optimization algorithm with the new feedback gains resulted in convergence. It is considered likely that Levine's algorithm would have similar characteristics.

It is clear that practical application of Levine's and Cassidy's techniques requires a method for determination of stabilizing output feedback gains. It would be desirable that the gains give more than marginal stability. A method is described in this Chapter for determination of such gains, when they exist.

II.2 Theory of the Gain Initialization Technique

The problem under consideration is as follows. Given the system

$$\dot{\underline{x}}(t) = A\, \underline{x}(t) + B\, \underline{u}(t) \qquad \text{(II.2-1)}$$

where $\underline{x}(t)$ is an NS-vector describing the state of the system at time t

$\underline{u}(t)$ is an NC-vector describing the control inputs at time t

A,B are matrices whose elements depend on the characteristics of the system

with

$$\underline{u}(t) = -K^T\, \underline{x}(t) \qquad \text{(II.2-2)}$$

where K is a matrix of feedback gains such that

$$\dot{\underline{x}}(t) = \left[A - BK^T\right]\, \underline{x}(t) \qquad \text{(II.2-3)}$$

where K is constrained to the form

$$K = \overbrace{\begin{bmatrix} k_{ij} \\ \hdashline 0 \end{bmatrix}}^{NC} \quad \left.\begin{matrix} \} \; NF \\ \\ \end{matrix}\right\} NS \qquad \text{(II.2-4)}$$

how should the feedback gain matrix K be modified, within the constraints of (II.2-4) to increase the stability of the least stable eigenvalue of (II.2-3)

The solution hinges on an expression given by Fadeev and Fadeeva[27] for the sensitivity of an eigenvalue of a matrix to a parameter of the matrix. Given any eigenvalue λ and corresponding row and column eigenvectors $\underline{v}^T$ and $\underline{w}$ of a square matrix D, the sensitivity of the eigenvalue to a parameter α of the matrix is given by

$$\frac{d\lambda}{d\alpha} = \frac{\underline{v}^T \frac{\partial D}{\partial \alpha} \underline{w}}{\underline{v}^T \underline{w}} \tag{II.2-5}$$

We note that $\underline{v}$ and $\underline{w}$ satisfy

$$D^T \underline{v} = \lambda \underline{v} \tag{II.2-6}$$

$$D\underline{w} = \lambda \underline{w} \tag{II.2-7}$$

To apply (II.2-5) to the system described by (II.2-3) and (II.2-4) we note that

$$\frac{\partial [A-BK^T]}{\partial k_{pq}} = \Big[\, 0 \;\vdots\; \underbrace{\underline{b}_q}_{p^{th}\ \text{column}} \;\vdots\; 0 \,\Big] \tag{II.2-8}$$

where $\underline{b}_q$ is the q^{th} column of the matrix B. Thus if λ_i is any eigenvalue of (II.2-3) and if $\underline{v}_i^T$ and $\underline{w}_i$ are corresponding row and column eigenvectors we see that

$$\frac{\partial \lambda_i}{\partial k_{pq}} = \frac{\underline{v}_i^T \Big[\, 0 \;\vdots\; \overbrace{\underline{b}_q}^{p^{th}\ \text{column}} \;\vdots\; 0 \,\Big] \underline{w}_i}{\underline{v}_i^T \underline{w}_i} \tag{II.2-9}$$

The sensitivity of the real part of λ_i to the real feedback gain k_{pq} is simply the real part of the right hand side of (II.2-9).

Suppose now that it is desired to decrease the real part of λ_i. The direction of the steepest descent in feedback gain space is defined by the gradient matrix G where

$$g_{pq} = \mathrm{Re}\left\{\frac{\partial \lambda_i}{\partial k_{pq}}\right\} \tag{II.2-10}$$

$p = 1, \ldots, NF$

$q = 1, \ldots, NC$

II.3 The Gain Initialization Algorithm

A gain initialization algorithm has been derived from the expressions given in (II.2-9) and (II.2-10), and has been incorporated in the digital computer program GRADGN. A listing of the program GRADGN is given in Appendix I. A description of the algorithm/program follows.

The algorithm is iterative. It increases the stability of the least stable eigenvalue of the system at each step. If, during a given interation a new eigenvalue should become less stable than the one being operated on, this fact is taken into account in subsequent iterations.

Consider the rearrangement of the variable elements of the feedback gain matrix K into an NF x NC vector "K". The rearrangement is performed by column ordering of K so that

$$K_{pq} = {}^{\prime\prime}K^{\prime\prime}_{(q-1)NF + p} \tag{II.3-1}$$

The gradient matrix G is similarly rearranged giving

$$G_{pq} = {}^{\prime\prime}G^{\prime\prime}_{(q-1)NF + p} \tag{II.3-2}$$

The steepest descent method requires that variations in the gain vector "K" must lie along the negative of the gradient vector "G". Thus

$$"\Delta K" = - \mu "G" \tag{II.3-3}$$

where μ is a positive constant determined by the step size to be taken and "ΔK" is a variation in the gain vector. Suppose now that we wish to determine the variation "ΔK" in the gain vector "K" required to cause a small negative change $\Delta \operatorname{Re}\{\lambda_i\}$ in the real part of the eigenvalue λ_i. The variation "ΔK" may be computed from the relationship

$$"G"^T "\Delta K" = \Delta \operatorname{Re}\{\lambda_i\} \tag{II.3-4}$$

When the variation "ΔK" lies in the direction of steepest descent then, from (II.3-3) and (II.3-4)

$$-"G"^T \quad \mu "G" = \Delta \operatorname{Re}\{\lambda_i\} \tag{II.3-5}$$

whence

$$\mu = \frac{-\Delta \operatorname{Re}\{\lambda_i\}}{\|"G"\|^2} \tag{II.3-6}$$

and so

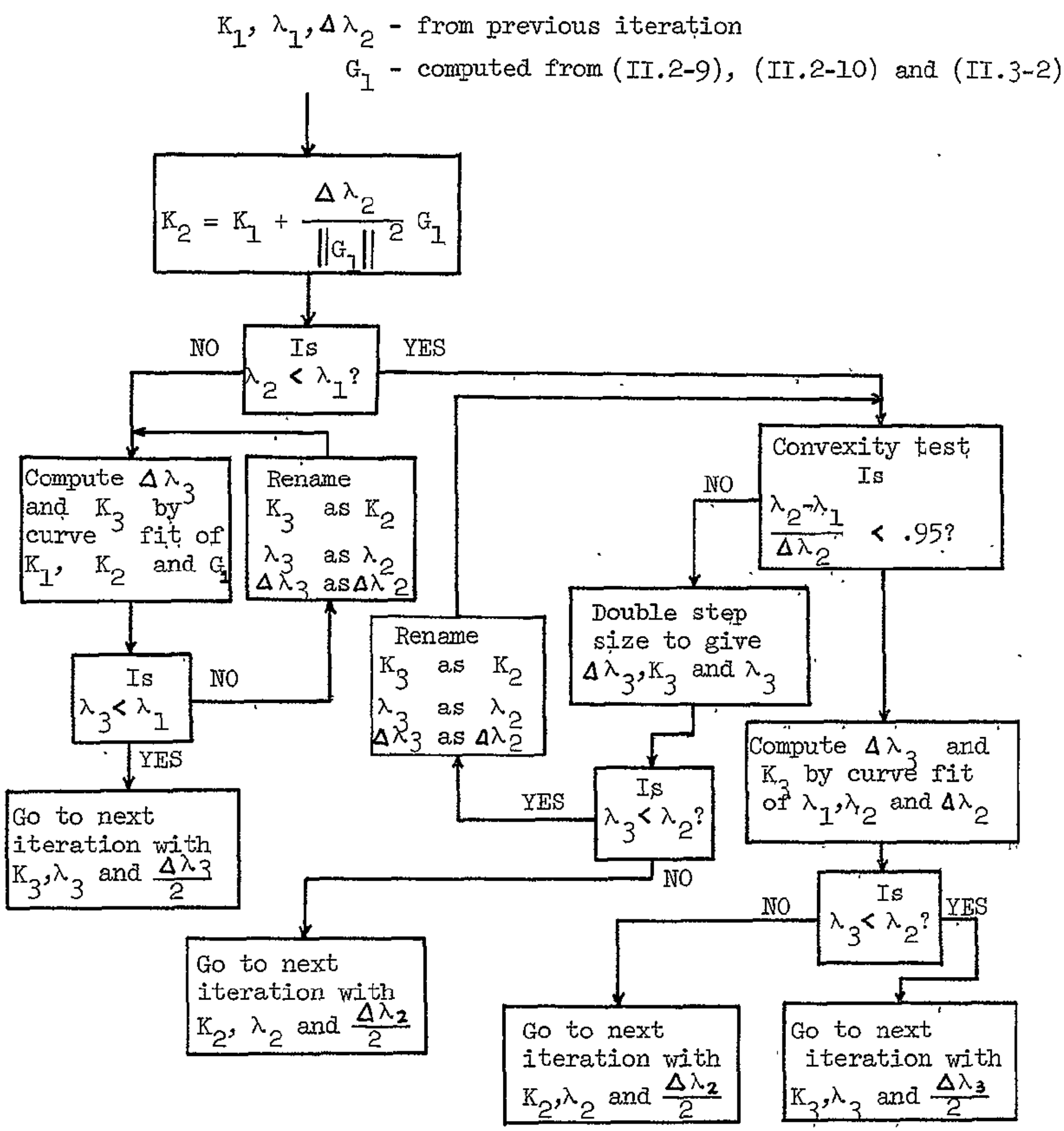
$$"\Delta K" = \frac{-\Delta \operatorname{Re}\{\lambda_i\}}{\|"G"\|^2} "G" \tag{II.3-7}$$

When the variation $\Delta \operatorname{Re}\{\lambda_i\}$ is sufficiently small the system corresponding to the feedback gains ("K" + "ΔK") should have an eigenvalue whose real part approximates $\operatorname{Re}\{\lambda_i\} + \Delta \operatorname{Re}\{\lambda_i\}$.

It remains to determine suitable values for the step size $\Delta \mathrm{Re}\{\lambda_i\}$. It is assumed that in most cases a user of GRADGN will have little idea of what comprises a suitable step size. The step size is therefore varied adaptively based on experience with previous iterations. The initial step size may either be assigned by the user or given a default value of -0.1. In either case the initial step size is used as a program criterion for termination due to diminishing returns. Should two successive iterations of the program fail to improve the stability by the initial step size, the program is terminated. It should be noted that apart from its use as a termination control the program is quite insensitive to the initial setp size, due to the rapidity of the step size adaptation.

The adaptation control for a given iterative step is described by Figure II.3-1. One of two basic step size adjustments is utilized, depending on the convexity of the trajectory of the least stable eigenvalue. If the trajectory is sufficiently convex then a quadratic curve is fitted to two points on the trajectory and the trajectory gradient at one of the points. This procedure is illustrated by Figure II.3-2. The gain vector corresponding to the minimum point of the quadratic curve is computed and tested for stability. If the trajectory is insufficiently convex then the step size is doubled. One or both of these procedures may be used repeatedly during a single iteration. Note that only one gradient vector is computed per iteration.

The initiation point for the next iteration is always the most stable gain vector found. The starting step size for the next iteration is set to one half of the current improvement in the least stable eigenvalue. This allows the step size to be changed by several orders of the magnitude in either direction during a single iteration, while allowing for the decreasing rate of improvement as a stability maximum is approached.



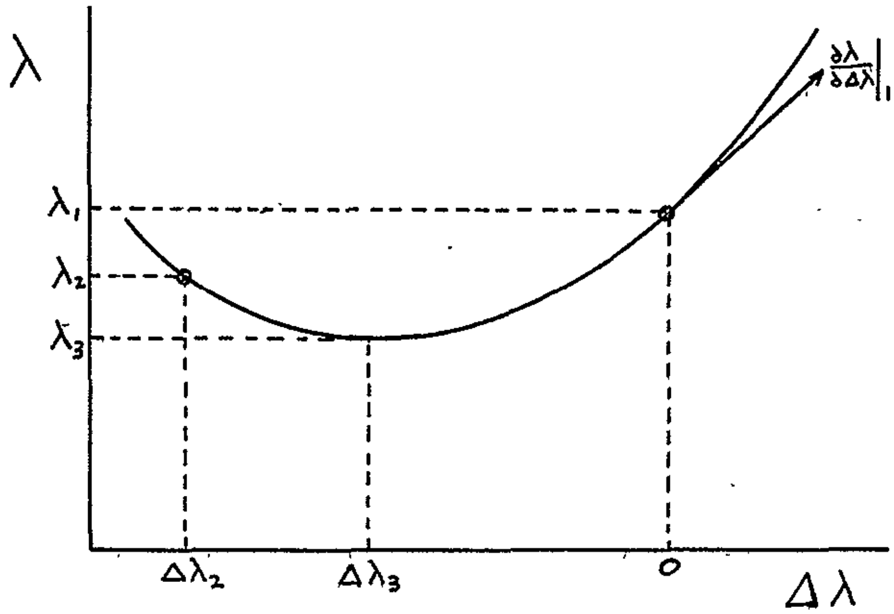
K_i = feedback gain vector at computation point i

λ_i = real part of least stable eigenvalue corresponding to feedback gains K_i

G_1 = gradient vector at computation point 1

$\Delta\lambda_i$ = step size from computation point 1 to computation point i

FIGURE II.3-1 Step Size Adaptation Control for the Gain Initialization Program GRADGN



$\Delta\lambda_i$ = step size from computation point 1 to computation point i

λ_i = real part of least stable eigenvalue at computation point i

Quadratic curve fit to variation of λ with $\Delta\lambda$ gives

$$\lambda = a(\Delta\lambda)^2 + b\,\Delta\lambda + c$$

and

$$c = \lambda_1$$

$$b = \left.\frac{\partial\lambda}{\partial\Delta\lambda}\right|_1 = 1$$

$$a = \frac{\lambda_2 - b\Delta\lambda_2 - c}{(\Delta\lambda_2)^2}$$

Curve has minimum point at $\Delta\lambda_3$ where $\left.\frac{\partial\lambda}{\partial\Delta\lambda}\right|_3 = 0$

$$\therefore\ \Delta\lambda_3 = -\frac{b}{2a} = -\frac{(\Delta\lambda_2)^2}{2(\lambda_2 - \lambda_1 - \Delta\lambda_2)}$$

FIGURE II.3-2 Step Size Adaptation in GRADGN by Quadratic Curve Fit

The computer program GRADGN incorporates an option for the degree of stability required for termination. Setting the input parameter ISTOP to zero instructs the program to maximize stability. Termination is then initiated by reaching a point of diminishing returns, as mentioned above. If ISTOP is set of 1 a minimum desired degree of stability, STOPR, may be input to the program. The program then terminates when the real part of the least stable eigenvalue becomes less than STOPR.

II.4 Example of the Use of the Gain Initialization Program GRADGN

The use of the gain initialization program GRADGN is illustrated in this Section by output feedback stabilization of a seven state model of the Saturn V booster rocket. The model represents the rocket dynamics at a point occuring 80 seconds after lift-off.

The seven states considered are

x_1 = measured pitch attitude angle ϕ_D

x_2 = measured pitch rate $\dot{\phi}_R$

x_3 = aerodynamic angle of attack α

x_4 = first bending mode deflection η_1

x_5 = first bending mode deflection rate $\dot{\eta}_1$

x_6 = engine gimbal angle β

x_7 = engine gimbal angle rate $\dot{\beta}$

It is assumed that only the first two states x_1 and x_2 are to be measured. Thus only these two states are available for feedback. Control is achieved via an actuator driving the engine gimbal angle at a rate proportional to the actuator input.

The model may be represented by

$$\underline{\dot{x}}(t) = A\,\underline{x}(t) + \underline{b}\,\underline{u}(t) \qquad \text{(II.4-1)}$$

with

$$\underline{u}(t) = -k_1\,x_1(t) - k_2\,x_2(t) \qquad \text{(II.4-2)}$$

where

$$A = \begin{bmatrix} 0. & 1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & .2030 & -.6535 & -.0020 & 2.558 & 0. \\ -.0137 & 1. & -.0407 & .0002 & -.0146 & -.0334 & 0. \\ 0. & 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & -44.67 & -.1337 & 254.6 & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 1. \\ 0. & 0. & 0. & 0. & 0. & -50. & -10. \end{bmatrix} \qquad \text{(II.4-3)}$$

$$b = \begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 1. \end{bmatrix} \qquad \text{(II.4-5)}$$

The derivation of this model is described by Cassidy.[20]

The program GRADGN was used to determine the feedback gains k_1 and k_2 corresponding to a local stability maximum of the above system. The program output is given in Figure II.4-1.

```
STATES   CONTROLS   FEEDBACKS   IGAINS   IDELR   ISTOP
   7         1          2          0        1       0

STABILITY INCREASE INCREMENT =          0.04000000

MINIMUM STABILITY REQUIRED FOR TERMINATION     -1000000.00000000

SYSTEM MATRIX AMAT
   0.0000000E 00    0.1000000E 01    0.0000000E 00    0.0000000E 00
   0.0000000E 00    0.0000000E 00    0.0000000E 00    0.0000000E 00
   0.0000000E 00    0.2030470E 00   -0.6534950E 00   -0.1955000E-02
   0.2558020E 01    0.0000000E 00   -0.1366150E-01    0.1000000E 01
  -0.4068250E-01    0.1998600E-03   -0.1463000E-01   -0.3338200E-01
   0.0000000E 00    0.0000000E 00    0.0000000E 00    0.0000000E 00
   0.0000000E 00    0.1000000E 01    0.0000000E 00    0.0000000E 00
   0.0000000E 00    0.0000000E 00    0.0000000E 00   -0.4466811E 02
  -0.1336680E 00    0.2546100E 03    0.0000000E 00    0.0000000E 00
   0.0000000E 00    0.0000000E 00    0.0000000E 00    0.0000000E 00
   0.0000000E 00    0.1000000E 01    0.0000000E 00    0.0000000E 00
   0.0000000E 00    0.0000000E 00    0.0000000E 00   -0.5000000E 02
  -0.1000000E 02

CONTROL MATRIX BMAT
   0.0000000E 00    0.0000000E 00    0.0000000E 00    0.0000000E 00
   0.0000000E 00    0.0000000E 00    0.1000000E 01

GAIN MATRIX K
   0.0000000E 00    0.0000000E 00

ROOTS    REAL PART           IMAG. PART
        -0.477798E 00        0.000000E 00
         0.423402E 00        0.000000E 00
         0.137118E-01        0.000000E 00
        -0.668340E-01        0.668309E 01
        -0.668340E-01       -0.668309E 01
        -0.500000E 01        0.500000E 01
        -0.500000E 01       -0.500000E 01

MAXIMUM REAL PART OF ROOTS
   0.4234025E 00
```


Figure II.4-1 Computer Print-Out For Example Problem

GAIN SEARCH PROCEEDS ALONG A NEW GRADIENT. ITERATION NUMBER 1

EIGVEC ERROR MESSAGES
SW1=0.8345E-06 ITER= 4 DIF=0.3576E-06

EIGENVECTORS CORRES TO MRP EIGENVALUE

ROW REAL PART	ROW IMAG PART	COL REAL PART	COL IMAG PART
-0.1411699E-01	0.0000000E 00	0.1000000E 01	0.0000000E 00
0.1000000E 01	0.0000000E 00	0.4234034E 00	0.0000000E 00
0.4375202E 00	0.0000000E 00	-0.8829014E 00	0.0000000E 00
0.2059637E-03	0.0000000E 00	-0.2525228E-26	0.0000000E 00
-0.1463000E-01	0.0000000E 00	0.8821868E-26	0.0000000E 00
-0.2263338E 00	0.0000000E 00	-0.1349601E-27	0.0000000E 00
-0.2171400E-01	0.0000000E 00	-0.4707256E-27	0.0000000E 00

GRADIENT MATRIX
0.2729351E-01 0.1155617E-01

STABILITY INCREASE STEP SIZE = 0.04000000

GAIN MATRIX K
-0.1242759E 01 -0.5261886E 00

ROOTS REAL PART	IMAG. PART
-0.708895E-01	0.666810E 01
-0.708895E-01	-0.666810E 01
-0.499273E 01	0.501011E 01
-0.499273E 01	-0.501011E 01
-0.452182E 00	0.000000E 00
0.380997E 00	0.000000E 00
0.239837E-01	0.000000E 00

STEP SIZE IS DOUBLED

GAIN MATRIX K
-0.2485518E 01 -0.1052377E 01

ROOTS REAL PART	IMAG. PART
-0.747366E-01	0.665300E 01
-0.747366E-01	-0.665300E 01
-0.498541E 01	0.502038E 01
-0.498541E 01	-0.502038E 01
-0.424900E 00	0.000000E 00
0.332093E 00	0.000000E 00
0.389951E-01	0.000000E 00

FIGURE II.4-1 (Cont'd)

```
STEP SIZE IS DOUBLED

GAIN MATRIX K
  -0.4971038E 01   -0.2104753E 01

ROOTS      REAL PART              IMAG. PART
       -0.827146E-01            0.662188E 01
       -0.827146E-01           -0.662188E 01
       -0.497082E 01            0.504117E 01
       -0.497082E 01           -0.504117E 01
       -0.364258E 00            0.000000E 00
        0.148511E 00            0.256780E-01
        0.148511E 00           -0.256780E-01

STEP SIZE IS DOUBLED

GAIN MATRIX K
  -0.9942076E 01   -0.4209508E 01

ROOTS      REAL PART              IMAG. PART
       -0.979548E-01            0.655847E 01
       -0.979548E-01           -0.655847E 01
       -0.494167E 01            0.508360E 01
       -0.494167E 01           -0.508360E 01
        0.625687E-01            0.243981E 00
        0.625687E-01           -0.243981E 00
       -0.220044E 00            0.000000E 00

STEP SIZE IS DOUBLED

GAIN MATRIX K
  -0.1988414E 02   -0.8419017E 01

ROOTS      REAL PART              IMAG. PART
       -0.126488E 00            0.642529E 01
       -0.126488E 00           -0.642529E 01
       -0.488370E 01            0.517278E 01
       -0.488370E 01           -0.517278E 01
       -0.315896E-01            0.533504E 00
       -0.315896E-01           -0.533504E 00
       -0.907448E-01            0.000000E 00

STEP SIZE IS DOUBLED
```

FIGURE II.4-1 (Cont'd)

```
GAIN MATRIX K
  -0.3976830E 02     -0.1683803E 02

ROOTS      REAL PART            IMAG. PART
        -0.477230E 01          0.536959E 01
        -0.477230E 01         -0.536959E 01
        -0.169915E 00          0.613263E 01
        -0.169915E 00         -0.613263E 01
        -0.113802E 00          0.913033E 00
        -0.113802E 00         -0.913033E 00
        -0.621070E-01          0.000000E 00

 STEP SIZE IS DOUBLED

 GAIN MATRIX K
  -0.7953661E 02     -0.3367606E 02

ROOTS      REAL PART            IMAG. PART
        -0.460448E 01          0.583911E 01
        -0.460448E 01         -0.583911E 01
        -0.144409E 00          0.542965E 01
        -0.144409E 00         -0.542965E 01
        -0.311907E 00          0.148304E 01
        -0.311907E 00         -0.148304E 01
        -0.528619E-01          0.000000E 00

 STEP SIZE TOO LARGE. COMPUTE NEW GRADIENT AT PREVIOUS BEST GAINS

 GAIN MATRIX K
  -0.3976830E 02     -0.1683803E 02

ROOTS      REAL PART            IMAG. PART
        -0.477230E 01          0.536959E 01
        -0.477230E 01         -0.536959E 01
        -0.169915E 00          0.613263E 01
        -0.169915E 00         -0.613263E 01
        -0.113802E 00          0.913033E 00
        -0.113802E 00         -0.913033E 00
        -0.621070E-01          0.000000E 00

MAXIMUM REAL PART OF ROOTS
  -0.6210702E-01
```

FIGURE II.4-1 (Cont'd)

```
GAIN SEARCH PROCEEDS ALONG A NEW GRADIENT. ITERATION NUMBER    2

EIGVEC ERROR MESSAGES
SW1=0.1207E-02          ITER=     5            DIF=0.0000E 00

EIGENVECTORS CORRES TO MRP EIGENVALUE
   ROW REAL PART        ROW IMAG PART       COL REAL PART        COL IMAG PART
  0.1000000E 01        0.0000000E 00       0.2210334E 00        0.0000000E 00
  0.1029001E 00        0.0000000E 00      -0.1373143E-01        0.0000000E 00
 -0.9744528E 00        0.0000000E 00       0.1000000E 01        0.0000000E 00
 -0.1416449E-01        0.0000000E 00       0.9880193E 00        0.0000000E 00
 -0.1529490E-02        0.0000000E 00      -0.6137960E-01        0.0000000E 00
 -0.1885111E-01        0.0000000E 00       0.1733184E 00        0.0000000E 00
 -0.1896896E-02        0.0000000E 00      -0.1076721E-01        0.0000000E 00

GRADIENT MATRIX
 -0.5431192E-03        0.3374061E-04

STABILITY INCREASE STEP SIZE =        0.24275460

GAIN MATRIX K
  0.4054773E 03      -0.4449835E 02

ROOTS     REAL PART             IMAG. PART
       -0.586954E 01          0.679102E 01
       -0.586954E 01         -0.679102E 01
        0.137404E 01          0.644784E 01
        0.137404E 01         -0.644784E 01
       -0.313356E 01          0.000000E 00
        0.199437E 01          0.000000E 00
       -0.442964E-01          0.000000E 00

CONVEX CURVE FIT TO STABILITY LOCUS

GAIN MATRIX K
 -0.1626357E 02      -0.1829822E 02

ROOTS     REAL PART             IMAG. PART
       -0.483643E 01          0.549917E 01
       -0.483643E 01         -0.549917E 01
       -0.307465E-01          0.612206E 01
       -0.307465E-01         -0.612206E 01
       -0.157808E 00          0.395643E 00
       -0.157808E 00         -0.395643E 00
       -0.124568E 00          0.000000E 00
```

FIGURE II.4-1 (Cont'd)

```
STEP SIZE TOO LARGE. PROGRAM CONTINUES WITH REDUCED STEP SIZE

CONVEX CURVE FIT TO STABILITY LOCUS

GAIN MATRIX K
  -0.2936047E 02    -0.1748459E 02

ROOTS      REAL PART              IMAG. PART
         -0.480006E 01           0.542807E 01
         -0.480006E 01          -0.542807E 01
         -0.107239E 00           0.612655E 01
         -0.107239E 00          -0.612655E 01
         -0.144420E 00           0.728661E 00
         -0.144420E 00          -0.728661E 00
         -0.707709E-01           0.000000E 00

MAXIMUM REAL PART OF ROOTS
  -0.7077086E-01
```

FIGURE II.4-1 (Cont'd)

```
GAIN SEARCH PROCEEDS ALONG A NEW GRADIENT. ITERATION NUMBER     3

EIGVEC ERROR MESSAGES
SHI=0.2492E-02          ITER=    4          DIF=0.2384E-06

EIGENVECTORS CORRES TO MRP EIGENVALUE
    ROW REAL PART      ROW IMAG PART      COL REAL PART      COL IMAG PART
   0.1000000E 01      0.0000000E 00      0.3028423E 00      0.0000000E 00
   0.1424851E 00      0.0000000E 00     -0.2144598E-01      0.0000000E 00
  -0.9601080E 00      0.0000000E 00      0.1000000E 01      0.0000000E 00
  -0.1390047E-01      0.0000000E 00      0.9848484E 00      0.0000000E 00
  -0.2110891E-02      0.0000000E 00     -0.6974274E-01      0.0000000E 00
  -0.2838436E-01      0.0000000E 00      0.1727620E 00      0.0000000E 00
  -0.2858681E-02      0.0000000E 00     -0.1223427E-01      0.0000000E 00

GRADIENT MATRIX
  -0.1275508E-02      0.9032596E-04

STABILITY INCREASE STEP SIZE                 0.00433192

GAIN MATRIX K
  -0.2598119E 02     -0.1772389E 02

ROOTS      REAL PART              IMAG. PART
        -0.480912E 01          0.544784E 01
        -0.480912E 01         -0.544784E 01
        -0.870506E-01          0.612440E 01
        -0.870506E-01         -0.612440E 01
        -0.153032E 00          0.658613E 00
        -0.153032E 00         -0.658613E 00
        -0.759749E-01          0.000000E 00

STEP SIZE IS DOUBLED

GAIN MATRIX K
  -0.2260190E 02     -0.1796320E 02

ROOTS      REAL PART              IMAG. PART
        -0.481842E 01          0.546706E 01
        -0.481842E 01         -0.546706E 01
        -0.667294E-01          0.612201E 01
        -0.667294E-01         -0.612201E 01
        -0.159904E 00          0.580474E 00
        -0.159904E 00         -0.580474E 00
        -0.841529E-01          0.000000E 00
```


FIGURE II.4-1 (Cont'd)

```
STEP SIZE TOO LARGE. RETURN TO PREVIOUS BEST GAINS

GAIN MATRIX K
   0.2598119E 02    0.1772389E 02

ROOTS      REAL PART              IMAG. PART
         -0.480912E-01          0.544784E 01
         -0.480912E-01         -0.544784E 01
         -0.870506E-01          0.612440E 01
         -0.870506E-01         -0.612440E 01
         -0.153032E 00          0.658613E 00
         -0.153032E 00         -0.658613E 00
         -0.759749E-01          0.000000E 00
```

FIGURE II.4-1 (Cont'd)

```
STABILITY IMPROVEMENT RATE TOO SLOW. PROGRAM TERMINATED.

LAST RESULTS SHOW MOST STABLE CONDITION FOUND
COMPILE TIME=    22.97 SEC,EXECUTION TIME=    41.39 SEC,OBJECT COD=     40832 BYT
```

FIGURE II.4-1 (Cont'd)

In order to verify that GRADGN did indeed attain a stability maximum, the eigenvalues of the system described by (II.4-1) to (II.4-4) were obtained for a matrix of feedback gains. The values of the real parts of the least stable eigenvalues were cross-plotted against the feedback gains, resulting in the equi-stability contours shown in Figure II.4-2. The gain trajectory produced by GRADGN is superimposed on Figure II.4-2. The steepest descent nature of GRADGN, and the fact that a stability maximum was achieved (within the limits of the diminishing returns termination) are evident from Figure II.4-2.

II.5 Comments on Gain Initialization

The digital computer program GRADGN clearly provides a way to improve the stability of a system by output feedback. As such it is a useful adjunct to the output feedback optimization techniques of Cassidy and Levine. It must be remembered, however, that GRADGN is designed to find purely local stability maxima. Thus it may on occasion fail to find feedback gains giving sufficient stability for initialization of the optimization algorithms, even though such gains actually exist. In an attempt to ameliorate this problem, provision has been included in GRADGN for initializing its gain search at any desired location. Thus any desired volume of gain space may be searched by initializing GRADGN at a suitable number of discrete points.

Since GRADGN is designed to perform a similar function to Koenigsberg's[21] algorithm, a comparison of the two technique's may be pertinent. Both are gradient techniques and both use adaptive step size variation, although Koenigsberg's adaptation procedure is much simpler than that used in GRADGN. A major difference occurs in the computation of the gradient. Koenigsberg's

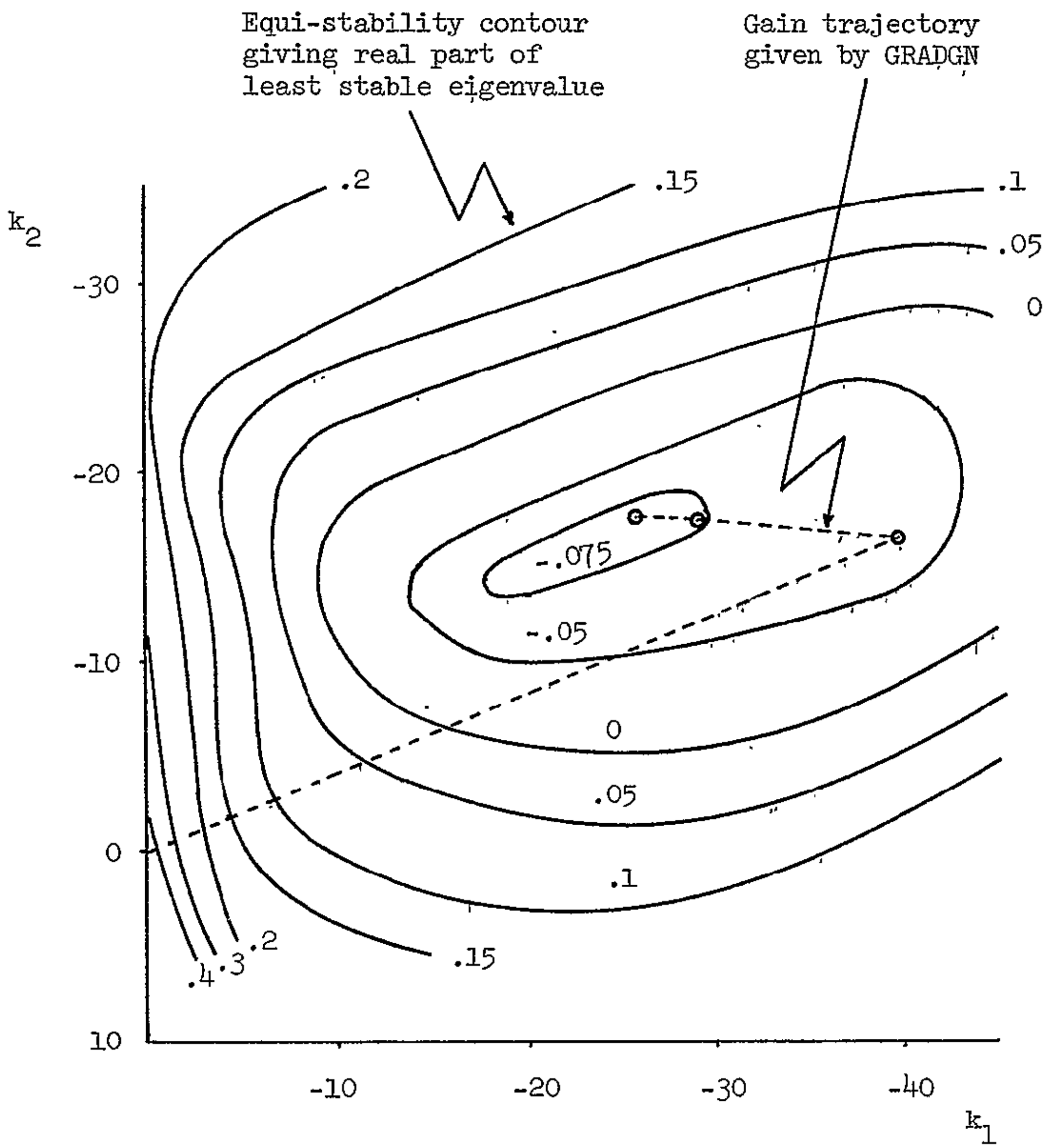


FIGURE II.4-2 Demonstration of GRADGN's Performance on Example Problem gradient computation is based on Reddy's[30] expression

$$\frac{\partial \lambda_k}{\partial \alpha_p} = \left. \frac{\text{Tr (conjoint } [A] \left[\frac{\partial A}{\partial \alpha_p}\right])}{\text{Tr (conjoint } [A])} \right|_{\lambda = \lambda_k} \tag{II.5-1}$$

for the sensitivity of the eigenvalue λ_k of the matrix A to the matrix parameter α_p. The conjoint is defined by

$$\text{conjoint } [A] = \text{adjoint } [A - \lambda I] \tag{II.5-2}$$

The equivalent expression used by GRADGN is (from (II.2-5))

$$\frac{\partial \lambda_k}{\partial \alpha_p} = \frac{\underline{v}^T \frac{\partial A}{\partial \alpha_p} \underline{w}}{\underline{v}^T \underline{w}} \tag{II.5-3}$$

where $\underline{v}$ and $\underline{w}$ respectively are the row and column eigenvectors of the matrix A corresponding to the eigenvalue λ_k. Equation (II.5-3) may be expanded to

$$\frac{\partial \lambda_k}{\partial \alpha_p} = \frac{\sum_{i=1}^{NS} \sum_{j=1}^{NS} v_i \left[\frac{\partial A}{\partial \alpha_p}\right]_{ij} w_j}{\sum_{i=1}^{NS} \sum_{j=1}^{NS} v_i \; w_j} \tag{II.5-4}$$

which may be reordered to

$$\frac{\partial \lambda_k}{\partial \alpha_p} = \frac{\sum_{j=1}^{NS} \sum_{i=1}^{NS} w_j \, v_i \left[\frac{\partial A}{\partial \alpha_p}\right]_{ij}}{\sum_{j=1}^{NS} \sum_{i=1}^{NS} w_j \; v_i} \tag{II.5-5}$$

Inspection of (II.5-5) shows it to be the same as

$$\frac{\partial \lambda_k}{\partial \alpha_p} = \frac{\mathrm{tr}\left[\underline{w}\,\underline{v}^T \frac{\partial A}{\partial \alpha_p}\right]}{\mathrm{tr}\left[\underline{w}\,\underline{v}^T\right]} \qquad \text{(II.5-6)}$$

Now the determination of a column eigenvector of a matrix may be achieved by the computation of only one column of the conjoint. Van Ness'[29] EIGVEC, used for eigenvector computation in GRADGN, gives both row and column eigenvectors with about the same computational effort as is required for only a column eigenvector. Thus (II.5-6) is computationally more efficient than (II.5-1). However (II.5-6) is simply (II.5-3) with each vector inner product replaced by the trace of a vector outer product, so that (II.5-3) is clearly preferable to (II.5-6) especially for systems of high order.

CHAPTER III

THE FINITE TIME INTERVAL PROBLEM

III.1 Introduction

In this Chapter we are concerned with the determination of the optimal time invariant output feedback controller for the system

$$\underline{\dot{x}}(t) = A\,\underline{x}(t) + B\,\underline{u}(t) \tag{III.1-1}$$

where $\underline{x}(t)$ is a NS vector

$\underline{u}(t)$ is a NC vector

with the cost functional

$$J = \underline{x}^T(T)\,F\,\underline{x}(T) + \int_0^T \underline{x}^T(\tau)\,Q\,\underline{x}(\tau) + \underline{u}^T(\tau)\,R\,\underline{u}(\tau)\,d\tau \tag{III.1-2}$$

where T is a fixed finite time and F, Q, R are suitably positive (semi) definite. Thus we require

$$\underline{u}(t) = -K^T\,\underline{x}(t) \tag{III,1-3}$$

and constrain K to be of the form

$$K = \begin{array}{c} \overbrace{\phantom{k_{ij}}}^{NC} \\ \left[\begin{array}{c} k_{ij} \\ \hdashline 0 \end{array}\right] \end{array} \begin{array}{l} \}\ NF \\ \\ \end{array} \Bigg\}\ NS \tag{III.1-4}$$

Two items set this problem apart from the normal linear-quadratic state regulator problem. These are the requirement for output feedback control and the requirement for the time-invariant feedback gains. It is remarkable that each of these requirements has been individually satisfied by different investigators, but by the use of similar techniques. Levine[19] used the Initial State Averaging (I.S.A.) technique to determine the optimal time-variable output feedback controller for the system (III.1-1) with the cost functional (III.1-2). On the other hand Kleinman, Fortmann and Athans[25] used I.S.A. to determine the optimal time-invariant allstate feedback controller. This report considers the use of I.S.A. to solve both requirements simultaneously.

The theory underlying the I.S.A. approach is presented in Section III.2

III.2 Theory of the Initial State Averaging Technique

The solution to the problem posed by (III.1-1) to (III.1-3) is underdefined in the sense that the optimal controller is a function of the undefined initial condition $\underline{x}(0)$. In general each different initial condition requires a different set of feedback gains. Since no single controller can minimize the cost functional for all initial conditions it seems wise to seek that controller which minimizes the expected value of the cost functional. This is the rationale behind I.S.A.

From (III.1-1) and (III.1-3) we see that

$$\dot{\underline{x}}(t) = \hat{A}\,\underline{x}(t) \qquad \text{(III.2-1)}$$

where

$$\hat{A} = \left[A - BK^T \right] \qquad \text{(III.2-1)}$$

Thus for a given initial condition $\underline{x}(0)$

$$\underline{x}(t) = \phi(t)\,\underline{x}(0) \qquad \text{(III.2-3)}$$

where $\phi(t)$ is the state transition matrix of $\hat{A}$ and is defined by

$$\dot{\phi}(t) = \hat{A}\,\phi(t) \qquad \text{(III.2-4)}$$

$$\phi(0) = I \tag{III.2-5}$$

Since we are dealing only with linear time-invariant systems we have[1]

$$\phi(t) = \exp\left[\hat{A}t\right] \tag{III.2-6}$$

Repeating (III.1-2)

$$J = \underline{x}^T(T)\ F\ \underline{x}(T) + \int_0^T \underline{x}^T(\tau)\ Q\ \underline{x}(\tau) + \underline{u}^T(\tau)\ R\ \underline{u}(\tau)\ d\tau \tag{III.2-7}$$

Using (III.1-3) reduces this to

$$J = \underline{x}^T(T)\ F\ \underline{x}(T) + \int_0^T x^T(\tau)\left[Q + KRK^T\right]\underline{x}(\tau)\ d\tau \tag{III.2-8}$$

whence, by (III.2-3)

$$J = \underline{x}(0)^T\ \phi^T(T)\ F\ \phi(T)\ \underline{x}(0) + \int_0^T \underline{x}^T(0)\ \phi^T(\tau)\left[Q + KRK^T\right]$$

$$\phi(\tau)\ \underline{x}(0)\ d\tau \tag{III.2-9}$$

Moving the time invariant $\underline{x}(0)$ outside the integration reduces (III.2-9) to

$$J = \underline{x}(0)^T\left[\phi^T(T)\ F\ \phi(T) + \int_0^T \phi^T(\tau)\left[Q + KRK^T\right]\phi(\tau)\ d\tau\right]\underline{x}(0) \tag{III.2-10}$$

which may be rewritten as

$$J = \underline{x}^T(0)\ P\ \underline{x}(0) \tag{III.2-11}$$

where

$$P = \phi^T(T)\ F\ \phi(T) + \int_0^T \phi^T(\tau) \left[Q + KRK^T\right] \phi(\tau)\ d\tau \tag{III.2-12}$$

Thus

$$E\left[J\right] = E\left[x^T(0)\ P\ x\ (0)\right] \tag{III.2-13}$$

i.e.,

$$E\left[J\right] = E\left[\sum_{i=1}^{NS} \sum_{j=1}^{NS} x_i(0)\ P_{ij}\ x_j(0)\right] \tag{III.2-14}$$

Now consider the $x_i(0)$, $i = 1, \ldots, NS$ to be random variable whose distributions are dependent on the distribution of the initial conditions in state space. Then $E\left[J\right]$ is given by (III.2-14) as the expected value of a finite sum of random variables. But the expected value of a finite sum of random variables is equal to the sum of the expected values. Thus

$$E\left[J\right] = \sum_{i=1}^{NS} \sum_{j=1}^{NS} E\left[x_i(0)\ P_{ij}\ x_j(0)\right] \tag{III.2-15}$$

Rearranging (III.2-15) gives

$$E\left[J\right] = \sum_{i=1}^{NS} \sum_{j=1}^{NS} E\left[P_{ij}\ x_i(0)\ x_j(0)\right] \tag{III.2-16}$$

It is clear from (III.2-12) that the P_{ij} in (III.2-16) do not depend on the $x_i(0)$ or the $x_j(0)$. Thus (III.2-16) can be rewritten as

$$E\left[J\right] = \sum_{i=1}^{NS} \sum_{j=1}^{NS} P_{ij}\ E\left[x_j(0)\ x_i(0)\right] \tag{III.2-17}$$

Suppose now that each of the $x_i(0)$ has zero mean and that we can estimate the covariance matrix V of the initial conditions in state space, i.e.,

$$V_{ij} = E\left[x_i(0)\ x_j(0)\right] \tag{III.2-18}$$

Then, from (III.2-17) and (III.2-18)

$$E\left[J\right] = \sum_{i=1}^{NS} \sum_{j=1}^{NS} P_{ij}\ V_{ij} \tag{III.2-19}$$

whence

$$E\left[J\right] = \mathrm{Tr}\left[PV\right] \tag{III.2-20}$$

Now consider that the system described by (III.1-1), (III.1-3) and (III.1-4) is linear. Thus superposition holds and the system response can be scaled up or down with the magnitude of the initial conditions. This implies that the response of the system to an initial condition lying on the surface of the unit hypersphere in state space characterizes the response to any colinear initial condition state vector. Furthermore it is clear from (III.1-2) that if any initial condition is ratioed by an amount r then the cost is ratioed by r^2. These two facts imply that the costs associated with the set of initial conditions lying on the surface of the unit hypersphere is state space can be used to conveniently characterize the costs for all initial conditions. It is shown below that with little loss of generality attention may be confined to a uniform distribution of the initial conditions on the surface of the unit hypersphere.

Suppose that the covariance matrix V is non-singular with inverse V^{-1}. Suppose also that the square root of V^{-1} exists, and let it be denoted by $V^{-1/2}$. Consider the description of the system (III.2-1) in terms of a new set of state variables $\underline{y}(t)$ defined by

$$\underline{y}(t) = (NS)^{-1/2} \; V^{-1/2} \; \underline{x}(t) \tag{III.2-21}$$

Then (III.2-1) becomes

$$(NS)^{1/2} \; V^{1/2} \; \dot{\underline{y}}(t) = \hat{A}(NS)^{1/2} \; V^{1/2} \; \underline{y}(t) \tag{III.2-22}$$

i.e.,

$$\dot{y}(t) = V^{-1/2} \; \hat{A} \; V^{1/2} \; \underline{y}(t) \tag{III.2-23}$$

which is similar in form to (III.2-1).

Also we find that

$$E\left[\underline{y}(0) \; \underline{y}^T(0)\right] = E\left[NS^{-1/2} \, V^{-1/2} \, \underline{x}(0) \; \underline{x}^T(0) \; (NS)^{-1/2} \, V^{-1/2}\right] \tag{III.2-24}$$

Thus

$$E\left[\underline{y}(0) \; \underline{y}^T(0)\right]_{ij} = (NS)^{-1} \, E\left[\sum_{p=1}^{NS} \; \sum_{q=1}^{NS} \; \left[V^{-1/2}\right]_{ip} x_p(0) \; x_q(0)\left[V^{-1/2}\right]_{qj}\right] \tag{III.2-25}$$

Interchanging finite sum and expectation gives

$$E\left[\underline{y}(0) \; \underline{y}^T(0)\right]_{ij} = (NS)^{-1} \sum_{p=1}^{NS} \; \sum_{q=1}^{NS} \; E\left[\left[V^{-1/2}\right]_{ip} x_p(0) \; x_q(0) \left[V^{-1/2}\right]_{qj}\right] \tag{III.2-26}$$

$$= (NS)^{-1} \sum_{p=1}^{NS} \; \sum_{q=1}^{NS} \; \left[V^{-1/2}\right]_{ip} V_{pq} \left[V^{-1/2}\right]_{qj} \tag{III.2-27}$$

$$= \frac{1}{NS} \left[V^{-1/2} \ V \ V^{-1/2} \right]_{ij} \tag{III.2-28}$$

$$= \frac{1}{NS} \left[I \right]_{ij} \tag{III.2-29}$$

where I is the identity matrix of order NS.

Thus the covariance matrix of the initial state vector $\underline{y}(0)$ is compatible with a uniform distribution of the $\underline{y}(0)$ vector on the surface of the unit hypersphere. Since such a distribution is convenient and since we can, within the limitations of the non-stringent assumptions made above, transform our initial state vector so that it is compatible with this distribution, we shall hereafter assume that

$$V = \frac{1}{NS} \ I \tag{III.2-30}$$

Thus, from (III.2-20) we see that

$$E \left[J \right] = \frac{1}{NS} \ \mathrm{tr} \ \left[P \right] \tag{III.2-31}$$

and our I.S.A. optimal control is that which minimizes $\frac{1}{NS} \mathrm{tr} \left[P \right]$ where P is given by (III.2-12).

III.3 The Initial State Averaging Algorithm

The objective of the I.S.A. algorithm is to determine that gain matrix K which minimizes the average cost of control as given by (III.2-31) under the constraint that K must be of the form given by (III.1-4). Thus we wish to minimize the gain function GF given by

$$GF = \frac{1}{NS} \ \mathrm{tr} \left[\phi^T(T) F \ \phi(T) + \int_0^T \phi^T(\tau) \ \left[Q + KRK^T \right] \ \phi(\tau) \ d\tau \right] \tag{III.3-1}$$

Now for any two matrices A and B of suitable dimensions we have the trace identity

$$\mathrm{tr}\left[AB\right] = \mathrm{tr}\left[BA\right] \qquad \text{(III.3-2)}$$

Using (III.3-2) and the fact that the integral of a matrix is the matrix of the integrals of the elements reduces (III.3-1) to

$$GF = \frac{1}{NS}\ \mathrm{tr}\left[F\ \phi(T)\ \phi^T(T) + (Q + KRK^T)\int_0^T \phi(\tau)\ \phi^T(\tau)\ d\tau\right] \qquad \text{(III.3-3)}$$

Since the variable elements of the gain matrix K are unconstrained the necessary conditions for K to minimize GF are given by

$$\frac{\partial GF}{\partial k_{ij}} = 0 \qquad \text{(III.3-4)}$$

$i = 1, \ldots, NF$

$j = 1, \ldots, NC$

Expanding (III.3-4) gives the necessary conditions as

$$\mathrm{tr}\left[F\left[\frac{\partial \phi}{\partial k_{ij}}\ \phi^T + \phi\frac{\partial \phi^T}{\partial k_{ij}}\right]_T + \left[\frac{\partial K}{\partial k_{ij}}\ R\ K^T + KR\ \frac{\partial K^T}{\partial k_{ij}}\right]\int_0^T \phi\ \phi^T\ d\tau\right.$$

$$\left. + \left[Q + KRK^T\right]\int_0^T \frac{\partial \phi}{\partial k_{ij}}\ \phi^T + \phi\ \frac{\partial \phi^T}{\partial k_{ij}}\ d\tau\right] = 0 \qquad \text{(III.3-5)}$$

where the arguments of the ϕ functions have been dropped for clarity.

Now for any matrices A and B

$$\mathrm{tr}\left[A\right] = \mathrm{tr}\left[A^T\right] \qquad \text{(III.3-6)}$$

and

$$\mathrm{tr}\left[A + B\right] = \mathrm{tr}\left[A\right] + \mathrm{tr}\left[B\right] \qquad \text{(III.3-7)}$$

Thus in (III.3-5)

$$\mathrm{tr}\left[F\left[\frac{\partial \phi}{\partial k_{ij}}\phi^T + \phi\frac{\partial \phi^T}{\partial k_{ij}}\right]\right] = \mathrm{tr}\left[F\frac{\partial \phi}{\partial k_{ij}}\phi^T\right] + \mathrm{tr}\left[F\phi\frac{\partial \phi^T}{\partial k_{ij}}\right]$$

by (III.3-7)

$$= \mathrm{tr}\left[F\frac{\partial \phi}{\partial k_{ij}}\phi^T\right] + \mathrm{tr}\left[\frac{\partial \phi}{\partial k_{ij}}\phi^T F\right]$$

by (III.3-6)

$$= \mathrm{tr}\left[F\frac{\partial \phi}{\partial k_{ij}}\phi^T\right] + \mathrm{tr}\left[F\frac{\partial \phi}{\partial k_{ij}}\phi^T\right]$$

by (III.3-2)

$$= 2\,\mathrm{tr}\left[F\frac{\partial \phi}{\partial k_{ij}}\phi^T\right] \qquad \text{(III.3-8)}$$

Similarly

$$\mathrm{tr}\left[\left[\frac{\partial K}{\partial k_{ij}}RK^T + KR\frac{\partial K^T}{\partial k_{ij}}\right]\int_0^T \phi\,\phi^T\,d\tau\right] = 2\,\mathrm{tr}\left[\frac{\partial K}{\partial k_{ij}}RK^T\int_0^T \phi\,\phi^T\,d\tau\right]$$

(III.3-9)

and

$$\mathrm{tr}\left[\left[Q + KRK^T\right] \int_0^T \frac{\partial \phi}{\partial k_{ij}} \phi^T + \phi \frac{\partial \phi^T}{\partial k_{ij}} \, d\tau \right] = 2 \, \mathrm{tr}\left[\left[Q + KRK^T\right] \int_0^T \phi \frac{\partial \phi^T}{\partial k_{ij}} \, d\tau \right]$$

(III.3-10)

Thus the necessary conditions of (III.3-5) reduce to

$$W = 0 \tag{III.3-11}$$

where the NF x NC matrix W has its elements defined by

$$w_{ij} = \mathrm{tr}\left[F \phi \frac{\partial \phi^T}{\partial k_{ij}} \Bigg|_T + \left[Q + KRK^T\right] \int_0^T \phi \frac{\partial \phi^T}{\partial k_{ij}} \, d\tau \right.$$

$$\left. + \frac{\partial K}{\partial k_{ij}} RK^T \int_0^T \phi \, \phi^T \, d\tau \right] \tag{III.3-12}$$

Equations (III.3-11) and (III.3-12) represent a set of NF x NC simultaneous equations in the NF x NC variable elements of the gain matrix K. Solution of these equations gives candidates for the otpimal controller. The solution is performed by Newton-Raphson iteration.

Consider the rearrangement of W into an NF x NC vector "W" by column ordering

i.e., $$w_{ij} = \text{"W"}_{NF(j-1)+i} \tag{III.3-13}$$

The variable elements of K may be similarly ordered, giving the gain vector "K" where

$$K_{ij} = \text{"K"}_{NF(j-1)+i} \tag{III.3-14}$$

Suppose that "K(n)" is the gain vector at the n^{th} iteration of the Newton-Raphson procedure, and that "W(n)" is the corresponding vector of necessary conditions. The Newton-Raphson technique computes $''K(n+1)''$, an improved estimate of a gain vector satisfying the necessary conditions, by

$$''K(n+1)'' = ''K(n)'' - \mu \left[\nabla(n) \right]^{-1} ''W(n)'' \qquad \text{(III.3-15)}$$

where $\nabla(n)$ is the (NF x NC) x (NF x NC) gradient matrix given by

$$\nabla(n)_{ij} = \frac{\partial ''W(n)''_i}{\partial ''K(n)''_j} \qquad \text{(III.3-16)}$$

and μ is a convergence factor .

It can be seen from (III.3-13) and (III.3-14) that computation of $\nabla(n)$ is equivalent to computation of $\frac{\partial w_{gh}}{\partial k_{pq}}$ for $g = 1, \ldots, NF$, $h = 1, \ldots, NC$; $p = 1, \ldots, NF$; $q = 1, \ldots, NC$. From (III.3-12) we have

$$\begin{aligned}
\frac{\partial w_{gh}}{\partial k_{pq}} = \mathrm{tr}\Bigg[& F \left[\frac{\partial \phi}{\partial k_{pq}} \frac{\partial \phi^T}{\partial k_{gh}} + \phi \frac{\partial^2 \phi^T}{\partial k_{pq} \partial k_{gh}} \right]_T \\
& + \left[Q + KRK^T \right] \int_0^T \frac{\partial \phi}{\partial k_{pq}} \frac{\partial \phi^T}{\partial k_{gh}} + \phi \frac{\partial^2 \phi^T}{\partial k_{pq} \partial k_{gh}} \, d\tau \\
& + \left[\frac{\partial K}{\partial k_{pq}} RK^T + KR \frac{\partial K^T}{\partial k_{pq}} \right] \int_0^T \phi \frac{\partial \phi^T}{\partial k_{gh}} \, d\tau + \frac{\partial^2 K}{\partial k_{pq} \partial k_{gh}} RK^T \int_0^T \phi \, \phi^T \, d\tau \\
& + \frac{\partial K}{\partial k_{gh}} R \left[\frac{\partial K^T}{\partial k_{pq}} \int_0^T \phi \, \phi^T \, d\tau + K^T \int_0^T \frac{\partial \phi}{\partial k_{pq}} \phi^T + \phi \frac{\partial \phi^T}{\partial k_{pq}} \, d\tau \right] \Bigg]
\end{aligned}$$

(III.3-17)

Clearly

$$\frac{\partial^2 K}{\partial k_{pq} \partial k_{gh}} = 0 \tag{III.3-18}$$

Also, by (III.3-6) and (III.3-2)

$$\operatorname{tr}\left[KR \frac{\partial K^T}{\partial k_{pq}} \int_0^T \phi \frac{\partial \phi^T}{\partial k_{gh}} d\tau \right] = \operatorname{tr}\left[\int_0^T \frac{\partial \phi}{\partial k_{gh}} \phi^T d\tau \frac{\partial K}{\partial k_{pq}} RK^T \right]$$

$$= \operatorname{tr}\left[\frac{\partial K}{\partial k_{pq}} RK^T \int_0^T \frac{\partial \phi}{\partial k_{gh}} \phi^T d\tau \right] \tag{III.3-19}$$

so that (III.3-17) reduces to

$$\frac{\partial w_{gh}}{\partial k_{pq}} = \operatorname{tr}\left[F \left[\frac{\partial \phi}{\partial k_{pq}} \frac{\partial \phi^T}{\partial k_{gh}} + \phi \frac{\partial^2 \phi^T}{\partial k_{pq} \partial k_{gh}} \right]_T \right.$$

$$+ \left[Q + KRK^T \right] \int_0^T \frac{\partial \phi}{\partial k_{pq}} \frac{\partial \phi^T}{\partial k_{gh}} + \phi \frac{\partial^2 \phi^T}{\partial k_{pq} \partial k_{gh}} \, d\tau$$

$$+ \frac{\partial K}{\partial k_{pq}} RK^T \int_0^T \frac{\partial \phi}{\partial k_{gh}} \phi^T + \phi \frac{\partial \phi^T}{\partial k_{gh}} \, d\tau$$

$$\left. + \frac{\partial K}{\partial k_{gh}} RK^T \int_0^T \frac{\partial \phi}{\partial k_{pq}} \phi^T + \phi \frac{\partial \phi^T}{\partial k_{pq}} \, d\tau + \frac{\partial K}{\partial k_{gh}} R \frac{\partial K^T}{\partial k_{pq}} \int_0^T \phi \phi^T d\tau \right] \tag{III.3-20}$$

From (III.3-3), (III.3-12) and (III.3-20) it can be seen that computation of the cost GF, the necessary conditions vector "W" and the gradient matrix ∇ for a given matrix K is dependent on the computation of a number of functions of the state transition matrix $\phi(t)$. Basic to evaluation of these functions is evaluation of $\frac{\partial \phi(t)}{\partial k_{pq}}$ and $\frac{\partial^2 \phi(t)}{\partial k_{pq} \partial k_{gh}}$ for p=1,...,NF; q=1,...,NC; g=1,...,NF; h=1,...,NC.

Consider the state transition matrix $\phi_{\hat{K}}(t)$ corresponding to a gain matrix $\hat{K}$ given by

$$\hat{K} = K_0 + \frac{\partial K}{\partial k_{pq}} \delta k_{pq} + \frac{\partial K}{\partial k_{gh}} \delta k_{gh} \tag{III.3-21}$$

where δk_{pq} and δk_{gh} are small. By (III.2-4) $\phi_{\hat{K}}(t)$ can be computed as the solution to

$$\frac{d\, \phi_{\hat{K}}(t)}{dt} = \left[A - BK_0^T - B\left(\frac{\partial K^T}{\partial k_{pq}} \delta k_{pq} + \frac{\partial K^T}{\partial k_{gh}} \delta k_{gh}\right)\right] \phi_{\hat{K}}(t) \tag{III.3-22}$$

with

$$\phi_{\hat{K}}(0) = I \tag{III.3-23}$$

Thus

$$\frac{d\, \phi_{\hat{K}}(t)}{dt} = \left[A-BK_0^T\right] \phi_{\hat{K}}(t) - B\left[\frac{\partial K^T}{\partial k_{pq}} \delta k_{pq} + \frac{\partial K^T}{\partial k_{gh}} \delta k_{gh}\right] \phi_{\hat{K}}(t) \tag{III.3-24}$$

Treating the second term in (III.3-24) as an external input and using (III.2-6)

$$\phi_{\hat{K}}(t) = \exp\left[\left[A-BK_0^T\right] t\right] - \int_0^T \exp\left[\left[A-BK_0^T\right] (t-\tau)\right] B$$

$$\left[\frac{\partial K^T}{\partial k_{pq}} \delta k_{pq} + \frac{\partial K^T}{\partial k_{gh}} \delta k_{gh}\right] \phi_{\hat{K}}(\tau)\, d\tau \tag{III.3-25}$$

We now substitute the function described by (III.3-25) for $\phi_{\hat{K}}(\tau)$ in (III.3-25). This is similar to the method for computation of the matizant given in DeRusso, Roy and Close[1], and results in

$$\phi_{\hat{K}}(t) = \phi_{K_o}(t) - \int_0^t \phi_{K_o}(t-\tau)\, B \left[\frac{\partial K^T}{\partial k_{pq}} \, \delta k_{pq} + \frac{\partial K^T}{\partial k_{gh}} \, \delta k_{gh} \right]$$

$$\left[\phi_{K_0}(\tau) - \int_0^\tau \phi_{K_0}(\tau - s)\, B \left[\frac{\partial K^T}{\partial k_{pq}} \, \delta k_{pq} + \frac{\partial K^T}{\partial k_{gh}} \, \delta k_{gh} \right] \phi_{\hat{K}}(s)\, ds \right] d\tau$$

(III.3-26)

where $\phi_{K_0}(t)$ is the state transition matrix corresponding to the gain matrix K_0. Thus, to second order in δk_{pq} and δk_{gh},

$$\phi_{\hat{K}}(t) = \phi_{K_0}(t) - \int_0^t \phi_{K_0}(t-\tau)\, B \left[\frac{\partial K^T}{\partial k_{pq}} \, \delta k_{pq} + \frac{\partial K^T}{\partial k_{gh}} \, \delta k_{gh} \right]$$

$$\left[\phi_{K_0}(\tau) - \int_0^\tau \phi_{K_0}(\tau - s)\, B \left[\frac{\partial K^T}{\partial k_{pq}} \, \delta k_{pq} + \frac{\partial K^T}{\partial k_{gh}} \, \delta k_{gh} \right] \phi_{K_0}(s)\, ds \right] d\tau$$

(III.3-27)

Expansion of (III.3-27) gives

$$\phi_{\hat{K}}(t) = \phi_{K_0}(t) - \left[\int_0^t \phi_{K_0}(t-\tau)\, B \frac{\partial K^T}{\partial k_{pq}} \phi_{K_0}(\tau)\, d\tau \right] \delta k_{pq}$$

$$- \left[\int_0^t \phi_{K_0}(t-\tau)\, B \frac{\partial K^T}{\partial k_{gh}} \phi_{K_0}(\tau)\, d\tau \right] \delta k_{gh}$$

$$+\left[\int_0^t \phi_{K_0}(t-\tau)\, B\, \frac{\partial K^T}{\partial k_{pq}} \int_0^\tau \phi_{K_0}(\tau-s)\, B\, \frac{\partial K^T}{\partial k_{pq}}\, \phi_{K_0}(s)\, ds\, d\tau\right] \delta^2 k_{pq}$$

$$+\left[\int_0^t \phi_{K_0}(t-\tau)\, B\, \frac{\partial K^T}{\partial k_{pq}} \int_0^\tau \phi_{K_0}(\tau-s)\, B\, \frac{\partial K^T}{\partial k_{gh}}\, \phi_{K_0}(s)\, ds\, d\tau\right] \delta k_{pq}\, \delta k_{gh}$$

$$+\left[\int_0^t \phi_{K_0}(t-\tau)\, B\, \frac{\partial K^T}{\partial k_{gh}} \int_0^\tau \phi_{K_0}(\tau-s)\, B\, \frac{\partial K^T}{\partial k_{pq}}\, \phi_{K_0}(s)\, ds\, d\tau\right] \delta k_{pq}\, \delta k_{gh}$$

$$+\left[\int_0^t 0_{K_0}(t-\tau)\, B\, \frac{\partial K^T}{\partial k_{gh}} \int_0^\tau \phi_{K_0}(\tau-s)\, B\, \frac{\partial K^T}{\partial k_{gh}}\, \phi_{K_0}(s)\, ds\, d\tau\right] \delta^2 k_{gh}$$

(III.3-28)

Now consider that $\phi(t)$ can be expanded in a Taylor's series about K_0 giving

$$\phi_{\hat{K}}(t) = \phi_{K_0}(t) + \left.\frac{\partial \phi(t)}{\partial k_{pq}}\right|_{K_0} \delta k_{pq} + \left.\frac{\partial \phi(t)}{\partial k_{gh}}\right|_{K_0} \delta k_{gh}$$

$$+ \frac{1}{2!}\left[\left.\frac{\partial^2 \phi(t)}{\partial k_{pq}^2}\right|_{K_0} \delta^2 k_{pq} + 2 \left.\frac{\partial^2 \phi(t)}{\partial k_{pq}\, \partial k_{gh}}\right|_{K_0} \delta k_{pq}\, \delta k_{gh}\right.$$

$$\left. + \left.\frac{\partial^2 \phi(t)}{\partial k_{gh}^2}\right|_{K_0} \delta^2 k_{gh}\right].$$

+ terms of higher order (III.3-29)

Comparison of (III.3-28) and (III.3-29) gives

$$\left.\frac{\partial \phi(t)}{\partial k_{pq}}\right|_{K_0} = - \int_0^t \phi_{K_0}(t-\tau)\ B\ \frac{\partial K^T}{\partial k_{pq}}\ \phi_{K_0}(\tau)\ d\tau \qquad (III.3\text{-}30)$$

and

$$\left.\frac{\partial^2 \phi(t)}{\partial k_{pq}\ k_{gh}}\right|_{K_0} = \int_0^t \phi_{K_0}(t-\tau)\ B\ \frac{\partial K^T}{\partial k_{pq}} \int_0^\tau \phi_{K_0}(\tau-s)\ \frac{\partial K^T}{\partial k_{gh}}\ \phi_{K_0}(s)\ ds\ d\tau$$

$$+ \int_0^t \phi_{K_0}(t-\tau)\ B\ \frac{\partial K^T}{\partial k_{gh}} \int_0^\tau \phi_{K_0}(\tau-s)\ \frac{\partial K^T}{\partial k_{pq}}\ \phi_{K_0}(s)\ ds\ d\tau$$

(III.3-31)

The result given by (III.3-30) agrees with a similar expression presented by Levine and Athans[31] for the semi-infinite time interval problem.

We note from (III.3-30) and (III.3-31) that

$$\left.\frac{\partial^2 \phi(t)}{\partial k_{pq} \partial k_{gh}}\right|_{K_0} = - \int_0^t \phi_{K_0}(t-\tau)\ B\left[\left.\frac{\partial K^T}{\partial k_{pq}}\ \frac{\partial \phi(t)}{\partial k_{gh}}\right|_{K_0} + \left.\frac{\partial K^T}{\partial k_{gh}}\ \frac{\partial \phi(t)}{\partial k_{pq}}\right|_{K_0}\right] d\tau$$

(III.3-32)

The functions of the state transition matrix to be evaluated are now seen to be given by

$$\left.\phi\ \frac{\partial \phi^T}{\partial k_{pq}}\right|_T = - \phi(T) \int_0^T \phi^T(\tau)\ \frac{\partial K}{\partial k_{pq}}\ B^T\ \phi^T(T-\tau)\ d\tau \qquad (III.3\text{-}33)$$

$$\left.\frac{\partial \phi}{\partial k_{pq}}\ \frac{\partial \phi^T}{\partial k_{gh}}\right|_T = \int_0^T \phi(T-\tau)\ B\ \frac{\partial K^T}{\partial k_{pq}}\ \phi(\tau)\ d\tau \int_0^T \phi^T(s)\ \frac{\partial K}{\partial k_{gh}}\ B^T\phi^T(T-s)ds$$

(III.3-34)

$$\phi \frac{\partial^2 \phi^T}{\partial k_{pq} \partial k_{gh}}\Bigg|_T = -\phi(T) \int_0^T \left[\frac{\partial \phi^T(\tau)}{\partial k_{pq}} \frac{\partial K}{\partial k_{gh}} + \frac{\partial \phi^T(\tau)}{\partial k_{pq}} \frac{\partial K}{\partial k_{pq}}\right] B^T \phi(T-\tau) d\tau \tag{III.3-35}$$

$$\int_0^T \phi \frac{\partial \phi^T}{\partial k_{pq}} d\tau = - \int_0^T \phi(\tau) \int_0^\tau \phi^T(s) \frac{\partial K}{\partial k_{pq}} B^T \phi^T(\tau - s) \, ds \, d\tau \tag{III.3-36}$$

$$\int_0^T \frac{\partial \phi}{\partial k_{pq}} \frac{\partial \phi^T}{\partial k_{gh}} d\tau = \int_0^T \int_0^\tau \phi(\tau - \nu) \; B \frac{\partial K^T}{\partial k_{pq}} \phi(\nu) \, d\nu$$

$$\int_0^\tau \phi^T(s) \frac{\partial K}{\partial k_{gh}} B^T \phi^T(\tau - s) \, ds \, d\tau \tag{III.3-37}$$

and

$$\int_0^T \phi \frac{\partial^2 \phi^T}{\partial k_{pq} \partial k_{gh}} d\tau = - \int_0^T \phi(\tau) \int_0^\tau \left[\frac{\partial \phi(s)}{\partial k_{pq}} \frac{\partial k}{\partial k_{gh}} + \frac{\partial \phi^T(s)}{\partial k_{gh}} \cdot \frac{\partial K}{\partial k_{pq}}\right] B^T \phi(\tau - s) ds d\tau \tag{III.3-38}$$

Consider (III.3-33) as being representative of (III.3-33) through (III.3-38). Supposing $\hat{A}$ to have distinct eigenvalues we can evaluate $\phi(t)$ by

$$\phi(t) = M \exp\left[\Lambda t\right] M^{-1} \tag{III.3-39}$$

where M is the modal matrix of eigenvectors of $\hat{A}$

Λ is the diagonal matrix of eigenvalues of $\hat{A}$

Substitution of (III.3-39) in (III.3-33) yields

$$\phi \frac{\partial \phi^T}{\partial k_{pq}}\Bigg|_T = - M e^{\Lambda T} M^{-1} \int_0^T \left[M^{-1}\right]^T e^{\Lambda \tau} M^T \frac{\partial K}{\partial k_{pq}} B^T \left[M^{-1}\right]^T e^{(T-\tau)} M^T d\tau \tag{III.3-40}$$

ϴ ✓

Now $\frac{\partial K}{\partial k_{pq}}$ is simply an NS x NC matrix having a 1.0 in it pq position and zeros elsewhere. Thus the ij element of $\phi \frac{\partial \phi^T}{\partial k_{pq}}\Big|_T$ is given by

$$\left[\phi \frac{\partial \phi^T}{\partial k_{pq}}\Big|_T\right]_{ij} = - \sum_{\substack{n_1, n_2, n_3 \\ n_4, n_5 = 1}}^{NS} M_{in_1} \, e^{\lambda n_1 T} \, M^{-1}_{n_1 n_2} \int_0^T M^{-1}_{n_3 n_2} \, e^{\lambda n_3} \, M_{pn_3} \, B_{n_4 q}$$

$$M^{-1}_{n_5 n_4} \, e^{\lambda n_5 (T-\tau)} \, M_{jn_5} \, d\tau \qquad \text{(III.3-41)}$$

where λ_{n_1} is the n_1 th eigenvalue of A.

Performing the integration yields

$$\left[\phi \frac{\partial \phi^T}{\partial k_{pq}}\Big|_T\right]_{ij} = - \sum_{\substack{n_1, n_2, n_3 \\ n_4, n_5 = 1}}^{NS} M_{in_1} \, M^{-1}_{n_1 n_2} \, M^{-1}_{n_3 n_2} \, M_{pn_3} \, B_{n_4 q} \, M^{-1}_{n_5 n_4} \, M_{jn_5}$$

$$\left[\underset{n_5 = n_3}{\text{iff}} \; T \, e^{(\lambda n_1 + \lambda n_5)^T} + \underset{n_5 \neq n_3}{\text{iff}} \; \frac{e^{(\lambda n_1 + \lambda n_3)^T} - e^{(\lambda n_1 + \lambda n_5)^T}}{\lambda n_3 - \lambda n_5} \right] \qquad \text{(III.3-42)}$$

Thus (III.3-33) can be evaluated in terms of the eigenvalues and eigenvectors of A. Equations (III.3-34) through (III.3-38) can be similarly evaluated. The terms can then be combined by (III.3-3), (III.3-12) and (III.3-20) to give the average cost GF, the necessary conditions vector "W" and the gradient matrix ∇ respectively.

The only other item required for application of the Newton-Raphson iteration technique described by (III.3-15) is a value for the convergence factor μ. Determination of a suitable value of the convergence factor is described in Section III.4.

III.4 Mechanization of the Initial State Averaging Algorithm

The algorithm described above for computation of optimal constant output feedback gains has been mechanized in the digital computer program ISAFT. The program was written in Fortran IV for an IBM 360/50 computer. A listing is given in Appendix II. The program comprises a MAIN and several subprograms. The functions of the major subprograms are described briefly below.

MAIN: performs most input-output functions and acts as the controller for the other subprograms. The MAIN subprogram also computes a suitable value for the Newton-Raphson convergence factor μ (see (III.3-15) and controls program termination. The operation is illustrated by the data flowchart given in Figure III.4-1.

Computation of the convergence factor μ is performed adaptively. A trial step is made with μ set to 1.0. The value of the gain function GF for the new gains is compared with its former value. If a decrease has occurred the program proceeds to the next iteration. Otherwise the value of μ is halved. This process is repeated up to five times. If no improvement in the gain function has resulted by the fifth cycle, i.e., $\mu = 2^{-5}$, the program is terminated due to poor convergence.

Assuming no termination due to poor convergence, normal program termination will occur when the feedback gains approach their final values. The criterion for this termination is that the proportional change in each individual gain during a single iteration should be less than an amount GNSTOP. The parameter GNSTOP is input by the user.

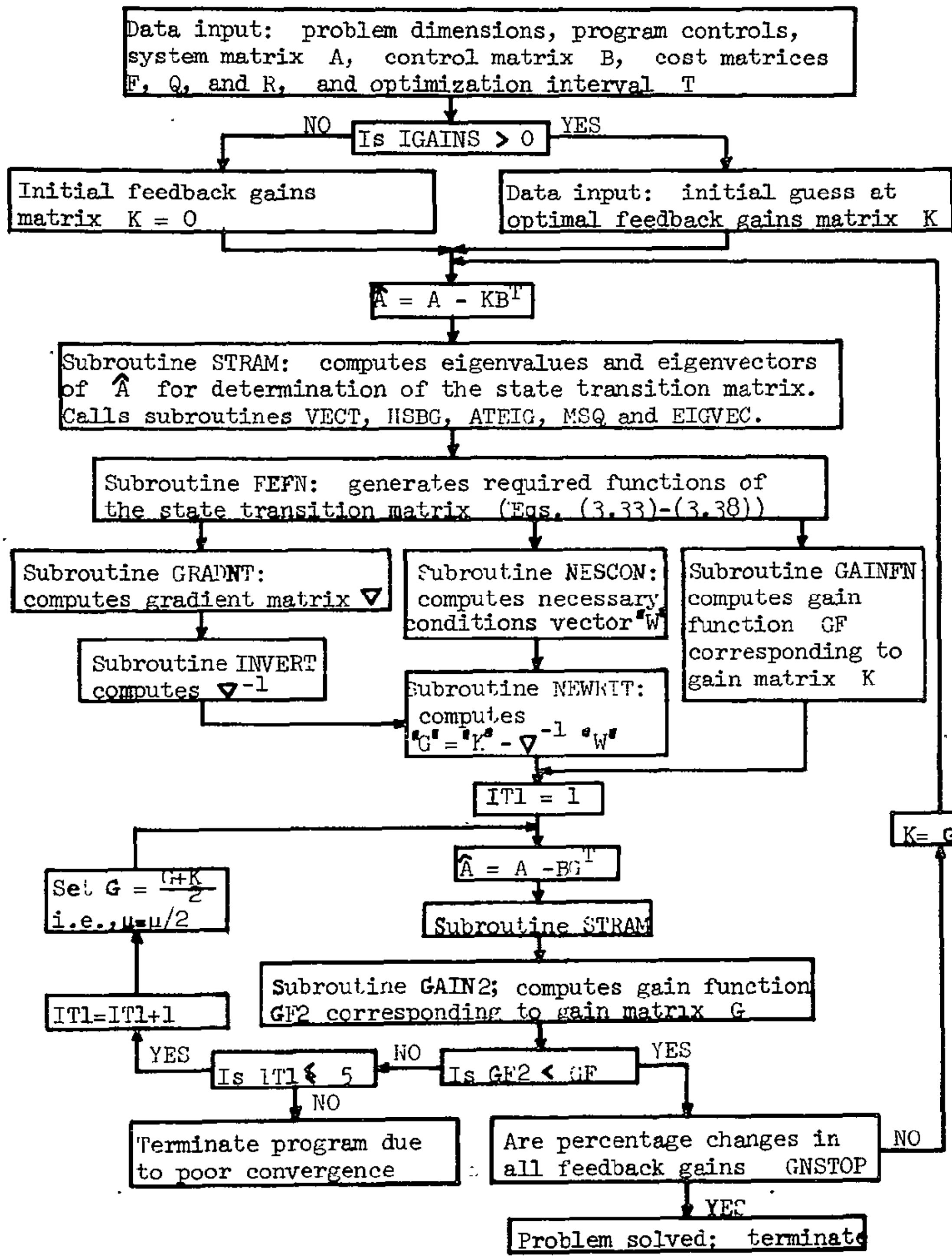


FIGURE III.4-1 Data Flowchart for Digital Computer Program ISAFT

STRAM: computes the eigenvalues and the model matrix of the matrix $\hat{A}$. The subprograms VECT, HSBG and ATEIG are used in the eigenvalue determination. The subprograms MSQ and EIGVEC are used to determine the model matrix and its inverse.

FEFN: computes the functions of the state transition matrix described by (III.3-33) through (III.3-38).

GAINFN: computes the value of the gain function GF at the beginning of each iteration - see (III.3-3).

NESCON: computes the necessary conditions vector "W" - see (III.3-12).

GRADNT: computes the gradient matrix ∇ - see (III.3-20).

INVERT: inverts the gradient matrix

NEWRIT: performs one iteration of the Newton-Raphson algorithm - see (III.3-15).

GAIN2: computes the value of the gain function GF at the end of each iteration for use in the determination of a suitable convergence factor μ - see (III.3-15).

III.5 Examples of the Use of the Optimization Program ISAFT

Three examples are given to illustrate the use of the digital computer program ISAFT. In keeping with the expository nature of this Chapter the examples are all short and are designed to illustrate different aspects of the computations.

Example 1

This example considered the case where all system states were available for feedback. The system equations were

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \qquad \text{(III.5-1)}$$

and the cost functional was

$$J = \underline{x}^T(1) \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \underline{x}(1) + \int_0^1 \underline{x}^T(\tau) \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \underline{x}(\tau) + 0.1\, u^2(\tau)\, d\tau \qquad \text{(III.5-2)}$$

The control was constrained to be of the form

$$u(t) = -k_1\, x_1(t) - k_2 \dot{x}_2(t) \qquad \text{(III.5-3)}$$

It was assumed that there was no prior knowledge about the system, and the initial feedback gains were set to the default value of zero. The computer print out is given in Figure III.5-1. The optimal feedback gains were computed to be

$$k_1 = 2.30 \qquad \text{(III.5-4)}$$

$$k_2 = 3.42 \qquad \text{(III.5-5)}$$

To verify that the values given by (III.5-4) and (III.5-5) were indeed optimal the results were checked by independent means. A matrix of feedback gains in the vicinity of the solution gains was chosen. At each set of gains the system described by (III.5-1) and (III.5-3) was simulated over the optimization time interval, starting from a number of different initial conditions, and the costs were computed numerically using (III.5-2). The average cost was then computed for each set of feedback gains. The initial conditions were chosen to be equally spaced around the unit circle in state space. Such an independent check will hereafter be referred to as an I.C. Cross-plotting the I.C. data resulted in the equal-cost contours shown in Figure III.5-2. The gain trajectory produced by ISAFT is superimposed

```
STATES    CONTROLS    FEEDBACKS    IGAINS
  2          1           2           0

SYSTEM MATRIX A
  0.0000000     1.0000000     -2.0000000     -3.0000000

CONTROL MATRIX B
  0.0000000     1.0000000

TERMINAL TIME T =     1.0000000

TERMINAL COST MATRIX F
  2.0000000     0.0000000     0.0000000     1.0000000

STATE WEIGHTING MATRIX Q
  1.0000000     0.0000000     0.0000000     2.0000000

CONTROL WEIGHTING MATRIX R
  0.1000000
```

FIGURE III.5-1 Computer Print-Out For Example 1

```
ITERATION NUMBER          1

GAIN MATRIX
        0.CCCCOOO         0.0C0C000

  SYSTEM EIGENVALUES
    -C.200CC000D 01    -0.0C000000D 00
    -C.1CCCCCOD 01     -0.0C0C0000D 00

AVERAGE COST =        0.1227681D 01

NECESSARY CONDITIONS VECTOR
   -C.6159624D-01     -0.9331765D-01

  GRACIENT MATRIX
     C.2239452D 00    -0.9323973D-01     -0.9323973D-01     0.1338320D 00

  INVERSE GRADIENT MATRIX
        6.2898689         4.3821046          4.3821046         10.5250361
```

FIGURE III.5-1 (Cont'd)

```
ITERATION NUMBER          2

NEW GAINS
        0.7963600         1.2520928

SYSTEM EIGENVALUES
  -C.34389490 01    -0.0000000D 00
  -C.81314380 C0    -0.0000000D 00

    GAIN TOLERANCE ACHIEVED =        1.0000000

REQUIRED STOPPING TOLERANCE =        0.0100000

AVERAGE COST =         0.1121653D 01

NECESSARY CONDITIONS VECTOR
  -C.2673182D-01    -C.3301758D-01

GRADIENT MATRIX
    C.1144905D C0    -0.5721144D-01    -0.5721144D-01    0.6673589D-01

INVERSE GRADIENT MATRIX
     15.2801933       13.0994265        13.0994265        26.2143374
```

FIGURE III.5-1 (Cont'd)

```
ITERATION NUMBER         3

NEW GAINS
       1.6373387          2.4677983

SYSTEM EIGENVALUES
  -C.46926910D 01     -C.0C0C0C0D 00
  -C.7751C72D CO      -C.0CCC0C0D 00

    GAIN TOLERANCE ACHIEVED =          0.5136254

REQUIRED STOPPING TOLERANCE =          0.0100000

AVERAGE COST =          C.1083638D 01

NECESSARY CONDITIONS VECTOR
  -C.8372973D-C2     -0.9586460D-02

GRADIENT MATRIX
   C.6909313D-01     -0.3735754D-01     -0.3735754D-01     0.3910556D-01

INVERSE GRADIENT MATRIX
     29.9352104        28.5971025         28.5971025         52.8906197
```

FIGURE III.5-1 (Cont'd)

```
ITERATION NLMBER

 NEW CAINS
         2.1621304          3.2142749

 SYSTEM EIGENVALUES
   -C.5450676D C1      -C.0C0C0000 00
   -C.7635989D C0.     -C.CCCC0C0D 00

     GAIN TOLERANCE ACHIEVED =          0.2427197

 RECLIRED STOPPING TCLERANCE =          0.0100000

 AVERAGE COST =        C.1C771720 01

 NECESSARY CONDITIONS VFCTCR
   -C.1332813D-C2      -C.1736675D-02

 GRACIENT MATRIX
    C.5362835D-C1      -0.2976007D-01      -0.2976007D-01      0.2943540D-01

 INVERSE GRADIENT MATRIX
        42.4808458          42.9494143          42.9494143          77.3958539
```

FIGURE III.5-1 (Cont'd)

```
ITERATION NUMBER          5

NEW GAINS
        2.2933385           3.4059298

SYSTEM EIGENVALUES
   -C.56454320 C1      -C.0C000000 00
   -C.76049780 C0      -C.0C000000 00

     GAIN TOLERANCE ACHIEVED =          0.0572127

REQUIRED STOPPING TOLERANCE =          0.0100000

AVERAGE COST =       0.10769100 01

NECESSARY CONDITIONS VECTOR
   -C.47924960-04      -0.86927270-04

GRADIENT MATRIX
    C.50592860-C1      -0.28172380-01      -0.28172380-01      0.27458210-01

INVERSE GRADIENT MATRIX
       46.1089873           47.3082380           47.3082380           84.9576547
```

FIGURE III.5-1 (Cont'd)

```
ITERATION NUMBER          6

NEW GAINS
       2.2996607          3.4155822

 SYSTEM EIGENVALUES
  -0.5655292D 01     -0.0000000D 00
  -0.7602897D 00     -0.0000000D 00

     GAIN TOLERANCE ACHIEVED =          0.0028260

REQUIRED STOPPING TOLERANCE =          0.0100000

 SOLUTION IS COMPLETE. ABOVE GAINS ARE OPTIMAL

 AVERAGE COST =       0.1076909D 01

COMPILE TIME=    28.40 SEC,EXECUTION TIME=   193.64 SEC,OBJECT CODE=   64192 BYTES,
```

FIGURE III.5-1 (Cont'd)

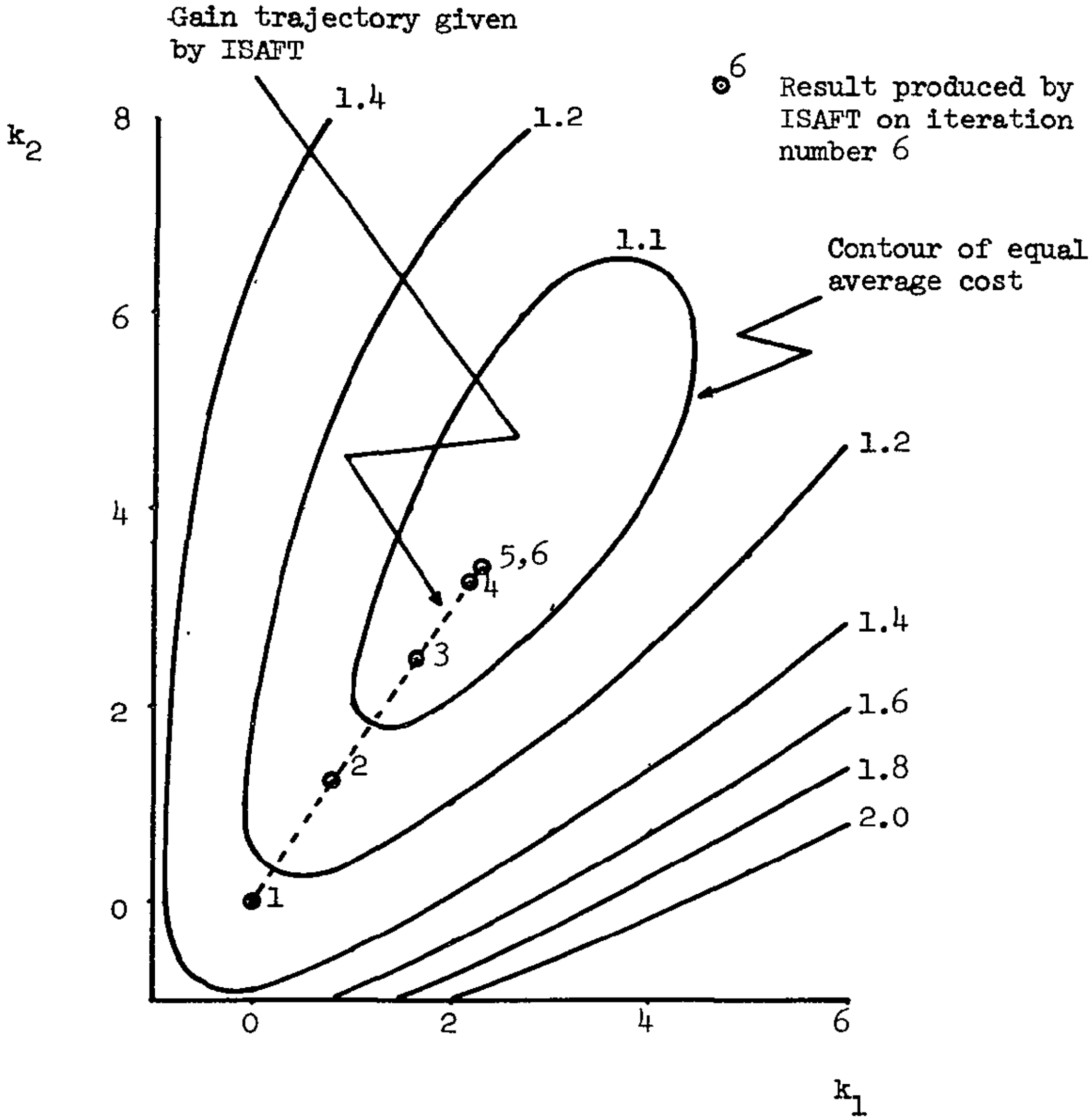


FIGURE III.5-2 Verification of the ISAFT results for Example 1

on Figure III.5-2. It is apparent that ISAFT determined feedback gains giving a minimum average cost.

To provide a comparison for the results produced by ISAFT, the problem defined by (III.5-1) and (III.5-2) was solved for the optimal time varying feedback gains by the normal procedure of Ricatti matrix backwards integration.[2] The resulting feedback gains are compared in Figure III.5-3. with those produced by ISAFT. The average cost for the optimal time varying gains was computed by numerical integration from a number of initial conditions. The average cost for the time-varying gains was 1.030 compared with 1.077 for the time invariant gains.

Example 2

This example used the same system equations and cost functional as Example 1 - see (III.5-1) and (III.5-2) respectively. The feedback structure however, was constrained to feedback of only the first state i.e.,

$$u(t) = - k_1 x_1(t) \tag{III.5-6}$$

A rather inaccurate guess at the optimal feedback gain resulted in the computer print out given in Figure III.5-4. The use of the convergence factor is evident.

Figure III.5-5 compares the results given by ISAFT with those determined by an I.C. It is again clear that ISAFT determined a feedback gain giving a minimum average cost.

The minimum average cost for the single time-invariant feedback was 1.219 compared with 1.077 for allstate time invariant feedback and 1.030 for allstate time varying feedback.

Example 3

This example was chosen to check the effect of complex eigenvalues

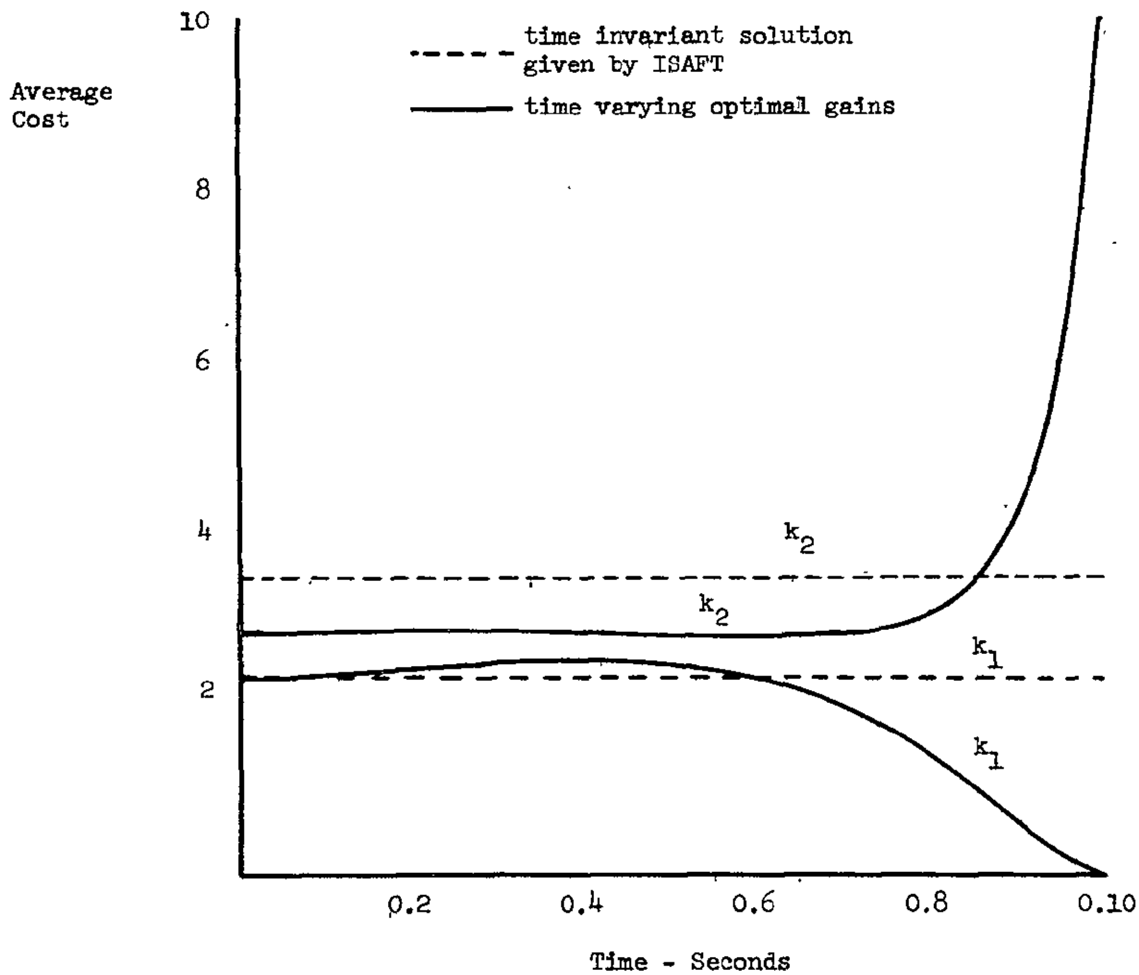


FIGURE III.5-3 Comparison of Time-varying and Time-invariant Optimal Feedback Gains for Example 1

STATES	CONTROLS	FEEDBACKS	IGAINS
2	1	1	1

SYSTEM MATRIX A

0.0000000	1.0000000	-2.0000000	-3.0000000

CONTROL MATRIX B

0.0000000	1.0000000

TERMINAL TIME T = 1.0000000

TERMINAL COST MATRIX F

2.0000000	0.0000000	0.0000000	1.0000000

STATE WEIGHTING MATRIX Q

1.0000000	0.0000000	0.0000000	2.0000000

CONTROL WEIGHTING MATRIX R

0.1000000

```
ITERATION NUMBER          1

GAIN MATRIX
        5.0000000

SYSTEM EIGENVALUES
   -0.15000000D 01     0.21794490D 01
   -0.15000000D 01    -0.21794490D 01

AVERAGE COST =        0.21604570D 01

NECESSARY CONDITIONS VECTOR
     0.27291080D 00

GRADIENT MATRIX
     0.98928970D-02

INVERSE GRADIENT MATRIX
      101.0826292
```

FIGURE III.5-4 (Cont'd)

```
ITERATION NUMBER        2

 NEW GAINS
       -22.5865428

 SYSTEM EIGENVALUES
   -0.62787600D 01     -0.00000000D 00
    0.32787600D 01      0.00000000D 00

 STEP SIZE TOO LARGE. NEW AVERAGE COST WOULD HAVE BEEN    0.37338830 04
 GAIN ADJUSTMENT IS HALVED

 NEW GAINS
       -8.7932714

 SYSTEM EIGENVALUES
   -0.45072030D 01     -0.00000000D 00
    0.15072030D 01      0.00000000D 00

 STEP SIZE TOO LARGE. NEW AVERAGE COST WOULD HAVE BEEN    0.50750500 02
 GAIN ADJUSTMENT IS HALVED

 NEW GAINS
       -1.8966357

 SYSTEM EIGENVALUES
   -0.29651400D 01     -0.00000000D 00
   -0.34859840D-01     -0.00000000D 00

     GAIN TOLERANCE ACHIEVED =          3.6362469

 REQUIRED STOPPING TOLERANCE =          0.0100000

 AVERAGE COST =        0.19246870 01

 NECESSARY CONDITIONS VECTOR
   -0.78732140 00

 GRADIENT MATRIX
    0.58702700 00
```

FIGURE III.5-4 (Cont'd)

INVERSE GRADIENT MATRIX
1.7034993

FIGURE III.5-4 (Cont'd)

```
ITERATION NUMBER          3

NEW GAINS
      -0.5554343

SYSTEM EIGENVALUES
   -0.2397460D 01     -0.0000000D 00
   -0.6025401D 00     -0.0000000D 00

GAIN TOLERANCE ACHIEVED =          2.4146897

REQUIRED STOPPING TOLERANCE =      0.0100000

AVERAGE COST =      0.1300264D 01

NECESSARY CONDITIONS VECTOR
   -0.2070736D 00

GRADIENT MATRIX
    0.3030369D 00

INVERSE GRADIENT MATRIX
          3.2999278
```

FIGURE III.5-4 (Cont'd)

```
ITERATION NUMBER

NEW GAINS
        0.1278936

SYSTEM EIGENVALUES
   -0.1849437D 01     -0.00000000 00
   -0.1150563D 01     -0.00000000 00

      GAIN TOLERANCE ACHIEVED =          5.3429401

REQUIRED STOPPING TOLERANCE =            0.0100000

AVERAGE COST =        0.1221592D 01

NECESSARY CONDITIONS VECTOR
  -0.3396536D-01

GRADIENT MATRIX
   0.2082957D 00

INVERSE GRADIENT MATRIX
        4.8008683
```

FIGURE III.5-4 (Cont'd)

```
ITERATION NUMBER        5

NEW GAINS
     0.2909568

SYSTEM EIGENVALUES
  -0.15000000 01      0.2023779D 00
  -0.15000000 01     -0.2023779D 00

    GAIN TOLERANCE ACHIEVED =          0.5604378

REQUIRED STOPPING TOLERANCE =          0.0100000

AVERAGE COST =        0.1218738D 01

NECESSARY CONDITIONS VECTOR
  -0.1542418D-02

GRADIENT MATRIX
   0.1896087D 00

INVERSE GRADIENT MATRIX
     5.2740210
```

FIGURE III.5-4 (Cont'd)

```
ITERATION NUMBER          6

NEW GAINS
        0.2990916

SYSTEM EIGENVALUES
  -0.15000000D 01     0.22156610D 00
  -0.15000000D 01    -0.22156610D 00

   GAIN TOLERANCE ACHIEVED =            0.0271982

REQUIRED STOPPING TOLERANCE =           0.0100000

AVERAGE COST =      0.12187320D 01

NECESSARY CONDITIONS VECTOR
  -0.36478570D-05

GRADIENT MATRIX
   0.18871240 00

INVERSE GRADIENT MATRIX
        5.2990702
```

FIGURE III.5-4 (Cont'd)

```
ITERATION NUMBER        7

NEW GAINS
      0.2991109

SYSTEM EIGENVALUES
   -0.15000000D 01      0.22160980D 00
   -0.15000000D 01     -0.22160980D 00

    GAIN TOLERANCE ACHIEVED =          0.0000646

REQUIRED STOPPING TOLERANCE =          0.0100000

SOLUTION IS COMPLETE. ABOVE GAINS ARE OPTIMAL

AVERAGE COST =        0.1218732D 01
COMPILE TIME=    27.14 SEC,EXECUTION TIME=    69.23 SEC,OBJECT CODE=   64216 BYTES
```

FIGURE III.5-4 (Cont'd)

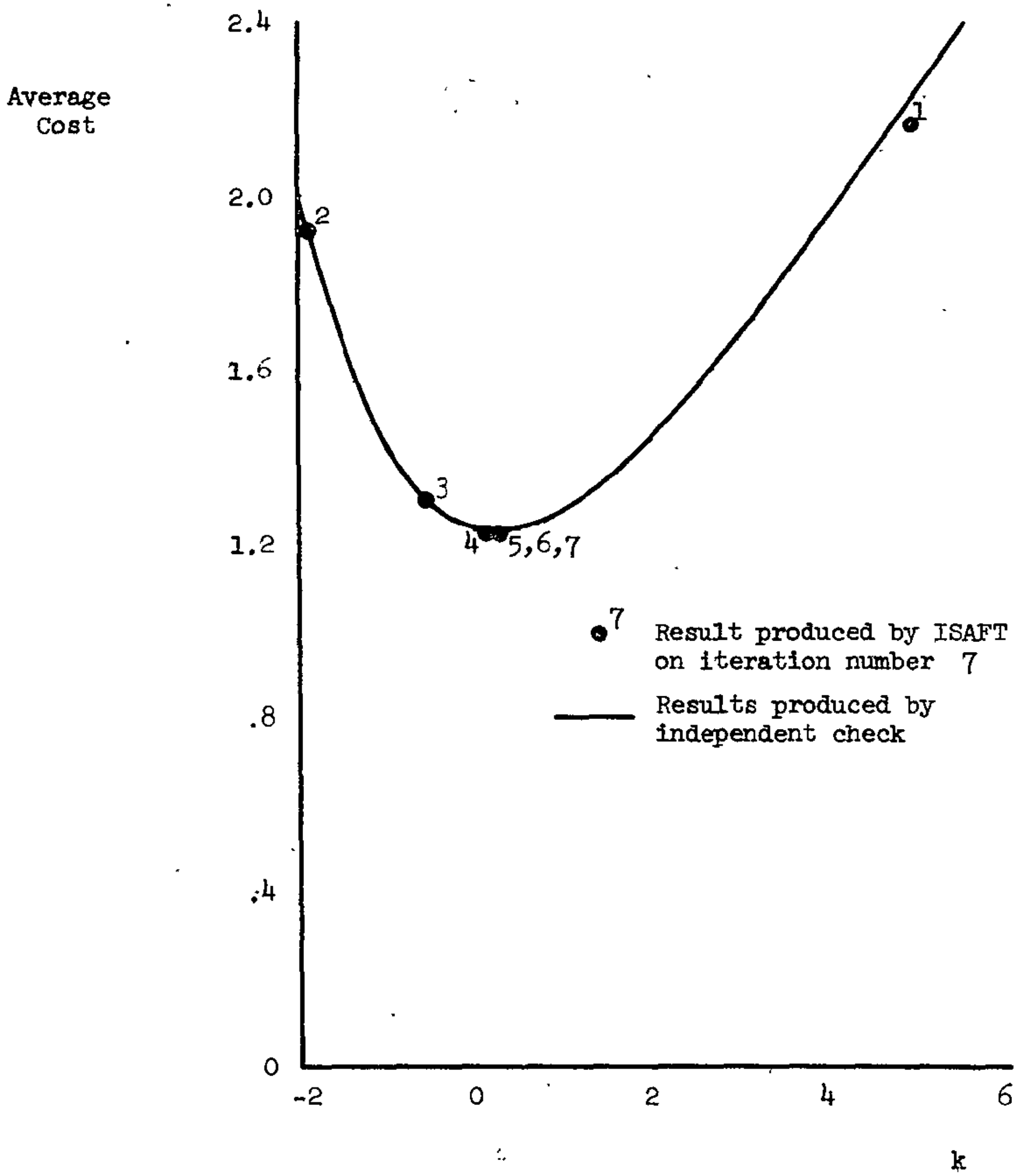


FIGURE III.5-5 Verification of the ISAFT results for Example 2

on the computations. The system equations were

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 6 & 7 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t) \qquad \text{(III.5-7)}$$

with the cost functional

$$J = \underline{x}^T(1) \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \underline{x}(1) + \int_0^1 \underline{x}^T(\tau) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \underline{x}(\tau) + 0.1\, u^2(\tau)\, d\tau \qquad \text{(III.5-8)}$$

It was assumed that only the first state was available for feedback so that

$$u(t) = -\, k_1\, x_1(t) \qquad \text{(III.5-9)}$$

The computer print-out for a starting gain of zero is given in Figure III.5-6 and is compared with an I.C. in Figure III.5-7. There is an offset in Figure III.5-7 between the average costs produced by ISAFT and those produced by the I.C. This is due to the difficulty in approximating a uniform distribution on the surface of a sphere by a finite number of discrete points. (This difficulty increases rapidly with the order of the system). It is nevertheless clear that ISAFT achieved a minimum average cost.

III.6 Comments on the Finite Time Problem

The objective of the I.S.A. approach is to determine the feedback gains giving the minimum value of the gain function GF, which is the expected value of the cost functional. The I.S.A. algorithm presented in Section III.3, however, is designed only to determine feedback gains

```
 STATES     CONTROLS   FEEDBACKS    IGAINS
      3          1           1         0

SYSTEM MATRIX A
        0.0000000          1.0000000          0.0000000          0.0000000
        0.0000000          1.0000000          6.0000000          7.0000000
        0.0000000

CONTROL MATRIX B
        0.0000000          0.0000000          1.0000000

TERMINAL TIME T =                1.0000000

TERMINAL COST MATRIX F
        3.0000000          0.0000000          0.0000000          0.0000000
        2.0000000          0.0000000          0.0000000          0.0000000
        1.0000000

STATE WEIGHTING MATRIX Q
        1.0000000          0.0000000          0.0000000          0.0000000
        2.0000000          0.0000000          0.0000000          0.0000000
        3.0000000

CONTROL WEIGHTING MATRIX R
        0.1000000
```

FIGURE III.6 Computer Print-Out For Example 3

```
ITERATION NUMBER          1

GAIN MATRIX
      0.0000000

SYSTEM EIGENVALUES
   0.30000000D 01     0.00000000D 00
  -0.20000000D 01    -0.00000000D 00
  -0.10000000D 01    -0.00000000D 00

AVERAGE COST =        0.68210960D 03

NECESSARY CONDITIONS VECTOR
  -0.1782183D 03

GRADIENT MATRIX
   0.2884249D 02

INVERSE GRADIENT MATRIX
      0.0346711
```

FIGURE III.5-6 (Cont'd)

```
ITERATION NUMBER        2

NEW GAINS
      6.1790200

SYSTEM EIGENVALUES
   -0.2658447D 01     -0.00000000D 00
    0.2632870D 01      0.00000000D 00
    0.2557668D-01      0.00000000D 00

        GAIN TOLERANCE ACHIEVED =          1.0000000

REQUIRED STOPPING TOLERANCE =          0.0500000

AVERAGE COST =        0.2563643D 03

NECESSARY CONDITIONS VECTOR
   -0.4158308D 02

GRADIENT MATRIX
    0.1611207D 02

INVERSE GRADIENT MATRIX
      0.0620653
```

FIGURE III.5-6 (Cont'd)

```
ITERATION NUMBER          3

NEW GAINS
      8.7598845

SYSTEM EIGENVALUES
  -0.28243870 01     -0.00000000 00
   0.24207210 01      0.00000000 00
   0.40366570 00     -0.00000000 00

  GAIN TOLERANCE ACHIEVED =          0.2946231

REQUIRED STOPPING TOLERANCE =        0.0500000

AVERAGE COST =        0.21743690 03

NECESSARY CONDITIONS VECTOR
   -0.54271280 01

GRADIENT MATRIX
     0.12017290 02

INVERSE GRADIENT MATRIX
        0.0832135
```

FIGURE III.5-6 (Cont'd)

```
ITERATION NUMBER        4

NEW GAINS
        9.2114946

SYSTEM EIGENVALUES
   -0.28507120 01     -0.00000000 00
    0.23767130 01      0.00000000 00
    0.47399860 00      0.00000000 00

    GAIN TOLERANCE ACHIEVED =          0.0490268

REQUIRED STOPPING TOLERANCE =          0.0500000

SOLUTION IS COMPLETE. ABOVE GAINS ARE OPTIMAL

AVERAGE COST =        0.21660510 03
COMPILE TIME=    26.74 SEC,EXECUTION TIME=    976.35 SEC,OBJECT CODE=    64216 BYTES
```


FIGURE III.5-6 (Cont'd)

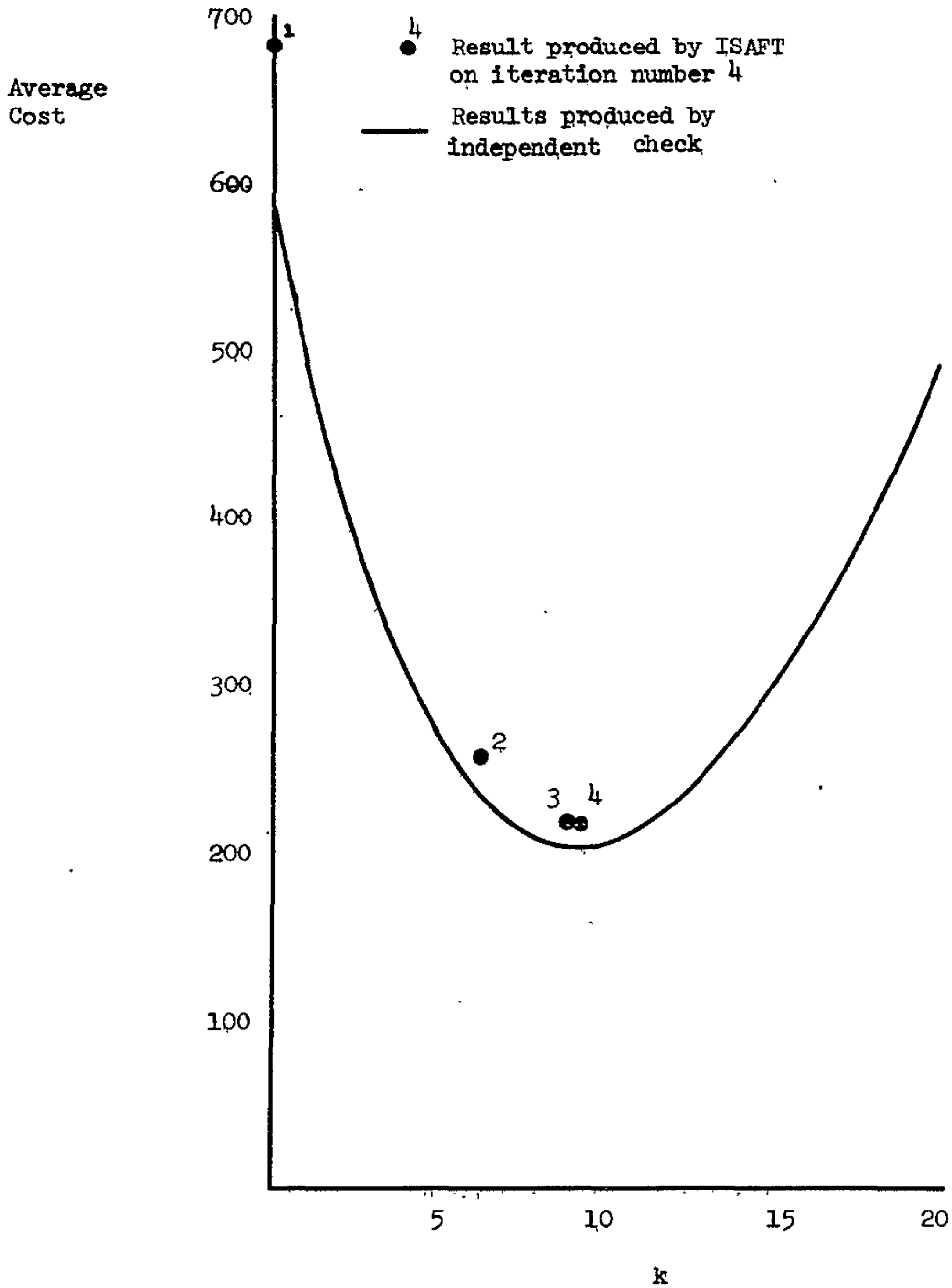


FIGURE III.5-7 Verification of the ISAFT results for Example 3

satisfying certain necessary conditions. The necessary conditions are that the partial derivatives of the gain function with respect to each feedback gain should be zero. These necessary conditions may be satisfied by local maxima, minima or points of inflection. With a multidimensional problem points of inflection seldom occur in practice. The convergence factor μ built into the digital computer program ISAFT prevents convergence to a local maximum. There still remains the problem of convergence to a local rather than a global minimum. No solution to this problem was found. It is felt, however, that it should be feasible to guarantee the existence of a single (global) minimum for suitable system and cost matrices, and useful work could be done in this area.

The digital computer program ISAFT computes the various required functions of the state transition matrix (see (III.3-33)) through (III.3-38)) analytically. It was felt that this best illustrated the procedure. For high order systems however, it should be more economical to evaluate the state transition matrix at a number of discrete time instants and then evaluate the required functions by numerical integration.

CHAPTER IV

SUMMARY AND CONCLUSIONS

This report has considered the determination of optimal time-invariant output-feedback controllers for linear dynamic systems with quadratic cost functionals. The need for such controllers in practical engineering was illustrated, and the contributions of earlier researchers in the field were reviewed.

It was determined that deficiencies existed in two areas. For the case where optimization was to take place over the semi-infinite time interval Levine[19] and Cassidy[20] had each derived a suitable computational algorithm. Both algorithms, however, required initialization by suitably stabilizing feedback gains and neither author gave a method for determination of such gains. For the case of optimization over a finite time interval, no satisfactory existing techniques were uncovered.

The problem of the determination of stabilizing feedback gains was approached via a gradient technique. The technique evolved from an eigenvalue sensitivity relationship given in Fadeev and Fadeeva[27]. It was mechanized in the digital computer program GRADGN (Appendix I) and provides a practical method for determination of local stability maxima in feedback gain space. It is felt that this gradient technique should complement the existing optimization algorithms for the semi-infinite time interval problem.

A new technique was derived for the finite time interval problem. The technique is based on the Initial State Averaging concept, previously used for somewhat different problems by Levine[19] and by Kleinman, Fortmann, and Athans[25]. A computational algorithm was derived and is incorporated in the digital computer program ISAFT (Appendix II). The algorithm satisfies a set of necessary conditions by Newton-Raphson iteration, using their gradient with respect to the feedback gains. This is equivalent to minižization of the expected value of the cost functional by the method of second variations.

The contributions made by this report should aid in the search for practical optimal controllers.

Several outstanding problems remain. Foremost among these is the question of the sufficiency of the solutions obtained by the optimization technique described above. The technique was designed to determine local minima of the expected value of the cost functional. An examination of the convexity of this quantity in feedback gain space might uncover conditions ensuring a single (global) minimum. The gain initialization technique is similarly local and would also benefit by extension to a global technique.

The computational methods used in the digital computer program ISAFT were designed to illustrate the theory. They are satisfactory for low order systems, but not for high order systems. It is felt that relatively simple modifications to ISAFT should eliminate this deficiency.

Bibliography

1. DeRusso, P.M., R.J. Roy, and C.M. Close, "State Variables for Engineers," John Wiley and Sons, New York, 1965.

2. Athans, M. and P.L. Falb, "Optimal Control", McGraw-Hill Book Company, New York, 1966.

3. Schultz, D.G. and J.L. Melsa, "State Functions and Linear Control Systems", McGraw-Hill Book Company, New York, 1967.

4. Sage, A.P., "Optimum Systems Control", Prentice - Hall, Inc., Englewood Cliffs, N.J., 1968.

5. D'Azzo, J.J. and C.H. Houpis, "Feedback Control System Analysis and Synthesis", McGraw-Hill Book Company, New York, 1960.

6. Nyquist, H., "Regeneration Theory", Bell System Tech. J., 1932.

7. Bode, H.W., "Network Analysis and Feedback Amplifier Design", D. Van Nostrand Company, Inc., Princeton, N.J., 1945.

8. Evans, W.R., "Graphical Analysis of Control Systems", Trans. AIEE, Vol. 67, 1948.

9. Wiener, N., "Extrapolation, Interpolation, and Smoothing of Stationary Time Series", John Wiley and Sons, New York, 1949.

10. Newton, G.C. Jr., L.A. Gould and J.F. Kaiser, "Analytical Design of Linear Feedback Controls", John Wiley and Sons, New York, 1957.

11. Pontryagin, L.S., V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko, "The Mathematical Theory of Optimal Processes", Interscience Publishers Inc., New York, 1962.

12. Kalman, R.E., "Contributions to the Theory of Optimal Control", Bol. Soc. Mat. Mex., Vol. 5, pp.102-119, 1960.

13. Kalman, R.E., and R.S. Bucy, "New Results in Linear Filtering and Prediction Theory", J. Basic Eng., ser. D, Vol. 83, pp.95-108, March, 1961.

14. Luenberger, D.G., "Determining the State of a Linear System with Observers of Low Dynamic Order", Ph.D. dissertation, Dept. of Elec. Eng., Stanford Univ., Calif. 1963.

15. Ash. R.H., Jr., "State Estimation in Linear Systems - A Unified Theory of Minimum Order Observers", Ph.D. dissertation, Systems Engineering Division, Rensselaer Polytechnic Institute, Troy, New York, 1969.

16. Hass, V. and S. Martuza, "On the Design of Specific Optimal Controllers", Purdue University, School of Electrical Engineering, May, 1967.

17. Agarwal, G., and R. Sridhar, "Design of Specific Optimal Control Systems", IEEE Region Six Conference, Tucson, Arizona, 1966.

18. Rekasius, Z., "Optimal Linear Regulators with Incomplete State Feedback", IEEE Trans. on Automatic Control, Vol. AC-12 pp. 296-299, June, 1967.

19. Levine, W.S., "Optimal Output-Feedback Controllers for Linear Systems", Ph.D. dissertation, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Mass., 1969.

20. Cassidy, J.F., Jr., "Optimal Control with Unavailable States", Ph.D dissertation, Systems Engineering Division, Rensselaer Polytechnic Institute, Troy, New York, 1969.

21. Koenigsberg, W.D., "Output Feedback Control with Application to Unstable Linear Systems." Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, New York, 1969.

22. Jameson, A., "Design of a Single Input System for Specified Roots Using Output Feedback", IEEE Trans. On Automatic Control, Vol. AC-15, pp. 345-348, June, 1970.

23. Davison, E.J., "On Pole Assignment in Linear Systems with Incomplete State Feedback", IEEE Trans. on Automatic Control, Vol. AC-15, pp. 348-351, June 1970.

24. Kleinman, D.L., and M. Athans, "The Design of Suboptimal Linear Time Varying Systems", IEEE Trans. on Automatic Control, Vol. AC-13, pp. 150-159, April, 1968.

25. Kleinman, D.L., T.Fortmann, and M. Athans, "On the Design of Linear Systems with Piecewise Constant Feedback Gains", IEEE Trans. on Automatic Control, Vol. AC-13, pp. 354-362, August, 1968.

26. McBrinn, D.E. and L. Willner, "Demonstration of Design Techniques", Final Report, Contract Number NAS8-21131, Rensselaer Polytechnic Institute, Troy, New York, November, 1969.

27. Fadeev, D.K. and V.N. Fadeeva, "Computational Methods of Linear Algebra", Freeman, San Francisco, California, 1963, pp.228-229.

28. "IBM System/360 Scientific Subroutine Package (360A-CM-03X), Version III. Application Description", I.B.M., White Plains, N.Y., 1968.

29. Van Ness, J.E., "Inverse Iteration Method for Finding Eigenvectors", IEEE Trans. on Automatic Control, Vol. AC-14, pp. 63-66, February, 1969.

30. Reddy, D.C., "Sensitivity of an Eigenvalue of a Multivariable Control System", Electronics Letters, pp. 446, Vol. 2, No. 12, December, 1966.

31. Levine, W.S., and M. Athans, "On the Determination of the Optimal Constant Output Feedback Gains for Linear Multivariable Systems", IEEE Transactions on Automatic Control, Vol. AC-15, pp. 44-48, February, 1970.

APPENDIX I

This Appendix comprises a listing of the gain initialization digital computer program GRADGN.

```
   /JOB          4257        MCRKINN,LINES=55
 1          DIMENSION W(30,4),XR(30),XI(30),VR(30),VI(30),IROW(30,2),
          1    GRADR(30,30),GRADI(30,30),GRAD(30,30),VRN(30),VIN(30),
          2         G(30,30)
 2          COMMON AMAT(30,30),BMAT(30,30),AHAT(30,30),AAAA(900),ASUR(30,30),
          1    RR(30),RI(30),IANA(30)
 3          REAL K(30,30)
 4    10    FORMAT (7I10)
 5    20    FORMAT (4E13.7)
 6    25    FORMAT (20A4)
 7    30    FORMAT (7F10.4)
 8    35    FORMAT ('1',T4,'STATES',4X,'CONTROLS',4X,'FEEDBACKS',4X,'IGAINS',
          1 4X,'IDELR',4X,'ISTOP')
 9    36    FORMAT (//T3,'STABILITY INCREASE INCREMENT =',F20.8)
10    37    FORMAT (//T3,'MINIMUM STABILITY REQUIRED FOR TERMINATION',3X,F20.8
          1)
11    40    FORMAT (//T3,'SYSTEM MATRIX AMAT')
12    45    FORMAT(//T3,'CONTROL MATRIX BMAT')
13    50    FORMAT (//T2,'MAXIMUM REAL PART OF ROOTS')
14    60    FORMAT (//T3,'EIGVEC ERROR MESSAGES')
15    65    FORMAT (T3,'S.1=',E10.4,10X,'ITER=',I5,10X,'DIF=',E10.4)
16    70    FORMAT (//T3,'EIGENVECTORS CORRES TO MRP EIGENVALUE')
17    75    FORMAT (T6,'ROW REAL PART',5X,'ROW IMAG PART',5X,'COL REAL PART',
          1 5X,'COL IMAG PART')
18    80    FORMAT (//T3,'GRADIENT MATRIX')
19    82    FORMAT ('1',T3,'PROGRAM TERMINATION DUE TO EXCESSIVE NUMBER OF ITE
          1RATIONS')
20    84    FORMAT ('1',T3,'SPECIFIED STABILITY HAS BEEN ATTAINED')
21    86    FORMAT ('1',T3,'GAIN SEARCH PROCEEDS ALONG A NEW GRADIENT. ITERATI
          1ON NUMBER',3X,I3)
22    87    FORMAT (//T3,'GAINS TOO FAR FROM STARTING POINT. COMPUTE NEW GRADI
          1ENT')
23    88    FORMAT (//T3,'STABILITY INCREASE STEP SIZE =',F20.8)
24    90    FORMAT (//T3,'STEP SIZE IS DOUBLED')
25    92    FORMAT (//T3,'CONVEX CURVE FIT TO STABILITY LOCUS')
26    93    FORMAT (//T3,'STEP SIZE TOO LARGE. RETURN TO PREVIOUS BEST GAINS')
27    94    FORMAT (//T3,'CURVE FIT NOT SUCCESSFUL. RETURN TO PREVIOUS BEST GA
          1INS')
28    95    FORMAT (//T3,'STEP SIZE TOO LARGE. COMPUTE NEW GRADIENT AT PREVIOU
          1S BEST GAINS')
29    96    FORMAT ('1',T3,'STABILITY IMPROVEMENT RATE TOO SLOW. PROGRAM TERMI
          1NATED.')
30    97    FORMAT (//T3,'LAST RESULTS SHOW MOST STABLE CONDITION FOUND')
31    98    FORMAT (//T3,'STEP SIZE TOO LARGE. PROGRAM CONTINUES WITH REDUCED
          1STEP SIZE')
      C
      C           NS = NUMBER OF SYSTEM STATES
      C           NC = NUMBER OF CONRTROLS
      C           NF = NUMBER OF FEEDBACK STATES
      C       IGAINS = 0 - INITIAL FEEDBACK GAINS ARE ZERO
      C              = 1 - INITIAL FEEDBACK GAINS ARE READ IN
      C        IDELR = 0 - STABILITY INCREMENT = 0.1
      C              = 1 - STABILITY INCREMENT IS READ IN AS DELR
      C        ISTOP = 0 - PROGRAM MAXIMIZES SYSTEM STABILITY
      C              = 1 - PROGRAM TERMINATES AT STABILITY SPECIFIED BY STOPR
```

```
C
      READ (1,10) NS,NC,NF,IGAINS,IDELR,ISTOP
      WRITE (3,30)
      WRITE (3,10) NS,NC,NF,IGAINS,IDELR,ISTOP
C
C      DELR CONTROLS STABILITY INCREASE INCREMENT
C
      IF (IDELR) 120,120,110
  110 READ (1,30) DELR
      GO TO 130
  120 DELR=0.1
  130 DELRI=DELR
      WRITE (3,36) DELR
C
C      STOPR CONTROLS STABILITY REQUIRED FOR PROGRAM TERMINATION
C
      IF (ISTOP) 150,150,140
  140 READ (1,30) STOPR
      GO TO 160
  150 STOPR=-1.0E6
  160 WRITE (3,37) STOPR
      DO 170 I=1,NS
      DO 170 J=1,NC
  170 K(I,J)=0.
C
C     SYSTEM DATA MATRICES ARE READ IN BY ROWS
C      AMAT IS SYSTEM MATRIX
C      BMAT IS CONTROL MATRIX
C
      READ (1,30) ((AMAT(I,J),J=1,NS),I=1,NS)
      WRITE (3,40)
      WRITE (3,20) ((AMAT(I,J),J=1,NS),I=1,NS)
      READ (1,30) ((BMAT(I,J),J=1,NC),I=1,NS)
      WRITE (3,45)
      WRITE (3,20) ((BMAT(I,J),J=1,NC),I=1,NS)
C
C     INITIALIZATION OF GAIN MATRIX K
C
      IF (IGAINS) 190,190,180
  180 READ (1,30) ((K(I,J),J=1,NC),I=1,NF)
  190 KOUNT=1
      ROOTT=0.
C
C      SUBROUTINE STAB COMPUTES THE SYSTEM STABILITY
C
  195 CALL STAB(NS,NC,NF,K)
  198 IF(RR(1)-STOPR) 999,200,200
  200 KOUNT=KOUNT+1
      IF (KOUNT-10) 210,210,1100
C
C      GAIN SEARCH PROCEEDS ALONG A NEW GRADIENT
C
  210 ROOTR=RR(1)
      ROOTI=RI(1)
      WRITE (3,50)
```

```
      WRITE (3,20) ROOTR
      RTEST=ROOTT
      ROOTT=ROOTR
      K1=KOUNT-1
      WRITE (3,86) K1
C
C     COMPUTATION OF EIGENVECTORS
C
      CALL MSQ(NS)
      CALL EIGVEC(3,AHAT,ASQR,W,IROW,XR,XI,VR,VI,ROOTR,ROOTI,NS,30,0,
     1SW1,ITER,DIF,2)
C
C     SW1 = 0 FOR AN EXACT EIGENVALUE AND NO ROUND-OFF ERROR
C     ITER = NUMBER OF ITERATIONS USED TO FIND EIGENVECTORS.
C     IF TOLERANCE IS NOT ACHIEVED, PROGRAM ACCEPTS VALUES AT ITER = 15.
C     DIF = LARGEST CHANGE IN ANY EIGENVECTOR COMPONENT AT FINAL ITER.
C
      WRITE (3,60)
      WRITE (3,65) SW1,ITER,DIF
      WRITE (3,70)
      WRITE (3,75)
      WRITE (3,20) (VR(I),VI(I),XR(I),XI(I),I=1,NS)
C
C     NORMALISE EIGENVECTORS INNER PRODUCT
C
      VECMGR=0.
      VECMGI=0.
      DO 240 I=1,NS
      VECMGR=VECMGR+VR(I)*XR(I)-VI(I)*XI(I)
  240 VECMGI=VECMGI+VR(I)*XI(I)+VI(I)*XR(I)
      VECMGS=VECMGR*VECMGR+VECMGI*VECMGI
      DO 250 I=1,NS
      VRN(I)=(VR(I)*VECMGR+VI(I)*VECMGI)/VECMGS
  250 VIN(I)=(VI(I)*VECMGR-VR(I)*VECMGI)/VECMGS
C
C     COMPUTE GRADIENT MATRIX
C
      DO 300 J=1,NF
      DO 300 L=1,NC
      GRADR(J,L)=0.
      GRADI(J,L)=0.
      DO 290 I=1,NS
      GRADR(J,L)=GRADR(J,L)-VRN(I)*BMAT(I,L)
  290 GRADI(J,L)=GRADI(J,L)+VIN(I)*BMAT(I,L)
  300 GRAD(J,L)=GRADR(J,L)*XR(J)+GRADI(J,L)*XI(J)
      WRITE (3,80)
      WRITE (3,20) ((GRAD(I,J),J=1,NC),I=1,NF)
      WRITE (3,88) DELR
      GRDSQ=0.
      DO 320 I=1,NF
      DO 320 J=1,NC
  320 GRDSQ=GRDSQ+GRAD(I,J)*GRAD(I,J)
      DELK1=-DELR/GRDSQ
      DO 340 I=1,NF
      DO 340 J=1,NC
```

```
      G(I,J)=K(I,J)
  340 K(I,J)=G(I,J)+DELK1*GRAD(I,J)
      ROOT1=ROOT3
      CALL STAB(NS,NC,NF,K)
      IF (RR(1)-STOPR) 999,390,390
  390 ROOT2=RR(1)
      KOUNT1=1
      KOUNT2=1
      IF (ROOT2-ROOT1) 400,470,600
  400 IF((ROOT1-ROOT2)/DELR-.95) 420,420,500
  420 DELK2=.5*DELR*DELK1/(ROOT2-ROOT1+DELR)
      WRITE (3,92)
      DO 440 I=1,NF
      DO 440 J=1,NC
  440 K(I,J)=G(I,J)+DELK2*GRAD(I,J)
      CALL STAB(NS,NC,NF,K)
      IF(RR(1)-STOPR) 999,999,460
  460 ROOT3=RR(1)
      IF(ROOT3-ROOT2) 470,470,480
  470 IF(KOUNT-2) 475,475,472
  472 IF(RTEST-ROOT3-DELR1) 1200,475,475
  475 DELR=.5*(ROOT1-ROOT3)
      GO TO 200
  480 IF(KOUNT-2) 485,485,482
  482 IF(RTEST-ROOT2-DELR1) 483,485,485
  483 DO 484 I=1,NF
      DO 484 J=1,NC
  484 K(I,J)=G(I,J)+DELK1*GRAD(I,J)
      WRITE (3,93)
      CALL STAB(NS,NC,NF,K)
      GO TO 1200
  485 DELR=.5*(ROOT1-ROOT2)
      WRITE (3,94)
      DO 490 I=1,NF
      DO 490 J=1,NC
  490 K(I,J)=G(I,J)+DELK1*GRAD(I,J)
      GO TO 195
  500 DELK2=2.*DELK1
      KOUNT1=KOUNT1+1
      IF(KOUNT1-11) 505,505,502
  502 DELR=.5*(ROOT1-ROOT2)
      WRITE (3,87)
      GO TO 200
  505 WRITE (3,90)
      DO 510 I=1,NF
      DO 510 J=1,NC
  510 K(I,J)=G(I,J)+DELK2*GRAD(I,J)
      CALL STAB(NS,NC,NF,K)
      IF(RR(1)-STOPR) 999,999,520
  520 ROOT3=RR(1)
      IF (ROOT3-ROOT2) 530,530,580
  530 ROOT2=ROOT3
      DELK1=DELK2
      GO TO 400
  580 IF(KOUNT-2) 585,585,582
```

```
582  IF(RTEST-ROOT2-DELR1) 583,585,685
583  DO 584 I=1,NF
     DO 584 J=1,NC
584  K(I,J)=G(I,J)+DELK1*GRAD(I,J)
     WRITE (3,93)
     CALL STAB(NS,NC,NF,K)
     GO TO 1200
585. DELR=.5*(ROOT1-ROOT2)
     WRITE (3,95)
     DO 590 I=1,NF
     DO 590 J=1,NC
590  K(I,J)=G(I,J)+DELK1*GRAD(I,J)
     GO TO 195
600  KOUNT2=KOUNT2+1
     IF(KOUNT2-6) 605,605,1100
605  DELK2=.5*DELR*DELK1/(ROOT2-ROOT1+DELR)
     WRITE (3,92)
     DO 620 I=1,NF
     DO 620 J=1,NC
620  K(I,J)=G(I,J)+DELK2*GRAD(I,J)
     CALL STAB(NS,NC,NF,K)
     IF(RR(1)-STOPR) 999,999,640
640  ROOT3=RR(1)
     IF(ROOT3-ROOT1) 680,680,660
660  DELK1=DELK2
     WRITE (3,98)
     ROOT2=ROOT3
     GO TO 600
680  IF(KOUNT-2) 685,685,682
682  IF(RTEST-ROOT3-DELR1) 1200,685,685
685. DELR=.5*(ROOT1-ROOT3)
     GO TO 200
999  WRITE (3,84)
     GO TO 2000
1100 WRITE (3,82)
     GO TO 2000
1200 WRITE (3,96)
     WRITE (3,97)
2000 CONTINUE
     STOP
     END
```

```
      SUBROUTINE STAB(NS,NC,NF,K)
      COMMON AMAT(30,30),BMAT(30,30),AHAT(30,30),AAAA(900),ASQR(30,30),
     1       RR(30),RI(30),IANA(30)
      REAL K(30,30)
   10 FORMAT (//T3,'GAIN MATRIX K')
   20 FORMAT (4E18.7)
   30 FORMAT (//T2,'ROOTS',5X,'REAL PART',11X,'IMAG. PART')
   40 FORMAT (2E20.6)
      IA=NS
      WRITE (3,10)
      WRITE (3,20) ((K(I,J),J=1,NC),I=1,NF)
      CALL AMHT(NS,NC,K)
      CALL VECT(NS)
      CALL HSBG(NS,AAAA,IA)
      CALL ATEIG(NS,AAAA,RR,RI,IANA,IA)
      WRITE (3,30)
      WRITE (3,40) (RR(I),RI(I),I=1,NS)
      CALL MAXRT(NS)
      RETURN
      END
```

```
      SUBROUTINE AMHT(NS,NC,K)
C
C     COMPUTES AHAT = AMAT - BMAT*K(TRANSPOSE)
C
      COMMON AMAT(30,30),BMAT(30,30),AHAT(30,30),AAAA(900),ASQR(30,30),
     1       RR(30),RI(30),IANA(30)
      REAL K(30,30)
      DO 100 I=1,NS
      DO 100 J=1,NS
      AHAT(I,J)=AMAT(I,J)
      DO 100 L=1,NC
  100 AHAT(I,J)=AHAT(I,J)-BMAT(I,L)*K(J,L)
      RETURN
      END
```

```
      SUBROUTINE VECT(NS)
C
C     CONVERTS AHAT TO SINGLE SUBSCRIPT FORM AAAA.
C
      COMMON AMAT(30,30),BMAT(30,30),AHAT(30,30),AAAA(900),ASQR(30,30),
     1       RR(30),RI(30),IANA(30)
      DO 100 J=1,NS
      DO 100 I=1,NS
      K=(J-1)*NS+I
  100 AAAA(K)=AHAT(I,J)
      RETURN
      END
```

```
      SUBROUTINE MSQ(NS)
C
C     COMPUTES ASQR = AHAT*AHAT
C
      COMMON AMAT(30,30),BMAT(30,30),AHAT(30,30),AAAA(900),ASQR(30,30),
     1       RR(30),RI(30),IANA(30)
      DO 100 I=1,NS
      DO 100 J=1,NS
      ASQR(I,J)=0.
      DO 100 K=1,NS
  100 ASQR(I,J)=ASQR(I,J)+AHAT(I,K)*AHAT(K,J)
      RETURN
      END
```

```
      SUBROUTINE HSDG(N,A,IA)
C
C     CONVERTS A TO UPPER HESSENBERG FORM
C
      DOUBLE PRECISION DABS,DFLOAT,DSIGN,DBLE,DEXP,DLOG,DLOG10,DATAN
     1,DSIN,DCOS,DSQRT,DTANH,DMOD,DMAX1,DMIN1
      DOUBLE PRECISION S
      DIMENSION A(900)
      L=N
      NIA=L*IA
      LIA=NIA-IA
   20 IF(L-3) 360,40,40
   40 LIA=LIA-IA
      L1=L-1
      L2=L1-1
      ISUB=LIA+L
      IPIV=ISUB-IA
      PIV=ABS(A(IPIV))
      IF(L-3) 90,90,50
   50 M=IPIV-IA
      DO 80 I=L,M,IA
      T=ABS(A(I))
      IF(T-PIV) 80,80,60
   60 IPIV=I
      PIV=T
   80 CONTINUE
   90 IF(PIV) 100,320,100
  100 IF(PIV-ABS(A(ISUB))) 180,180,120
  120 M=IPIV-L
      DO 140 I=1,L
      J=M+I
      T=A(J)
      K=LIA+I
      A(J)=A(K)
  140 A(K)=T
      M=L2-M/IA
      DO 160 I=L1,NIA,IA
      T=A(I)
      J=I-M
      A(I)=A(J)
  160 A(J)=T
  180 DO 200 I=L,LIA,IA
  200 A(I)=A(I)/A(ISUB)
      J=-IA
      DO 240 I=1,L2
      J=J+IA
      LJ=L+J
      DO 220 K=1,L1
      KJ=K+J
      KL=K+LIA
  220 A(KJ)=A(KJ)-A(LJ)*A(KL)
  240 CONTINUE
      K=-IA
      DO 300 I=1,N
      K=K+IA
```

```
      LK=K+L1
      S=A(LK)
      LJ=L-IA
      DO 280 J=1,L2
      JK=K+J
      LJ=LJ+IA
  280 S=S+A(LJ)*A(JK)*1.0D0
  300 A(LK)=S
      DO 310 I=L,LIA,IA
  310 A(I)=0.0
  320 L=L1
      GO TO 20
  360 RETURN
      END
```

```
      SUBROUTINE ATEIG(M,A,RR,RI,IANA,IA)
C
C     COMPUTES ROOTS OF UPPER HESSENBERG MATRIX A
C
      DOUBLE PRECISION DABS,DFLOAT,DSIGN,DBLE,DEXP,DLOG,DLOG10,DATAN
     1,DSIN,DCOS,DSQRT,DTANH,DMOD,DMAX1,DMIN1
      DIMENSION A(900),RR(30),RI(30),PRR(2),PRI(2),IANA(30)
      INTEGER P,P1,Q
      B7=1.0E-8
      E6=1.0E-6
      E10=1.0E-10
      DELTA=0.5
      MAXIT=30
      N=M
   20 N1=N-1
      IN=N1*IA
      NN=IN+N
      IF(N1) 30,1300,30
   30 NP=N+1
      IT=0
      DO 40 I=1,2
      PRR(I)=0.0
   40 PRI(I)=0.0
      PAN=0.0
      PAN1=0.0
      R=0.0
      S=0.0
      N2=N1-1
      IN1=IN-IA
      NN1=IN1+N
      N1N=IN+N1
      N1N1=IN1+N1
   60 T=A(N1N1)-A(NN)
      U=T*T
      V=4.0*A(N1N)*A(NN1)
      IF(ABS(V)-U*E7) 100,100,65
   65 T=U+V
      IF(ABS(T)-AMAX1(U,ABS(V))*E6) 67,67,68
   67 T=0.0
   68 U=(A(N1N1)+A(NN))/2.0
      V=SQRT(ABS(T))/2.0
      IF(T)140,70,70
   70 IF(U) 80,75,75
   75 RR(N1)=U+V
      RR(N)=U-V
      GO TO 130
   80 RR(N1)=U-V
      RR(N)=U+V
      GO TO 130
  100 IF(T)120,110,110
  110 RR(N1)=A(N1N1)
      RR(N)=A(NN)
      GO TO 130
  120 RR(N1)=A(NN)
      RR(N)=A(N1N1)
```

```
  130 RI(N)=0.0
      RI(N1)=0.0
      GO TO 160
  140 RR(N1)=U
      RR(N)=U
      RI(N1)=V
      RI(N)=-V
  160 IF(N2)1280,1280,180
  180 N1N2=N1N1-IA
      RMOD=RR(N1)*RR(N1)+RI(N1)*RI(N1)
      EPS=E10*SQRT(RMOD)
      IF(ABS(A(N1N2))-EPS)1280,1280,240
  240 IF(ABS(A(NN1))-E10*ABS(A(NN))) 1300,1300,250
  250 IF(ABS(PAN1-A(N1N2))-ABS(A(N1N2))*E6) 1240,1240,260
  260 IF(ABS(PAN-A(NN1))-ABS(A(NN1))*E6)1240,1240,300
  300 IF(IT-MAXIT) 320,1240,1240
  320 J=1
      DO 360 I=1,2
      K=NP-I
      IF(ABS(RR(K)-PRR(I))+ABS(RI(K)-PRI(I))-DELTA*(ABS(RR(K))
     1    +ABS(RI(K)))) 340,360,360
  340 J=J+I
  360 CONTINUE
      GO TO (440,460,460,480),J
  440 R=0.0
      S=0.0
      GO TO 500
  460 J=N+2-J
      R=RR(J)*RR(J)
      S=RR(J)+RR(J)
      GO TO 500
  480 R=RR(N)*RR(N1)-RI(N)*RI(N1)
      S=RR(N)+RR(N1)
  500 PAN=A(NN1)
      PAN1=A(N1N2)
      DO 520 I=1,2
      K=NP-I
      PRR(I)=RR(K)
  520 PRI(I)=RI(K)
      P=N2
      IF(N-3)600,600,525
  525 IPI=N1N2
      DO 580 J=2,N2
      IPI=IPI-IA-1
      IF(ABS(A(IPI))-EPS) 600,600,530
  530 IPIP=IPI+IA
      IPIP2=IPIP+IA
      D=A(IPIP)*(A(IPIP)-S)+A(IPIP2)*A(IPIP+1)+R
      IF(D)540,560,540
  540 IF(ABS(A(IPI)*A(IPIP+1))*(ABS(A(IPIP)+A(IPIP2+1)-S)+ABS(A(IPIP2+2)
     1 )) -ABS(D)*EPS) 620,620,560
  560 P=N1-J
  580 CONTINUE
  600 Q=P
      GO TO 680
```

```
  620 P1=P-1
      Q=P1
      IF(P1-1)680,680,650
  650 DO 660 I=2,P1
      IPI=IPI-IA-1
      IF(ABS(A(IPI))-EPS)680,680,660
  660 Q=Q-1
  680 II=(P-1)*IA+P
      DO 1220 I=P,N1
      II1=II-IA
      IIP=II+IA
      IF(I-P)720,700,720
  700 IPI=II+1
      IPIP=IIP+1
      G1=A(II)*(A(II)-S)+A(IIP)*A(IPI)+R
      G2=A(IPI)*(A(IPIP)+A(II)-S)
      G3=A(IPI)*A(IPIP+1)
      A(IPI+1)=0.0
      GO TO 780
  720 G1=A(II1)
      G2=A(II1+1)
      IF(I-N2)740,740,760
  740 G3=A(II1+2)
      GO TO 780
  760 G3=0.0
  780 CAP=SQRT(G1*G1+G2*G2+G3*G3)
      IF(CAP)800,860,800
  800 IF(G1)820,840,840
  820 CAP=-CAP
  840 T=G1+CAP
      PSI1=G2/T
      PSI2=G3/T
      ALPHA=2.0/(1.0+PSI1*PSI1+PSI2*PSI2)
      GO TO 880
  860 ALPHA=2.0
      PSI1=0.0
      PSI2=0.0
  880 IF(I-Q)900,960,900
  900 IF(I-P)920,940,920
  920 A(II1)=-CAP
      GO TO 960
  940 A(II1)=-A(II1)
  960 IJ=II
      DO 1040 J=I,N
      T=PSI1*A(IJ+1)
      IF(I-N1)980,1000,1000
  980 IP2J=IJ+2
      T=T+PSI2*A(IP2J)
 1000 ETA=ALPHA*(T+A(IJ))
      A(IJ)=A(IJ)-ETA
      A(IJ+1)=A(IJ+1)-PSI1*ETA
      IF(I-N1)1020,1040,1040
 1020 A(IP2J)=A(IP2J)-PSI2*ETA
 1040 IJ=IJ+IA
      IF(I-N1)1080,1060,1060
```

```
 1060 K=N
      GO TO 1100
 1080 K=I+2
 1100 IP=IIP-I
      DO 1180 J=0,K
      JIP=IP+J
      JI=JIP-IA
      T=PSI1*A(JIP)
      IF(I-N1)1120,1140,1140
 1120 JIP2=JIP+IA
      T=T+PSI2*A(JIP2)
 1140 ETA=ALPHA*(T+A(JI))
      A(JI)=A(JI)-ETA
      A(JIP)=A(JIP)-ETA*PSI1
      IF(I-N1)1160,1180,1180
 1160 A(JIP2)=A(JIP2)-ETA*PSI2
 1180 CONTINUE
      IF(I-N2)1200,1220,1220
 1200 JI=II+3
      JIP=JI+IA
      JIP2=JIP+IA
      ETA=ALPHA*PSI2*A(JIP2)
      A(JI)=-ETA
      A(JIP)=-ETA*PSI1
      A(JIP2)=A(JIP2)-ETA*PSI2
 1220 II=IIP+1
      IT=IT+1
      GO TO 60
 1240 IF(ABS(A(NN1))-ABS(A(N1N2))) 1300,1280,1280
 1280 IANA(N)=3
      IANA(N1)=2
      N=N2
      IF(N2)1400,1400,20
 1300 RR(N)=A(NN)
      RI(N)=0.0
      IANA(N)=1
      IF(N1)1400,1400,1320
 1320 N=N1
      GO TO 20
 1400 RETURN
      END
```

```
      SUBROUTINE MAXRT(NS)
C
C     COMPUTES ROOT HAVING MAXIMUM REAL PART
C
      COMMON AMAT(30,30),BMAT(30,30),AHAT(30,30),AAAA(900),ASQR(30,30),
     1        RR(30),RI(30),IANA(30)
      DO 100 I=2,NS
      IF (RR(1)-RR(I)) 50,100,100
   50 RI(1)=RI(I)
      RR(1)=RR(I)
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE EIGVEC(IVC,  A, B, W, IROW, XR, XI, VR, VI, ROOTRE,    ESY1
     1  ROOTIE, NE, NMAX, T2, SW1, COUNTE, ERR,MMM)                     ESY1
C        SUBROUTINE TO FIND THE EIGENVECTORS OF A NON-SYMMETRIC MATRIX  ESY1
C        BY A MODIFIED WILKINSON'S INVERSE ITERATION METHOD.            ESY1
C        CONTROL IVC CODE IS                                            ESY1
C               1  FIND ONLY THE REGULAR EIGENVECTORS   (A X = LAMBDA X) ESY1
C               2  FIND ONLY THE TRANSPOSED EIGENVECTORS (AT V = LAMBDA V)ESY1
C               3  FIND BOTH TYPES OF EIGENVECTORS.                     ESY1
      DIMENSION A(30,30),B(30,30),W(30,4),XR(30),XI(30),VR(30),VI(30),
     1IROW(30,2)
      INTEGER COUNT, COUNTE, T2                                         ESY1
      I01=1
      I03=3
      ROOTR = ROOTRE                                                    ESY1
      ROOTI = ROOTIE                                                    ESY1
      N = NE                                                            ESY1
      MM = MMM - 1                                                      ESY1
      N1 = N - 1                                                        ESY1
      NP1 = N + 1                                                       ESY1
      IVC1 = IVC - 1                                                    ESY1
      IVC2 = IVC1 - 1                                                   ESY1
      COUNT = 1                                                         ESY1
      DO 400 I=1,N
      W(I,1)=0.0E0
      XR(I)=0.0E0
  400 CONTINUE
      CLIM = 1.0E-4                                                     ESY1
      IF(ROOTI) 1, 60, 1                                                ESY1
C                                                                       ESY1
C        COMPLEX EIGENVALUE.                                            ESY1
C                                                                       ESY1
    1 TEMP = - ROOTR - ROOTR                                            ESY1
      ISW = 2                                                           ESY1
      TEMP2=ROOTR*ROOTR+ROOTI*ROOTI
      JJ = 300                                                          ESY1
      DO 606 I = 1, N                                                   ESY1
      IF(T2) 600, 603, 600                                              ESY1
  600 DO 602 J = 1, N                                                   ESY1
      JJ = JJ + 1                                                       ESY1
      IF(JJ - 251) 602, 601, 601                                        ESY1
  601 JJ = 1                                                            ESY1
      READ (T2) (W(LL,1), LL = 1,250)                                   ESY1
  602 B(I,J) = A(I,J)*TEMP + W(JJ,1)                                    ESY1
      GO TO 605                                                         ESY1
  603 DO 604 J = 1, N                                                   ESY1
  604 B(I,J) = A(I,J)*TEMP + B(I,J)                                     ESY1
  605 B(I,I) = B(I,I) + TEMP2                                           ESY1
  606 A(I,I) = A(I,I) - ROOTR                                           ESY1
      IF(T2 .NE. 0) REWIND T2                                           ESY1
      GO TO 700                                                         ESY1
  607 IF(ICC) 622, 608, 622                                             ESY1
C                                                                       ESY1
C        MATRIX SINGULAR.                                               ESY1
C                                                                       ESY1
  622 IF(IVC2) 623, 625, 623                                            ESY1
```

```
      623 DO 624 LL = 1, N
          W(LL,2)=0.0
      624 XI(LL)=0.0
          IF(IVC1) 625, 514, 625
      625 DO 626 LL = 1, N
          W(LL,4)=0.0
      626 VI(LL)=0.0
          GO TO 511
C
C         MATRIX NOT SINGULAR.
C
      608 DO 609 LL = 1, N
          W(LL,1)=1.0
          W(LL,2)=1.0
          W(LL,3)=1.0
      609 W(LL,4)=1.0
      699 IF(IVC2) 610, 612, 610
      610 DO 611 I = 1, N
          I2 = IROW(I,2)
          XI(I2) = W(I,1)*ROOTI
          DO 611 J = 1, N
      611 XI(I2) = XI(I2) + A(I,J)*W(J,2)
          IF(IVC1) 612, 500, 612
      612 DO 613 I = 1, N
          VI(I) = W(I,3)*ROOTI
          DO 613 J = 1,N
      613 VI(I) = VI(I) + A(J,I)*W(J,4)
          GO TO 499
      615 CERR = 0.0
          IF(IVC2) 616, 619, 616
      616 DO 618 I = 1, N
          XR(I) = -W(I,2)
          DO 617 J = 1, N
      617 XR(I) = XR(I) + A(I,J)*XI(J)
      618 XR(I) = XR(I)/ROOTI
          IF(IVC1) 619, 633, 619
      619 DO 621 I = 1, N
          VR(I) = -W(I,4)
          DO 620 J = 1, N
      620 VR(I) = VR(I) + A(J,I)*VI(J)
      621 VR(I) = VR(I)/ROOTI
C
C         SEARCH VECTORS FOR LARGEST ELEMENT AND NORMALIZE.
C
      627 AMAX = 0.0
          DO 629 L = 1, N
          TEMP = VR(L)**2 + VI(L)**2
          IF(TEMP - AMAX) 629, 629, 628
      628 AMAX = TEMP
          I2 = L
      629 CONTINUE
          C1 = VR(I2)/AMAX
          C2 = -VI(I2)/AMAX
          DO 630 L = 1, N
          TEMP = VI(L)
```

```
      VI(L) = VR(L)*C2 + TEMP*C1                                      ESY
  630 VR(L) = VR(L)*C1 - TEMP*C2                                      ESY
      IF(COUNT .EQ. 1) GO TO 632                                      ESY
      DO 631 LL = 1, N                                                ESY
  631 CERR = AMAX1(CERR, ABS(VR(LL) - W(LL,3)), ABS(VI(LL) - W(LL,4)))ESY
  632 IF(IVC2) 633, 638, 633                                          ESY
  633 AMAX = 0.0                                                      ESY
      DO 635 L = 1, N                                                 ESY
      TEMP = XR(L)**2 + XI(L)**2                                      ESY
      IF(TEMP - AMAX) 635, 635, 634                                   ESY
  634 AMAX = TEMP                                                     ESY
      I2 = L                                                          ESY
  635 CONTINUE                                                        ESY
      C1 = XR(I2)/AMAX                                                ESY
      C2 = -XI(I2)/AMAX                                               ESY
      DO 636 L = 1, N                                                 ESY
      TEMP = XI(L)                                                    ESY
      XI(L) = XR(L)*C2 + TEMP*C1                                      ESY
  636 XR(L) = XR(L)*C1 - TEMP*C2                                      ESY
      IF(COUNT .EQ. 1) GO TO 646                                      ESY
      DO 637 LL = 1, N                                                ESY
  637 CERR = AMAX1(CERR, ABS(XR(LL) - W(LL,1)), ABS(XI(LL) - W(LL,2)))ESY
C                                                                     ESY
C          TEST FOR CONVERGENCE.                                      ESY
C                                                                     ESY
  638 IF(COUNT .EQ. 1) GO TO 646                                      ESY
      IF(CERR .GE. 1.0E-4) GO TO 639                                  ESY
      IF(CERR .GE. CLIM) GO TO 648                                    ESY
      CLIM = CERR                                                     ESY
      IF(CLIM .LE. 1.0E-8) GO TO 648                                  ESY
  639 IF(COUNT .GE. 15) GO TO 68                                      ESY
  647 COUNT = COUNT + 1                                               ESY
      IF(ROOTI) 642, 673, 642                                         ESY
  642 IF(IVC2) 640, 644, 640                                          ESY
  640 DO 641 LL = 1, N                                                ESY
      W(LL,1) = XR(LL)                                                ESY
  641 W(LL,2) = XI(LL)                                                ESY
      IF(IVC1) 644, 610, 644                                          ESY
  644 DO 645 LL = 1, N                                                ESY
      W(LL,3) = VR(LL)                                                ESY
  645 W(LL,4) = VI(LL)                                                ESY
      GO TO 699                                                       ESY
  646 CERR = 0.0                                                      ESY
      IF(ICC) 648, 647, 648                                           ESY
  648 ERR = CERR                                                      ESY
      COUNTE = COUNT                                                  ESY
      IF(ROOTI) 667, 668, 667                                         ESY
  667 DO 649 I = 1, N                                                 ESY
  649 A(I,I) = A(I,I) + ROOTR                                         ESY
      RETURN                                                          ESY
   68 PRINT 101, ROOTR, ROOTI, CERR                                   ESY
      GO TO 648                                                       ESY
C                                                                     ESY
C          REAL EIGENVECTORS.                                         ESY
C                                                                     ESY
```

83

```
   60 ISW = 1
      DO 651 I = 1, N
      DO 650 J = 1, N
  650 B(I,J) = A(I,J)
  651 B(I,I) = B(I,I) - ROOTR
      GO TO 700
  652 IF(ICC) 680, 685, 680
C
C        SINGULAR MATRIX.
C
  680 IF(IVC2) 681, 683, 681
  681 DO 682 L = 1, N
  682 XI(L) = 0.0
      IF(IVC1) 683, 514, 683
  683 DO 684 L = 1, N
  684 VI(L) = 0.0
      GO TO 511
C
C        MATRIX NOT SINGULAR.
C
  685 IF(IVC2) 653, 656, 653
  653 DO 654 L = 1, N
  654 XI(L) = 1.0
      IF(IVC1) 656, 500, 656
  656 DO 657 L = 1, N
  657 VI(L) = 1.0
      GO TO 499
C
C        NORMALIZE REAL VECTORS.
C
  655 CERR = 0.0
      IF(IVC2) 658, 662, 658
  658 C1=0.0
      C2=0.0
      DO 660 L = 1, N
      TEMP = ABS(XI(L))
      IF(TEMP - C1) 660, 660, 659
  659 C1 = TEMP
      C2 = XI(L)
  660 CONTINUE
      DO 661 L = 1, N
      XI(L) = XI(L)/C2
      CERR = AMAX1(CERR, ABS(XI(L) - XR(L)))
  661 XR(L) = XI(L)
      IF(IVC1) 662, 638, 662
  662 C2=0.0
      C1=0.0
      DO 664 L = 1, N
      TEMP = ABS(VI(L))
      IF(TEMP - C1) 664, 664, 663
  663 C1 = TEMP
      C2 = VI(L)
  664 CONTINUE
      DO 665 LL = 1, N
      VI(LL) = VI(LL)/C2
```

```
      CERR = AMAX1(CERR, ABS(VI(LL) - W(LL,1)))                         ESY
      W(LL,1)=VI(LL)
  665 VR(LL)=W(LL,1)
      GO TO 638                                                         ESY
  668 IF(IVC2) 669, 671, 669                                            ESY
  669 DO 670 L = 1, N                                                   ESY
  670 XI(L) = 0.0                                                       ESY
      IF(IVC1) 671, 70, 671                                             ESY
  671 DO 672 L = 1, N                                                   ESY
  672 VI(L) = 0.0                                                       ESY
   70 RETURN                                                            ESY
  673 IF(IVC2) 674, 502, 674                                            ESY
  674 DO 675 I = 1, N                                                   ESY
      I2 = IROW(I,2)                                                    ESY
  675 XI(I2) = XR(I)                                                    ESY
      GO TO 500                                                         ESY
C                                                                       ESY
C        BACK SUBSTITUTION SECTION.                                     ESY
C                                                                       ESY
  499 IF(IVC2) 500, 502, 500                                            ESY
  500 DO 501 I = 2, N                                                   ESY
      I1 = I - 1                                                        ESY
      DO 501 J = 1, I1                                                  ESY
  501 XI(I) = XI(I) - B(I,J)*XI(J)                                      ESY
  511 IF(IVC1) 502, 514, 502                                            ESY
  502 DO 510 I = 1, N                                                   ESY
      I1 = I - 1                                                        ESY
      IF(I1) 503, 505, 503                                              ESY
  503 DO 504 J = 1, I1                                                  ESY
  504 VI(I) = VI(I) - B(J,I)*VI(J)                                      ESY
      IF(ICC) 505, 506, 505                                             ESY
  505 IF(B(I,I)) 506, 507, 506                                          ESY
  506 VI(I) = VI(I)/B(I,I)                                              ESY
      GO TO 510                                                         ESY
  507 IF(VI(I)) 508, 509, 508                                           ESY
  508 VI(I) = VI(I)*1.0E+15                                             ESY
      GO TO 510                                                         ESY
  509 VI(I) = 1.0                                                       ESY
  510 CONTINUE                                                          ESY
      IF(IVC2) 514, 525, 514                                            ESY
  514 DO 522 I = 1, N                                                   ESY
      IR = NP1 - I                                                      ESY
      IF(I - 1) 515, 517, 515                                           ESY
  515 I2 = IR + 1                                                       ESY
      DO 516 J = I2, N                                                  ESY
  516 XI(IR) = XI(IR) - B(IR,J)*XI(J)                                   ESY
      IF(ICC) 517, 518, 517                                             ESY
  517 IF(B(IR,IR)) 518, 519, 518                                        ESY
  518 XI(IR) = XI(IR)/B(IR,IR)                                          ESY
      GO TO 522                                                         ESY
  519 IF(XI(IR)) 520, 521, 520                                          ESY
  520 XI(IR) = XI(IR)*1.0E+15                                           ESY
      GO TO 522                                                         ESY
  521 XI(IR) = 1.0                                                      ESY
  522 CONTINUE                                                          ESY
```

```
      IF(IVC1) 525, 529, 525
  525 DO 526 I = 2, N
      IR = NP1 - I
      I2 = IR + 1
      DO 526 J = I2, N
  526 VI(IR) = VI(IR) - B(J,IR)*VI(J)
      DO 527 L = 1, N
      I2 = IROW(L,1)
  527 VR(I2) = VI(L)
      DO 528 L = 1, N
  528 VI(L) = VR(L)
  529 IF(ROOTI) 615, 655, 615
C
C          FACTOR MATRIX.
C
  700 ICC = 0
      SW1=1.0E72
      DO 701 LL = 1, N
  701 IROW(LL,1) = LL
      DO 708 K = 1, N1
      AMAX = ABS(B(K,K))
      IMAX = K
      K1 = K + 1
      DO 702 I = K1, N
      IF(AMAX .GT. ABS(B(I,K))) GO TO 702
      AMAX = ABS(B(I,K))
      IMAX = I
  702 CONTINUE
      IF(AMAX .LT. SW1) SW1 = AMAX
      IF(AMAX .GE. 1.0E-25) GO TO 723
      B(K,K) = 0.0
      ICC = ICC + 1
      GO TO 708
  723 IF(IMAX .EQ. K) GO TO 704
      DO 703 J = 1, N
      AMAX = B(K,J)
      B(K,J) = B(IMAX,J)
  703 B(IMAX,J) = AMAX
      I2 = IROW(K,1)
      IROW(K,1) = IROW(IMAX,1)
      IROW(IMAX,1) = I2
  704 DO 707 I = K1, N
      IF(B(I,K)) 705, 707, 705
  705 B(I,K) = B(I,K)/B(K,K)
      DO 706 J = K1, N
  706 B(I,J) = B(I,J) - B(K,J)*B(I,K)
  707 CONTINUE
  708 CONTINUE
      AMAX = ABS(B(N,N))
      IF(AMAX - 1.0E-25) 712, 712, 713
  712 B(N,N)=0.0
      SW1=0.0
      ICC = ICC + 1
      GO TO 709
  713 IF(AMAX .LT. SW1) SW1 = AMAX
```

```
      709 IF(ICC .LE. ILW) GO TO 710                                      ESY
          IF(MM) 1051,1050,1051
     1051 WRITE(103,1021) ICC
          COUNTE = 0                                                      ESY
          RETURN                                                          ESY
     1050 WRITE(103,1052) ICC
      710 DO 711 LL = 1, N                                                ESY
          I2 = IROW (LL,1)                                                ESY
      711 IROW(I2,2) = LL                                                 ESY
          IF(ROOTI) 607, 652, 607                                         ESY
     1052 FORMAT(///23H ****** WARNING ******     ,'   SUBROUTINE EIGVEC HAS ESY
         1FOUND AN EIGENVALUE OF APPARENT MULTIPLICITY',                  ESY
         1                                     I4,/23X,'   COMPUTATION OF EIESY
         2GENVECTOR(S) CONTINUES AT USER S OPTION'//)                     ESY
      101 FORMAT(38H)MORE THAN 15 LOOPS FOR EIGENVECTOR OF,2E12.4,        ESY
         2   14H DIFFERENCE OF,E12.4)                                     ESY
      102 FORMAT(16H0****WARNING**** , I4, 71H ZERCS ON DIAGONAL OF FACTOREDESY
         1 MATRIX.  CHECK FOR MULTIPLE EIGENVALUES./20X,                  ESY
         2' SUBROUTINE EIGVEC WILL NOT PERFORM COMPUTATION FOR THIS EIGENVECESY
         3TOR '//)                                                        ESY
          END
```

```
/DATA
```

APPENDIX II

This Appendix comprises a listing of the I.S.A. optimization digital computer program ISAFT.

```
/JOB          4257       NOPRINK,LINES=55
C
C        PROGRAM ISAFT
C        DETERMINES OPTIMAL CONSTANT GAIN OUTPUT FEEDBACK CONTROLLERS
C        FOR FINITE TIME STATE REGULATOR PROBLEMS
C        OUTPUTS ARE ASSUMED TO BE FIRST NF STATES
C
      COMMON M(3,3),MI(3,3),RC(3),AHAT(3,3),T,K(3,3),G(3,3),GHAT(3,3),
     1       FT(3,3),FFTI(3,3),DFT(3,3,3,3),FDFTI(3,3,3,3),
     2       D2FT(3,3,3,3,3,3),DFDFTI(3,3,3,3,3,3),FD2FTI(3,3,3,3,3,3),
     3       VW(9),VGRAD(9,9),GRADI(9,9),B(3,3),F(3,3),R(3,3),NS,NC,NF
      DIMENSION A(3,3),C(3,3)
      COMPLEX*16 M,MI,RC
      DOUBLE PRECISION AHAT,T,K,G,GHAT,FT,FFTI,DFT,FDFTI,D2FT,DFDFTI,
     1                 FD2FTI,VW,VGRAD,GRADI
      DOUBLE PRECISION GF,GF2,TEST,TEST1,GNSTOP
      DOUBLE PRECISION DMAX1,DABS
   10 FORMAT (4F18.7)
   11 FORMAT (4D18.7)
   15 FORMAT (7I10)
   20 FORMAT (14I5)
   25 FORMAT ('1',T4,'STATES',4X,'CONTROLS',4X,'FEEDBACKS',4X,'IGAINS')
   30 FORMAT (//T3,'INVERSE GRADIENT MATRIX')
   35 FORMAT (//T3,'SYSTEM MATRIX A')
   40 FORMAT (//T3,'NEW GAINS')
   45 FORMAT (//T3,'CONTROL MATRIX B')
   50 FORMAT (//T3,'GAIN MATRIX')
   55 FORMAT (//T3,'TERMINAL TIME T =',F18.7)
   60 FORMAT (//T7,'GAIN TOLERANCE ACHIEVED =',F18.7)
   62 FORMAT (//T3,'AVERAGE COST =',D2 .7)
   65 FORMAT (//T3,'TERMINAL COST MATRIX F')
   70 FORMAT (//T3,'REQUIRED STOPPING TOLERANCE =',F18.7)
   75 FORMAT (//T3,'STATE WEIGHTING MATRIX Q')
   80 FORMAT (//T3,'CONTROL WEIGHTING MATRIX R')
   85 FORMAT ('1','ITERATION NUMBER',I1.)
   90 FORMAT (//T3,'SOLUTION IS COMPLETE. ABOVE GAINS ARE OPTIMAL')
   92 FORMAT (//T3,'STEP SIZE TOO LARGE. NEW AVERAGE COST WOULD HAVE BEE
     1N',D16.7)
   93 FORMAT (T3,'GAIN ADJUSTMENT IS HALVED')
   95 FORMAT (//T3,'CONVERGENCE TOO SLOW. PROGRAM TERMINATED')
  100 READ (1,20) NS,NC,NF,IGAINS
      READ (1,10) ((A(I,J),J=1,NS),I=1,NS)
      READ (1,10) ((B(I,J),J=1,NC),I=1,NS)
      READ (1,11) T
      READ (1,10) ((F(I,J),J=1,NS),I=1,NS)
      READ (1,10) ((Q(I,J),J=1,NS),I=1,NS)
      READ (1,10) ((R(I,J),J=1,NC),I=1,NC)
      READ (1,11) GNSTOP
      WRITE (3,25)
      WRITE (3,15) NS,NC,NF,IGAINS
      WRITE (3,35)
      WRITE (3,10) ((A(I,J),J=1,NS),I=1,NS)
      WRITE (3,45)
      WRITE (3,10) ((B(I,J),J=1,NC),I=1,NS)
      WRITE (3,55) T
```

```
      WRITE (3,65)
      WRITE (3,10) ((F(I,J),J=1,NS),I=1,NS)
      WRITE (3,75)
      WRITE (3,10) ((Q(I,J),J=1,NS),I=1,NS)
      WRITE (3,80)
      WRITE (3,10) ((R(I,J),J=1,NC),I=1,NC)
      NFP1=NF+1
      NFC=NF*NC
      DO 1000 I=1,NS
      DO 1000 J=1,NC
      G(I,J)=0.000
 1000 K(I,J)=0.000
      IF(IGAINS) 1200,1200,1100
 1100 READ (1,11) ((K(I,J),J=1,NC),I=1,NF)
 1200 CONTINUE
      DO 1220 I=1,NS
      DO 1220 J=NF,NS
      QHAT(I,J)=Q(I,J)
 1220 AHAT(I,J)=A(I,J)
      IT=1
      WRITE (3,85) IT
      WRITE (3,50)
      WRITE (3,10) ((K(I,J),J=1,NC),I=1,NF)
 1230 CONTINUE
      DO 1250 I=1,NS
      DO 1250 J=1,NF
      AHAT(I,J)=A(I,J)
      QHAT(I,J)=Q(I,J)
      DO 1250 N1=1,NC
      AHAT(I,J)=AHAT(I,J)-B(I,N1)*K(J,N1)
      DO 1250 N2=1,NC
 1250 QHAT(I,J)=QHAT(I,J)+K(I,N1)*R(N1,N2)*K(J,N2)
      CALL STRAM
 1260 CALL FEFN
      IT=IT+1
      CALL GAINFN(GF)
      WRITE (3,62) GF
      CALL NESCON
      CALL GRADNT
      CALL INVERT(VGRAD,GRADI,NFC)
      WRITE (3,30)
      WRITE (3,10) ((GRADI(I,J),J=1,NFC),I=1,NFC)
      CALL NEWRIT
      WRITE (3,85) IT
      WRITE (3,40)
      WRITE (3,10) ((G(I,J),J=1,NC),I=1,NF)
      IT1=0
 1285 CONTINUE
      DO 1300 I=1,NS
      DO 1300 J=1,NF
      AHAT(I,J)=A(I,J)
      QHAT(I,J)=Q(I,J)
      DO 1300 N1=1,NC
      AHAT(I,J)=AHAT(I,J)-B(I,N1)*G(J,N1)
      DO 1300 N2=1,NC
```

```
 1300 GHAT(I,J)=GHAT(I,J)+G(I,N1)*R(N1,N2)*G(J,N2)
      CALL STRAN
      CALL GAIN2(GF2)
      IF(GF-GF2.GT.0.000) GO TO 1380
      WRITE (3,92) GF2
      WRITE (3,93)
      DO 1350 I=1,NF
      DO 1350 J=1,NC
 1350 G(I,J)=G(I,J)/2.+K(I,J)/2.
      WRITE (3,40)
      WRITE (3,10) ((G(I,J),J=1,NC),I=1,NF)
      IT1=IT1+1
      IF(IT1.LE.5) GO TO 1280
      GO TO 3000
 1380 TEST1=0.00
      DO 2000 I=1,NF
      DO 2000 J=1,NC
      IF(G(I,J).EQ.0.000) GO TO 1400
      TEST=DABS((G(I,J)-K(I,J))/G(I,J))
      TEST1=DMAX1(TEST,TEST1)
      GO TO 2000
 1400 IF(G(I,J)-K(I,J).EQ.0.000) GO TO 2000
      TEST1=1000.*GNSTOP
 2000 K(I,J)=G(I,J)
      WRITE (3,60) TEST1
      WRITE (3,70) GNSTOP
      IF(TEST1-GNSTOP.GT.0.000) GO TO 1260
      WRITE (3,90)
      WRITE (3,62) GF2
      GO TO 8000
 3000 WRITE (3,95)
 8000 CONTINUE
 9000 CONTINUE
      STOP
      END
```

```
      SUBROUTINE STRAM
C
C        COMPUTES THE STATE TRANSITION MATRIX
C
      COMMON M(3,3),MI(3,3),RC(3),AHAT(3,3),T,K(3,3),G(3,3),QHAT(3,3),
     1       FT(3,3),FFTI(3,3),DFT(3,3,3,3),FDFTI(3,3,3,3),
     2       D2FT(3,3,3,3,3,3),DFDFTI(3,3,3,3,3,3),FD2FTI(3,3,3,3,3,3),
     3       VW(9),VGRAD(9,9),GRADI(9,9),B(3,3),F(3,3),R(3,3),NS,NC,NF
      DIMENSION AAAA(9),RR(3),RI(3),ASQR(3,3),ASQ2(3,3),XR(3),XI(3),VR(3
     1           ),VI(3),IANA(3),IROW(3,2),VRN(3),VIN(3),W(3,4)
      COMPLEX*16 M,MI,RC
      COMPLEX*16 DCMPLX,DCONJG
      DOUBLE PRECISION AHAT,T,K,G,QHAT,FT,FFTI,DFT,FDFTI,D2FT,DFDFTI,
     1                  FD2FTI,VW,VGRAD,GRADI
      DOUBLE PRECISION AAAA,RR,RI,ASQR,ASQ2,XR,XI,VR,VI,VRN,VIN,W,VECMGR
     1                 ,VECMGI,VECMGS,SW1
   10 FORMAT (2D18.7)
   30 FORMAT (//T3,'SYSTEM EIGENVALUES')
      CALL VECT(AHAT,AAAA,NS)
      CALL HSBG(NS,AAAA,NS)
      CALL ATEIG(NS,AAAA,RR,RI,IANA,NS)
      WRITE (3,30)
      WRITE (3,10) (RR(I),RI(I),I=1,NS)
      CALL MSC(AAAA,NS,ASQR)
      DO 100 I=1,NS
      RC(I)=DCMPLX(RR(I),RI(I))
      DO 100 J=1,NS
  100 ASQ2(I,J)=ASQR(I,J)
      CALL EIGVEC(3,AHAT,ASQR,W,IROW,XR,XI,VR,VI,RR(1),RI(1),NS,NS,0,
     1            SW1,ITER,DIF,2)
      VECMGR=0.0D0
      VECMGI=0.0D0
      DO 110 I=1,NS
      VECMGR=VECMGR+VR(I)*XR(I)-VI(I)*XI(I)
  110 VECMGI=VECMGI+VR(I)*XI(I)+VI(I)*XR(I)
      VECMGS=VECMGR*VECMGR+VECMGI*VECMGI
      DO 120 I=1,NS
      VRN(I)=(VR(I)*VECMGR+VI(I)*VECMGI)/VECMGS
      VIN(I)=(VI(I)*VECMGR-VR(I)*VECMGI)/VECMGS
      M(I,1)=DCMPLX(XR(I),XI(I))
  120 MI(1,I)=DCMPLX(VRN(I),VIN(I))
      DO 1000 KOUNT=2,NS
      KOUNT1=KOUNT-1
      IF(RR(KOUNT)-RR(KOUNT1)) 200,140,200
  140 CONTINUE
      DO 150 I=1,NS
      M(I,KOUNT)=DCONJG(M(I,KOUNT1))
  150 MI(KOUNT,I)=DCONJG(MI(KOUNT1,I))
      GO TO 1000
  200 DO 210 I=1,NS
      DO 210 J=1,NS
  210 ASQR(I,J)=ASQ2(I,J)
      CALL EIGVEC(3,AHAT,ASQR,W,IROW,XR,XI,VR,VI,RR(KOUNT),RI(KOUNT),NS,
     1            NS,0,SW1,ITER,DIF,2)
      VECMGR=0.0D0
```

```
      VECMGI=0.00
      DO 220 I=1,NS
      VECMGR=VECMGR+VR(I)*XR(I)-VI(I)*XI(I)
  220 VECMGI=VECMGI+VR(I)*XI(I)+VI(I)*XR(I)
      VECMGS=VECMGR*VECMGR+VECMGI*VECMGI
      DO 230 I=1,NS
      VRN(I)=(VR(I)*VECMGR+VI(I)*VECMGI)/VECMGS
      VIN(I)=(VI(I)*VECMGR-VR(I)*VECMGI)/VECMGS
      M(I,KOUNT)=DCMPLX(XR(I),XI(I))
  230 MI(KOUNT,I)=DCMPLX(VRN(I),VIN(I))
 1000 CONTINUE
      RETURN
      END
```

```
            SUBROUTINE FEFN
C
C        COMPUTES REQUIRED FUNCTIONS OF THE STATE TRANSITION MATRIX
C
         COMMON M(3,3),MI(3,3),RC(3),AHAT(3,3),T,K(3,3),G(3,3),GHAT(3,3),
     1         FT(3,3),FFTI(3,3),DFT(3,3,3,3),FDFTI(3,3,3,3),
     2         D2FT(3,3,3,3,3,3),DFDFTI(3,3,3,3,3,3),FD2FTI(3,3,3,3,3,3),
     3         VW(9),VGRAD(9,9),GRADI(9,9),B(3,3),F(3,3),R(3,3),NS,NC,NF
         DIMENSION D1(3,3,3,3,3,3),D2(3,3,3,3,3,3)
         COMPLEX*16 EX1(3),EX2(3,3),R1(3,3),R2(3,3),DUM1,DUM2,DUM3,DUM4,
     1         DUM5,DUM6,DUM7
         COMPLEX*16 M,MI,RC
         COMPLEX*16 CDEXP
         DOUBLE PRECISION AHAT,T,K,G,GHAT,FT,FFTI,DFT,FDFTI,D2FT,DFDFTI,
     1                    FD2FTI,VW,VGRAD,GRADI
         DOUBLE PRECISION D1,D2
         NS1=NS-1
         NF1=NF-1
         NC1=NC-1
         DO 100 I=1,NS
         EX1(I)=CDEXP(RC(I)*T)
         DO 100 J=1,NS
         EX2(I,J)=CDEXP((RC(I)+RC(J))*T)
         R1(I,J)=RC(I)+RC(J)
  100    R2(I,J)=RC(I)-RC(J)
         DO 1000 I=1,NS
         DO 1000 J=1,NS
         FT(I,J)=0.D0
         FFTI(I,J)=0.D0
         DO 300 K1=1,NF
         DO 300 K2=1,NC
         DFT(I,J,K1,K2)=0.D0
         FDFTI(I,J,K1,K2)=0.D0
         DO 300 K3=1,NF
         DO 300 K4=1,NC
         D2FT(I,J,K1,K2,K3,K4)=0.D0
         DFDFTI(I,J,K1,K2,K3,K4)=0.D0
  300    FD2FTI(I,J,K1,K2,K3,K4)=0. D0
         DO 1000 N1=1,NS
         FT(I,J)=FT(I,J)+M(I,N1)*MI(N1,J)*EX1(N1)
         DO 1000 N2=1,NS
         DO 1000 N3=1,NS
         DUM1=M(I,N1)*MI(N1,N2)*M(J,N3)
         FFTI(I,J)=FFTI(I,J)+DUM1*MI(N3,N2)*(EX2(N1,N3)-1.)/R1(N1,N3)
         DO 600 K1=1,NF
         DO 600 K2=1,NC
         DUM2=M(I,N1)*MI(N1,N2)*B(N2,K2)*M(K1,N3)
         IF(N1-N3) 320,310,320
  310    DFT(I,J,K1,K2)=DFT(I,J,K1,K2)-DUM2*MI(N3,J)*T*EX1(N1)
         GO TO 330
  320    DFT(I,J,K1,K2)=DFT(I,J,K1,K2)-DUM2*MI(N3,J)*(EX1(N3)-EX1(N1))/R2(N
        13,N1)
  330    DO 600 N4=1,NS
         DO 600 N5=1,NS
         DUM3=DUM1*MI(N3,N4)*B(N4,K2)*M(K1,N5)*MI(N5,N2)
```

```
      IF(N3-N5) 350,340,350
  340 FDFTI(I,J,K1,K2)=FDFTI(I,J,K1,K2)-DUM3*(EX2(N1,N3)*(R1(N1,N3)*T-1.
     1)+1.)/(R1(N1,N3)*R1(N1,N3))
      GO TO 360
  350 FDFTI(I,J,K1,K2)=FDFTI(I,J,K1,K2)-DUM3*((EX2(N1,N5)-1.)/R1(N1,N5)-
     1(EX2(N1,N3)-1.)/R1(N1,N3))/R2(N5,N3)
  360 DO 600 K3=1,NF
      DO 600 K4=1,NC
      DUM4=2.*DUM2*MI(N3,N4)*B(N4,K4)*M(K3,N5)*MI(N5,J)
      IF(N3-N5) 400,370,400
  370 IF(N1-N3) 390,380,390
  380 D2FT(I,J,K1,K2,K3,K4)=D2FT(I,J,K1,K2,K3,K4)+DUM4*T*T*EX1(N1)/2.
      GO TO 455
  390 D2FT(I,J,K1,K2,K3,K4)=D2FT(I,J,K1,K2,K3,K4)+DUM4*((R2(N3,N1)*T-1.)
     1*EX1(N3)+EX1(N1))/(R2(N3,N1)*R2(N3,N1))
      GO TO 455
  400 CONTINUE
      IF(N1-N3) 420,410,420
  410 D2FT(I,J,K1,K2,K3,K4)=D2FT(I,J,K1,K2,K3,K4)-DUM4*T*EX1(N1)/R2(N5,N
     11)
      GO TO 430
  420 D2FT(I,J,K1,K2,K3,K4)=D2FT(I,J,K1,K2,K3,K4)-DUM4*(EX1(N3)-EX1(N1))
     1/(R2(N3,N1)*R2(N5,N3))
  430 IF(N1-N5) 440,450,440
  440 D2FT(I,J,K1,K2,K3,K4)=D2FT(I,J,K1,K2,K3,K4)+DUM4*(EX1(N5)-EX1(N1))
     1/(R2(N5,N1)*R2(N5,N3))
      GO TO 455
  450 D2FT(I,J,K1,K2,K3,K4)=D2FT(I,J,K1,K2,K3,K4)+DUM4*T*EX1(N5)/R2(N5,N
     13)
  455 DO 600 N6=1,NS
      DO 600 N7=1,NS
      DUM5=MI(N4,N5)*B(N5,K4)*M(K3,N6)*MI(N6,N7)
      DUM6=DUM2*MI(N3,N7)*M(J,N4)*DUM5
      DUM7=2.*M(I,N1)*MI(N1,N7)*M(J,N2)*MI(N2,N3)*B(N3,K2)*M(K1,N4)*DUM5
      IF(N1-N3) 490,460,490
  460 IF(N4-N6) 480,470,480
  470 DFDFTI(I,J,K1,K2,K3,K4)=DFDFTI(I,J,K1,K2,K3,K4)+DUM6*((T*T*R1(N1,N
     14)*R1(N1,N4)-2.*T*R1(N1,N4)+2.)*EX2(N1,N4)-2.)/(R1(N1,N4)*R1(N1,N4
     2)*R1(N1,N4))
      GO TO 520
  480 DFDFTI(I,J,K1,K2,K3,K4)=DFDFTI(I,J,K1,K2,K3,K4)+DUM6*(((R1(N1,N6)*
     1T-1.)*EX2(N1,N6)+1.)/(R1(N1,N6)*R1(N1,N6))-((R1(N1,N4)*T-1.)*EX2(N
     21,N4)+1.)/(R1(N1,N4)*R1(N1,N4)))/R2(N6,N4)
      GO TO 520
  490 IF(N4-N6) 510,500,510
  500 DFDFTI(I,J,K1,K2,K3,K4)=DFDFTI(I,J,K1,K2,K3,K4)+DUM6*(((R1(N3,N4)*
     1T-1.)*EX2(N3,N4)+1.)/(R1(N3,N4)*R1(N3,N4))-((R1(N1,N4)*T-1.)*EX2(N
     21,N4)+1.)/(R1(N1,N4)*R1(N1,N4)))/R2(N3,N1)
      GO TO 520
  510 DFDFTI(I,J,K1,K2,K3,K4)=DFDFTI(I,J,K1,K2,K3,K4)+DUM6*((EX2(N3,N6)-
     11.)/R1(N3,N6)-(EX2(N3,N4)-1.)/R1(N3,N4)-(EX2(N1,N6)-1.)/R1(N1,N6)+
     2(EX2(N1,N4)-1.)/R1(N1,N4))/(R2(N3,N1)*R2(N6,N4))
  520 IF(N4-N6) 560,530,560
  530 IF(N2-N4) 550,540,550
  540 FD2FTI(I,J,K1,K2,K3,K4)=FD2FTI(I,J,K1,K2,K3,K4)+.5*DUM7*(EX2(N1,N2
```

```
     1)*(T*T*R1(N1,N2)*R1(N1,N2)-2.*T*R1(N1,N2)+2.)-2.)/(R1(N1,N2)*R1(N1
     2,N2)*R1(N1,N2))
      GO TO 600
 550  FD2FTI(I,J,K1,K2,K3,K4)=FD2FTI(I,J,K1,K2,K3,K4)+DUM7*((EX2(N1,N4)*
     1(R1(N1,N4)*T-1.)+1.)/(R1(N1,N4)*R1(N1,N4))-(EX2(N1,N4)-1.)/(R1(N1,
     2N4)*R2(N4,N2))+(EX2(N1,N2)-1.)/(R1(N1,N2)*R2(N4,N2)))/R2(N4,N2)
      GO TO 600
 560  IF(N2-N4) 580,570,580
 570  FD2FTI(I,J,K1,K2,K3,K4)=FD2FTI(I,J,K1,K2,K3,K4)-DUM7*(EX2(N1,N2)*(
     1R1(N1,N2)*T-1.)+1.)/(R1(N1,N2)*R1(N1,N2)*R2(N6,N2))
      GO TO 590
 580  FD2FTI(I,J,K1,K2,K3,K4)=FD2FTI(I,J,K1,K2,K3,K4)-DUM7*((EX2(N1,N4)
     1-1.)/R1(N1,N4)-(EX2(N1,N2)-1.)/R1(N1,N2))/(R2(N4,N2)*R2(N6,N4))
 590  IF(N2-N6) 594,592,594
 592  FD2FTI(I,J,K1,K2,K3,K4)=FD2FTI(I,J,K1,K2,K3,K4)+DUM7*(EX2(N1,N6)*
     1(R1(N1,N6)*T-1.)+1.)/(R1(N1,N6)*R2(N6,N4)*R1(N1,N6))
      GO TO 600
 594  FD2FTI(I,J,K1,K2,K3,K4)=FD2FTI(I,J,K1,K2,K3,K4)+DUM7*((EX2(N1,N6)-
     11.)/R1(N1,N6)-(EX2(N1,N2)-1.)/R1(N1,N2))/(R2(N6,N2)*R2(N6,N4))
 600  CONTINUE
      DO 700 K1=1,NF
      DO 700 K2=1,NC
      DO 700 K3=1,NF
      DO 700 K4=1,NC
      D1(I,J,K1,K2,K3,K4)=D2FT(I,J,K1,K2,K3,K4)
 700  D2(I,J,K1,K2,K3,K4)=FD2FTI(I,J,K1,K2,K3,K4)
      DO 800 K1=1,NF
      DO 800 K2=1,NC
      DO 800 K3=1,NF
      DO 800 K4=1,NC
      D2FT(I,J,K1,K2,K3,K4)=(D1(I,J,K1,K2,K3,K4)+D1(I,J,K3,K4,K1,K2))/2.
 800  FD2FTI(I,J,K1,K2,K3,K4)=(D2(I,J,K1,K2,K3,K4)+D2(I,J,K3,K4,K1,K2))/
     12.
1000  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GAINFN(GF)
C
C        COMPUTES AVERAGE COST
C
      COMMON M(3,3),MI(3,3),RC(3),AHAT(3,3),T,K(3,3),G(3,3),QHAT(3,3),
     1       FT(3,3),FFTI(3,3),DFT(3,3,3,3),FDFTI(3,3,3,3),
     2       D2FT(3,3,3,3,3,3),DFDFTI(3,3,3,3,3,3),FD2FTI(3,3,3,3,3,3),
     3       VW(9),VGRAD(9,9),GRADI(9,9),P(3,3),F(3,3),W(3,3),NS,NC,NF
      COMPLEX*16 M,MI,RC
      DOUBLE PRECISION AHAT,T,K,G,QHAT,FT,FFTI,DFT,FDFTI,D2FT,DFDFTI,
     1                  FD2FTI,VW,VGRAD,GRADI
      DOUBLE PRECISION GF
      GF=0.0D0
      DO 1000 N1=1,NS
      DO 1000 N2=1,NS
      GF=GF+QHAT(N1,N2)*FFTI(N2,N1)
      DO 1000 N3=1,NS
 1000 GF=GF+F(N1,N2)*FT(N2,N3)*FT(N1,N3)
      GF=GF/NS
      RETURN
      END
```

```
      SUBROUTINE NESCON
C
C        COMPUTES NECESSARY CONDITIONS
C
      COMMON M(3,3),MI(3,3),RC(3),AHAT(3,3),T,K(3,3),G(3,3),QHAT(3,3),
     1      FT(3,3),FFTI(3,3),DFT(3,3,3,3),FDFTI(3,3,3,3),
     2      D2FT(3,3,3,3,3,3),DFDFTI(3,3,3,3,3,3),FD2FTI(3,3,3,3,3,3),
     3      VW(9),VGRAD(9,9),GRADI(9,9),B(3,3),F(3,3),R(3,3),NS,NC,NF
      COMPLEX*16 M,MI,RC
      DOUBLE PRECISION AHAT,T,K,G,CHAT,FT,FFTI,DFT,FDFTI,D2FT,DFDFTI,
     1                FD2FTI,VW,VGRAD,GRADI
   10 FORMAT (//T3,'NECESSARY CONDITIONS VECTOR')
   20 FORMAT (4D18.7)
      NFC=NF*NC
      DO 1100 I=1,NF
      DO 1100 J=1,NC
      IN=NF*(J-1)+I
      VW(IN)=0.0D0
      DO 1100 N1=1,NS
      DO 1000 N2=1,NS
      VW(IN)=VW(IN)+QHAT(N1,N2)*FDFTI(N2,N1,I,J)
      DO 1000 N3=1,NS
 1000 VW(IN)=VW(IN)+F(N1,N2)*FT(N2,N3)*DFT(N1,N3,I,J)
      DO 1100 N4=1,NC
 1100 VW(IN)=VW(IN)+R(J,N4)*K(N1,N4)*FFTI(N1,I)
      WRITE (3,10)
      WRITE (3,20) (VW(I),I=1,NFC)
      RETURN
      END
```

```
340          SUBROUTINE GRADNT
      C
      C          COMPUTES GRADIENT MATRIX
      C
341          COMMON M(3,3),MI(3,3),RC(3),AHAT(3,3),T,K(3,3),G(3,3),CHAT(3,3),
            1      FT(3,3),FFTI(3,3),DFT(3,3,3,3),FDFTI(3,3,3,3),
            2      D2FT(3,3,3,3,3,3),DFDFTI(3,3,3,3,3,3),FD2FTI(3,3,3,3,3,3),
            3      VW(9),VGRAD(9,9),GRADI(9,9),B(3,3),F(3,3),R(3,3),NS,NC,NF
342          COMPLEX*16 M,MI,RC
343          DOUBLE PRECISION AHAT,T,K,G,CHAT,FT,FFTI,DFT,FDFTI,D2FT,DFDFTI,
            1                FD2FTI,VW,VGRAD,GRADI
344      10  FORMAT (//T3,'GRADIENT MATRIX')
345      20  FORMAT (4D18.7)
346          NFC=NF*NC
347          DO 1100 I=1,NF
348          DO 1100 J=1,NC
349          IN=NF*(J-1)+I
350          DO 1100 K1=1,NF
351          DO 1100 K2=1,NC
352          ID=NF*(K2-1)+K1
353          VGRAD(IN,ID)=R(J,K2)*FFTI(K1,I)
354          DO 1100 N1=1,NS
355          DO 1000 N2=1,NS
356          VGRAD(IN,ID)=VGRAD(IN,ID)+CHAT(N1,N2)*(DFDFTI(N2,N1,K1,K2,I,J)+FD2
            1FTI(N2,N1,K1,K2,I,J))
357          DO 1000 N3=1,NS
358    1000  VGRAD(IN,ID)=VGRAD(IN,ID)+F(N1,N2)*(DFT(N2,N3,K1,K2)*DFT(N1,N3,I,J
            2)+FT(N2,N3)*D2FT(N1,N3,K1,K2,I,J))
359          DO 1100 N4=1,NC
360    1100  VGRAD(IN,ID)=VGRAD(IN,ID)+R(K2,N4)*K(N1,N4)*(FDFTI(N1,K1,I,J)+
            1           FDFTI(K1,N1,I,J)) + R(J,N4)*K(N1,N4)*
            2           (FDFTI(N1,I,K1,K2)+FDFTI(I,N1,K1,K2))
361          WRITE (3,10)
362          WRITE (3,20) ((VGRAD(I,J),J=1,NFC),I=1,NFC)
363          RETURN
364          END
```

```
      SUBROUTINE INVERT(A,B,N)
C
C        INVERTS A TO GIVE B
C
      DIMENSION A(9,9),B(9,9)
      DOUBLE PRECISION A,B,C,D,X
      DOUBLE PRECISION DABS
      IF(N-1) 100,100,101
  100 B(1,1)=1.0D0/A(1,1)
      RETURN
  101 DO 102 I=1,N
      DO 102 J=1,N
  102 B(I,J)=0.0D0
      DO 103 I=1,N
  103 B(I,I)=1.0D0
C     PICK UP PIVOT ELEMENT
      DO 114 K=1,N
      L=K
      IF(N-K) 110,110,104
  104 I=K+1
      DO 106 JJ=I,N
      IF(DABS(A(JJ,K))-DABS(A(L,K)))106,106,105
  105 L=JJ
  106 CONTINUE
      IF(L-K) 107,110,107
C     PERFORM ROW INTERCHANGE
  107 DO 108 J=K,N
      C=A(K,J)
      A(K,J)=A(L,J)
  108 A(L,J)=C
      DO 109 J=1,N
      C=B(K,J)
      B(K,J)=B(L,J)
  109 B(L,J)=C
C     COLUMN ELIMINATION
  110 DO 114 I=1,N
      IF(K-I) 111,114,111
  111 D=A(I,K)/A(K,K)
      DO 112 J=K,N
  112 A(I,J)=A(I,J)-D*A(K,J)
      A(I,K)=0.0D0
      DO 113 J=1,N
  113 B(I,J)=B(I,J)-D*B(K,J)
  114 CONTINUE
C     SOLVE FOR INVERSE
      DO 115 J=1,N
      DO 115 I=1,N
      X=B(I,J)/A(I,I)
  115 B(I,J)=X
      RETURN
      END
```

```
      SUBROUTINE NEWRIT
C
C        PERFORMS NEWTON RAPHSON ITERATION
C
      COMMON M(3,3),MI(3,3),RC(3),AHAT(3,3),T,K(3,3),G(3,3),GHAT(3,3),
     1       FT(3,3),FFTI(3,3),DFT(3,3,3,3),FDFTI(3,3,3,3),
     2       D2FT(3,3,3,3,3,3),DFDFTI(3,3,3,3,3,3),FD2FTI(3,3,3,3,3,3),
     3       VW(9),VGRAD(9,9),GRADI(9,9),B(3,3),F(3,3),R(3,3),NS,NC,NF
      COMPLEX*16 M,MI,RC
      DOUBLE PRECISION AHAT,T,K,G,GHAT,FT,FFTI,DFT,FDFTI,D2FT,DFDFTI,
     1                 FD2FTI,VW,VGRAD,GRADI
      DO 1000 I=1,NF
      DO 1000 J=1,NC
      G(I,J)=K(I,J)
      IN=NF*(J-1)+I
      DO 1000 K1=1,NF
      DO 1000 K2=1,NC
      ID=NF*(K2-1)+K1
 1000 G(I,J)=G(I,J)-GRADI(IN,ID)*VW(ID)
      RETURN
      END
```

```
      SUBROUTINE VECT(AHAT,AAAA,NS)
C
C     CONVERTS AHAT TO SINGLE SUBSCRIPT FORM AAAA
C
      DIMENSION AHAT(3,3),AAAA(9)
      DOUBLE PRECISION AHAT,AAAA
      DO 100 J=1,NS
      DO 100 I=1,NS
      K=(J-1)*NS+I
  100 AAAA(K)=AHAT(I,J)
      RETURN
      END
```

```
      SUBROUTINE GAIN2(GF2)
C
C        COMPUTES AVERAGE COST FOR CONVERGENCE CHECK
C
      COMMON M(3,3),MI(3,3),RC(3),AHAT(3,3),T,K(3,3),G(3,3),QHAT(3,3),
     1       FT(3,3),FFTI(3,3),DFT(3,3,3,3),FDFTI(3,3,3,3),
     2       D2FT(3,3,3,3,3,3),DFDFTI(3,3,3,3,3,3),FD2FTI(3,3,3,3,3,3),
     3       VW(9),VGRAD(9,9),GRADI(9,9),B(3,3),F(3,3),R(3,3),NS,NC,NF
      COMPLEX*16 EX1(3),EX2(3,3),R1(3,3),DUM1
      COMPLEX*16 M,MI,RC
      COMPLEX*16 CDEXP
      DOUBLE PRECISION AHAT,T,K,G,QHAT,FT,FFTI,DFT,FDFTI,D2FT,DFDFTI,
     1                 FD2FTI,VW,VGRAD,GRADI
      DOUBLE PRECISION GF2
      DO 500 I=1,NS
      EX1(I)=CDEXP(RC(I)*T)
      DO 500 J=1,NS
      EX2(I,J)=CDEXP((RC(I)+RC(J))*T)
  500 R1(I,J)=RC(I)+RC(J)
      DO 1000 I=1,NS
      DO 1000 J=1,NS
      FT(I,J)=0.0D0
      FFTI(I,J)=0.0D0
      DO 1000 N1=1,NS
      FT(I,J)=FT(I,J)+M(I,N1)*MI(N1,J)*EX1(N1)
      DO 1000 N2=1,NS
      DO 1000 N3=1,NS
      DUM1=M(I,N1)*MI(N1,N2)*M(J,N3)
 1000 FFTI(I,J)=FFTI(I,J)+DUM1*MI(N3,N2)*(EX2(N1,N3)-1.)/R1(N1,N3)
      GF2=0.0D0
      DO 2000 N1=1,NS
      DO 2000 N2=1,NS
      GF2=GF2+QHAT(N1,N2)*FFTI(N2,N1)
      DO 2000 N3=1,NS
 2000 GF2=GF2+F(N1,N2)*FT(N2,N3)*FT(N1,N3)
      GF2=GF2/NS
      RETURN
      END
```

```
      SUBROUTINE HSBG(N,A,IA)
C
C     CONVERTS A TC UPPER HESSENBERG FCRM
C
      DIMENSION A(9)
      DOUBLE PRECISION A,PIV,T,S
      DOUBLE PRECISION DABS
      L=N
      NIA=L*IA
      LIA=NIA-IA
   20 IF(L-3) 360,40,40
   40 LIA=LIA-IA
      L1=L-1
      L2=L1-1
      ISUB=LIA+L
      IPIV=ISUB-IA
      PIV=DABS(A(IPIV))
      IF(L-3) 90,90,50
   50 M=IPIV-IA
      DO 80 I=L,M,IA
      T=DABS(A(I))
      IF(T-PIV) 80,80,60
   60 IPIV=I
      PIV=T
   80 CONTINUE
   90 IF(PIV) 100,320,100
  100 IF(PIV-DABS(A(ISUB))) 180,180,120
  120 M=IPIV-L
      DO 140 I=1,L
      J=M+I
      T=A(J)
      K=LIA+I
      A(J)=A(K)
  140 A(K)=T
      M=L2-M/IA
      DO 160 I=L1,NIA,IA
      T=A(I)
      J=I-M
      A(I)=A(J)
  160 A(J)=T
  180 DO 200 I=L,LIA,IA
  200 A(I)=A(I)/A(ISUB)
      J=-IA
      DO 240 I=1,L2
      J=J+IA
      LJ=L+J
      DO 220 K=1,L1
      KJ=K+J
      KL=K+LIA
  220 A(KJ)=A(KJ)-A(LJ)*A(KL)
  240 CCNTINUE
      K=-IA
      DO 300 I=1,N
      K=K+IA
      LK=K+L1
```

```
      S=A(LK)
      LJ=L-IA
      DO 280 J=1,L2
      JK=K+J
      LJ=LJ+IA
  280 S=S+A(LJ)*A(JK)*1.0D0
  300 A(LK)=S
      DO 310 I=L,LIA,IA
  310 A(I)=0.0D0
  320 L=L1
      GO TO 20
  360 RETURN
      END
```

```
      SUBROUTINE ATEIG(M,A,RR,RI,IANA,IA)
C
C     COMPUTES ROOTS OF UPPER HESSENBERG MATRIX A
C
      DIMENSION A(9),RR(3),RI(3),PRR(2),PRI(2),IANA(3)
      DOUBLE PRECISION E7,E6,E10,DELTA,PRR,PRI,PAN,PAN1,R,S,T,A,U,V,RR,
     1                 RI,RMOD,EPS,D,G1,G2,G3,CAP,PSI1,PSI2,ALPHA,ETA
      DOUBLE PRECISION DABS,DSQRT,DMAX1
      INTEGER P,P1,Q
      E7=1.0D-8
      E6=1.0D-6
      E10=1.0D-10
      DELTA=0.5D0
      MAXIT=30
      N=M
   20 N1=N-1
      IN=N1*IA
      NN=IN+N
      IF(N1) 30,1300,30
   30 NP=N+1
      IT=0
      DO 40 I=1,2
      PRR(I)=0.0D0
   40 PRI(I)=0.0D0
      PAN=0.0D0
      PAN1=0.0D0
      R=0.0D0
      S=0.0D0
      N2=N1-1
      IN1=IN-IA
      NN1=IN1+N
      N1N=IN+N1
      N1N1=IN1+N1
   60 T=A(N1N1)-A(NN)
      U=T*T
      V=4.0D0*A(N1N)*A(NN1)
      IF(DABS(V)-U*E7) 100,100,65
   65 T=U+V
      IF(DABS(T)-DMAX1(U,DABS(V))*E6) 67,67,68
   67 T=0.0D0
   68 U=(A(N1N1)+A(NN))/2.0D0
      V=DSQRT(DABS(T))/2.0D0
      IF(T)140,70,70
   70 IF(U) 80,75,75
   75 RR(N1)=U+V
      RR(N)=U-V
      GO TO 130
   80 RR(N1)=U-V
      RR(N)=U+V
      GO TO 130
  100 IF(T)120,110,110
  110 RR(N1)=A(N1N1)
      RR(N)=A(NN)
      GO TO 130
  120 RR(N1)=A(NN)
```

```
      RR(N)=A(N1N1)
  130 RI(N)=0.0D0
      RI(N1)=0.0D0
      RI(N1)=0.0
      GO TO 160
  140 RR(N1)=U
      RR(N)=U
      RI(N1)=V
      RI(N)=-V
  160 IF(N2)1280,1280,180
  180 N1N2=N1N1-IA
      RMOD=RR(N1)*RR(N1)+RI(N1)*RI(N1)
      EPS=E10*DSQRT(RMOD)
      IF(DABS(A(N1N2))-EPS) 1280,1280,240
  240 IF(DABS(A(NN1))-E10*DABS(A(NN))) 1300,1300,250
  250 IF(DABS(PAN1-A(N1N2))-DABS(A(N1N2))*E6) 1240,1240,260
  260 IF(DABS(PAN-A(NN1))-DABS(A(NN1))*E6) 1240,1240,300
  300 IF(IT-MAXIT) 320,1240,1240
  320 J=1
      DO 360 I=1,2
      K=NP-I
      IF(DABS(RR(K)-PRR(I))+DABS(RI(K)-PRI(I))-DELTA*(DABS(RR(K))
     1     +DABS(RI(K)))) 340,360,360
  340 J=J+I
  360 CONTINUE
      GO TO (440,460,460,480),J
  440 R=0.0D0
      S=0.0D0
      GO TO 500
  460 J=N+2-J
      R=RR(J)*RR(J)
      S=RR(J)+RR(J)
      GO TO 500
  480 R=RR(N)*RR(N1)-RI(N)*RI(N1)
      S=RR(N)+RR(N1)
  500 PAN=A(NN1)
      PAN1=A(N1N2)
      DO 520 I=1,2
      K=NP-I
      PRR(I)=RR(K)
  520 PRI(I)=RI(K)
      P=N2
      IF(N-3)600,600,525
  525 IPI=N1N2
      DO 580 J=2,N2
      IPI=IPI-IA-1
      IF(DABS(A(IPI))-EPS) 600,600,530
  530 IPIP=IPI+IA
      IPIP2=IPIP+IA
      D=A(IPIP)*(A(IPIP)-S)+A(IPIP2)*A(IPIP+1)+R
      IF(D)540,560,540
  540 IF(DABS(A(IPI)*A(IPIP+1))*(DABS(A(IPIP)+A(IPIP2+1)-S)+DABS(A(IPIP2
     1     +2)))-DABS(D)*EPS) 620,620,560
  560 P=N1-J
  580 CONTINUE
```

```
 600 Q=P
     GO TO 680
 620 P1=P-1
     Q=P1
     IF(P1-1)680,680,650
 650 DO 660 I=2,P1
     IPI=IPI-IA-1
     IF(DABS(A(IPI))-EPS) 680,680,660
 660 Q=Q-1
 680 II=(P-1)*IA+P
     DO 1220 I=P,N1
     II1=II-IA
     IIP=II+IA
     IF(I-P)720,700,720
 700 IPI=II+1
     IPIP=IIP+1
     G1=A(II)*(A(II)-S)+A(IIP)*A(IPI)+R
     G2=A(IPI)*(A(IPIP)+A(II)-S)
     G3=A(IPI)*A(IPIP+1)
     A(IPI+1) =0.0D0
     GO TO 780
 720 G1=A(II1)
     G2=A(II1+1)
     IF(I-N2)740,740,760
 740 G3=A(II1+2)
     GO TO 780
 760 G3=0.0D0
 780 CAP=DSQRT(G1*G1+G2*G2+G3*G3)
     IF(CAP)800,860,800
 800 IF(G1)820,840,840
 820 CAP=-CAP
 840 T=G1+CAP
     PSI1=G2/T
     PSI2=G3/T
     ALPHA=2.0D0/(1.0D0+PSI1*PSI1+PSI2*PSI2)
     GO TO 880
 860 ALPHA=2.0D0
     PSI1=0.0D0
     PSI2=0.0D0
 880 IF(I-Q)900,960,900
 900 IF(I-P)920,940,920
 920 A(II1)=-CAP
     GO TO 960
 940 A(II1)=-A(II1)
 960 IJ=II
     DO 1040 J=I,N
     T=PSI1*A(IJ+1)
     IF(I-N1)980,1000,1000
 980 IP2J=IJ+2
     T=T+PSI2*A(IP2J)
1000 ETA=ALPHA*(T+A(IJ))
     A(IJ)=A(IJ)-ETA
     A(IJ+1)=A(IJ+1)-PSI1*ETA
     IF(I-N1)1020,1040,1040
1020 A(IP2J)=A(IP2J)-PSI2*ETA
```

```
1040 IJ=IJ+IA
     IF(I-N1)1080,1060,1060
1060 K=N
     GO TO 1100
1080 K=I+2
1100 IP=IIP-I
     DO 1180 J=Q,K
     JIP=IP+J
     JI=JIP-IA
     T=PSI1*A(JIP)
     IF(I-N1)1120,1140,1140
1120 JIP2=JIP+IA
     T=T+PSI2*A(JIP2)
1140 ETA=ALPHA*(T+A(JI))
     A(JI)=A(JI)-ETA
     A(JIP)=A(JIP)-ETA*PSI1
     IF(I-N1)1160,1180,1180
1160 A(JIP2)=A(JIP2)-ETA*PSI2
1180 CONTINUE
     IF(I-N2)1200,1220,1220
1200 JI=II+3
     JIP=JI+IA
     JIP2=JIP+IA
     ETA=ALPHA*PSI2*A(JIP2)
     A(JI)=-ETA
     A(JIP)=-ETA*PSI1
     A(JIP2)=A(JIP2)-ETA*PSI2
1220 II=IIP+1
     IT=IT+1
     GO TO 60
1240 IF(DABS(A(NN1))-DABS(A(N1N2))) 1300,1280,1280
1280 IANA(N)=0
     IANA(N1)=2
     N=N2
     IF(N2)1400,1400,20
1300 RR(N)=A(NN)
     RI(N)=0.0D0
     IANA(N)=1
     IF(N1)1400,1400,1320
1320 N=N1
     GO TO 20
1400 RETURN
     END
```

```
      SUBROUTINE MSQ(AAAA,NS,ASQR)
C
C     COMPUTES ASQR = AAAA*AAAA
C
      DIMENSION AAAA(9),ASQR(3,3)
      DOUBLE PRECISION AAAA,ASQR
      DO 100 I=1,NS
      DO 100 J=1,NS
      ASQR(I,J)=0.0D0
      DO 100 K=1,NS
      K1=NS*(J-1)+K
      K2=NS*(K-1)+I
  100 ASQR(I,J)=ASQR(I,J)+AAAA(K1)*AAAA(K2)
      RETURN
      END
```

```
      SUBROUTINE EIGVEC(IVC,  A, B, W, IROW, XR, XI, VR, VI, ROOTRE,
     1   ROOTIE, NE, AMAX, T2, SW1, COUNTE, ERR,MMM)
C        SUBROUTINE TO FIND THE EIGENVECTORS OF A NON-SYMMETRIC MATRIX
C        BY A MODIFIED WILKINSON'S INVERSE ITERATION METHOD.
C        CONTROL IVC CODE IS
C              1  FIND ONLY THE REGULAR EIGENVECTORS    (A X = LAMBDA X)
C              2  FIND ONLY THE TRANSPOSED EIGENVECTORS (AT V = LAMBDA V)
C              3  FIND BOTH TYPES OF EIGENVECTORS.
      DIMENSION A(3,3),B(3,3),W(3,2),XR(3),XI(3),VR(3),VI(3),IROW(3,2)
      DOUBLE PRECISION ROOTR,ROOTI,ROOTRE,ROOTIE,TEMP,TEMP2,AMAX,C1,C2,
     1                 SW1,W,XR,XI,VR,VI,B,ZERO,CCERR,A
      DOUBLE PRECISION DABS,DSIGN,DSQRT,DMAX1
      INTEGER COUNT, COUNTE, T2
      I01=1
      I03=3
      ROOTR = ROOTRE
      ROOTI = ROOTIE
      N = NE
      MM = MMM - 1
      N1 = N - 1
      NP1 = N + 1
      IVC1 = IVC - 1
      IVC2 = IVC1 - 1
      COUNT = 1
      DO 400 I=1,N
      W(I,1)=0.0D0
      XR(I)=0.0D0
  400 CONTINUE
      CLIM = 1.0E-4
      IF(ROOTI) 1, 60, 1
C
C        COMPLEX EIGENVALUE.
C
    1 TEMP = - ROOTR - ROOTR
      ISW = 2
      TEMP2=ROOTR*ROOTR+ROOTI*ROOTI
      JJ = 300
      DO 606 I = 1, N
      IF(T2) 600, 603, 600
  600 DO 602 J = 1, N
      JJ = JJ + 1
      IF(JJ - 251) 602, 601, 601
  601 JJ = 1
      READ (T2) (W(LL,1), LL = 1,250)
  602 B(I,J) = A(I,J)*TEMP + W(JJ,1)
      GO TO 605
  603 DO 604 J = 1, N
  604 B(I,J) = A(I,J)*TEMP + B(I,J)
  605 B(I,I) = B(I,I) + TEMP2
  606 A(I,I) = A(I,I) - ROOTR
      IF(T2 .NE. 0) REWIND T2
      GO TO 700
  607 IF(ICC) 622, 608, 622
C
C        MATRIX SINGULAR.
```

```
C                                                                       ES
  622 IF(IVC2) 623, 625, 623                                            ES
  623 DO 624 LL = 1, N                                                  ES
      W(LL,2)=0.0D0
  624 XI(LL)=0.0D0
      IF(IVC1) 625, 514, 625                                            ES
  625 DO 626 LL = 1, N                                                  ES
      W(LL,4)=0.0D0
  626 VI(LL)=0.0D0
      GO TO 511                                                         ES
C                                                                       ES
C        MATRIX NOT SINGULAR.                                           ES
C                                                                       ES
  608 DO 609 LL = 1, N                                                  ES
      W(LL,1)=1.0D0
      W(LL,2)=1.0D0
      W(LL,3)=1.0D0
  609 W(LL,4)=1.0D0
  699 IF(IVC2) 610, 612, 610                                            ES
  610 DO 611 I = 1, N                                                   ES
      I2 = IROW(I,2)                                                    ES
      XI(I2) = W(I,1)*RCOTI                                             ES
      DO 611 J = 1, N                                                   ES
  611 XI(I2) = XI(I2) + A(I,J)*W(J,2)                                   ES
      IF(IVC1) 612, 500, 612                                            ES
  612 DO 613 I = 1, N                                                   ES
      VI(I) = W(I,3)*RCOTI                                              ES
      DO 613 J = 1,N                                                    ES
  613 VI(I) = VI(I) + A(J,I)*W(J,4)                                     ES
      GO TO 499                                                         ES
  615 CERR = 0.0                                                        ES
      DCERR=0.0D0
      IF(IVC2) 616, 619, 616                                            ES
  616 DO 618 I = 1, N                                                   ES
      XR(I) = -W(I,2)                                                   ES
      DO 617 J = 1, N                                                   ES
  617 XR(I) = XR(I) + A(I,J)*XI(J)                                      ES
  618 XR(I) = XR(I)/RCOTI                                               ES
      IF(IVC1) 619, 633, 619                                            ES
  619 DO 621 I = 1, N                                                   ES
      VR(I) = -W(I,4)                                                   ES
      DO 620 J = 1, N                                                   ES
  620 VR(I) = VR(I) + A(J,I)*VI(J)                                      ES
  621 VR(I) = VR(I)/RCOTI                                               ES
C                                                                       ES
C        SEARCH VECTORS FOR LARGEST ELEMENT AND NORMALIZE.              ES
C                                                                       ES
  627 AMAX=0.0D0
      DO 629 L = 1, N                                                   ES
      TEMP = VR(L)**2 + VI(L)**2                                        ES
      IF(TEMP - AMAX) 629, 629, 628                                     ES
  628 AMAX = TEMP                                                       ES
      I2 = L                                                            ES
  629 CONTINUE                                                          ES
      C1 = VR(I2)/AMAX                                                  ES
```

```
      C2 = -VI(I2)/AMAX
      DO 630 L = 1, N
      TEMP = VI(L)
      VI(L) = VR(L)*C2 + TEMP*C1
  630 VR(L) = VR(L)*C1 - TEMP*C2
      IF(COUNT .EQ. 1) GO TO 632
      DO 631 LL = 1, N
 631  DCERR=DMAX1(DCERR,DABS(VR(LL)-W(LL,3)),DABS(VI(LL)-W(LL,4)))
  632 IF(IVC2) 633, 638, 633
 633  AMAX=0.D0
      DO 635 L = 1, N
      TEMP = XR(L)**2 + XI(L)**2
      IF(TEMP - AMAX) 635, 635, 634
  634 AMAX = TEMP
      I2 = L
  635 CONTINUE
      C1 = XR(I2)/AMAX
      C2 = -XI(I2)/AMAX
      DO 636 L = 1, N
      TEMP = XI(L)
      XI(L) = XR(L)*C2 + TEMP*C1
  636 XR(L) = XR(L)*C1 - TEMP*C2
      IF(COUNT .EQ. 1) GO TO 646
      DO 637 LL = 1, N
 637  DCERR=DMAX1(DCERR,DABS(XR(LL)-W(LL,1)),DABS(XI(LL)-W(LL,2)))
C
C        TEST FOR CONVERGENCE.
C
  638 IF(COUNT .EQ. 1) GO TO 646
      CERR=DCERR
      IF(CERR .GE. 1.0E-4) GO TO 639
      IF(CERR .GE. CLIM) GO TO 648
      CLIM = CERR
      IF(CLIM .LE. 1.0E-8) GO TO 648
  639 IF(COUNT .GE. 15) GO TO 68
  647 COUNT = COUNT + 1
      IF(RCOTI) 642, 673, 642
  642 IF(IVC2) 643, 644, 643
  643 DO 641 LL = 1, N
      W(LL,1) = XR(LL)
  641 W(LL,2) = XI(LL)
      IF(IVC1) 644, 610, 644
  644 DO 645 LL = 1, N
      W(LL,3) = VR(LL)
  645 W(LL,4) = VI(LL)
      GO TO 659
  646 CERR = 0.0
      DCERR=0.D0
      IF(ICC) 648, 647, 648
  648 ERR = CERR
      COUNTE = COUNT
      IF(RCOTI) 667, 668, 667
  667 DO 649 I = 1, N
  649 A(I,I) = A(I,I) + RCOTR
      RETURN
```

```
   68 PRINT 1C1, RCCTR, RCCTI, CERR
      GO TC 648
C
C        REAL EIGENVECTCRS.
C
   60 ISW = 1
      DO 651 I = 1, N
      DO 65C J = 1, N
  65C B(I,J) = A(I,J)
  651 B(I,I) = B(I,I) - RCCTR
      GO TO 700
  652 IF(ICC) 680, 685, 680
C
C        SINGULAR MATRIX.
C
  680 IF(IVC2) 681, 683, 681
  681 DO 682 L = 1, N
  682 XI(L)=0.0DO
      IF(IVC1) 683, 514, 683
  683 DO 684 L = 1, N
  684 VI(L)=0.0DC
      GO TC 511
C
C        MATRIX NOT SINGLLAR.
C
  685 IF(IVC2) 653, 656, 653
  653 DO 654 L = 1, N
  654 XI(L)=1.0DO
      IF(IVC1) 656, 500, 656
  656 DO 657 L = 1, N
  657 VI(L)=1.0DO
      GO TO 499
C
C        NCRMALIZE REAL VECTCRS.
C
  655 CERR = C.C
      DCERR=0.0DO
      IF(IVC2) 658, 662, 658
  658 C1=C.0DC
      C2=C.0DC
      DO 660 L = 1, N
      TEMP=DABS(XI(L))
      IF(TEMP - C1) 660, 660, 659
  659 C1 = TEMP
      C2 = XI(L)
  660 CCNTINUE
      DC 661 L = 1, N
      XI(L) = XI(L)/C2
      DCERR=DMAX1(DCERR,DABS(XI(L)-XR(L)))
  661 XR(L) = XI(L)
      IF(IVC1) 662, 638, 662
  662 C2=0.0DC
      C1=C.0DC
      DO 664 L = 1, N
      TEMP=DABS(VI(L))
```

```
      IF(TEMP - C1) 664, 664, 663                                    ES'
  663 C1 = TEMP                                                      ES'
      C2 = VI(L)                                                     ES'
  664 CONTINUE                                                       ES'
      DO 665 LL = 1, N                                               ES'
      VI(LL) = VI(LL)/C2                                             ES'
      DCERR=DMAX1(DCERR,CABS(VI(LL)-W(LL,1)))
      W(LL,1)=VI(LL)
  665 VR(LL)=W(LL,1)
      GO TO 638                                                      ES'
  668 IF(IVC2) 669, 671, 669                                         ES'
  669 DO 670 L = 1, N                                                ES'
 670  XI(L)=0.0D0
      IF(IVC1) 671, 70, 671                                          ES'
  671 DO 672 L = 1, N                                                ES'
 672  VI(L)=0.0D0
   70 RETURN                                                         ES'
  673 IF(IVC2) 674, 502, 674                                         ES'
  674 DO 675 I = 1, N                                                ES'
      I2 = IROW(I,2)                                                 ES'
  675 XI(I2) = XR(I)                                                 ES'
      GO TO 500                                                      ES'
C                                                                    ES'
C          BACK SUBSTITUTION SECTION.                                ES'
C                                                                    ES'
  499 IF(IVC2) 500, 502, 500                                         ES'
  500 DO 501 I = 2, N                                                ES'
      I1 = I - 1                                                     ES'
      DO 501 J = 1, I1                                               ES'
  501 XI(I) = XI(I) - B(I,J)*XI(J)                                   ES'
  511 IF(IVC1) 502, 514, 502                                         ES'
  502 DO 510 I = 1, N                                                ES'
      I1 = I - 1                                                     ES'
      IF(I1) 503, 505, 503                                           ES'
  503 DO 504 J = 1, I1                                               ES'
  504 VI(I) = VI(I) - B(J,I)*VI(J)                                   ES'
      IF(ICC) 505, 506, 505                                          ES'
  505 IF(B(I,I)) 506, 507, 506                                       ES'
  506 VI(I) = VI(I)/B(I,I)                                           ES'
      GO TO 510                                                      ES'
  507 IF(VI(I)) 508, 509, 508                                        ES'
  508 VI(I) = VI(I)*1.0E+15                                          ES'
      GO TO 510                                                      ES'
  509 VI(I) = 1.0                                                    ES'
  510 CONTINUE                                                       ES'
      IF(IVC2) 514, 525, 514                                         ES'
  514 DO 522 I = 1, N                                                ES'
      IR = NP1 - I                                                   ES'
      IF(I - 1) 515, 517, 515                                        ES'
  515 I2 = IR + 1                                                    ES'
      DO 516 J = I2, N                                               ES'
  516 XI(IR) = XI(IR) - B(IR,J)*XI(J)                                ES'
      IF(ICC) 517, 518, 517                                          ES'
  517 IF(B(IR,IR)) 518, 519, 518                                     ES'
  518 XI(IR) = XI(IR)/B(IR,IR)                                       ES'
```

```
      GO TC 522
  519 IF(XI(IR)) 52C, 521, 520
  52C XI(IR) = XI(IR)*1.CE+15
      GC TC 522
 521  XI(IR)=1.0DC
  522 CCNTINUE
      IF(IVC1) 525, 529, 525
  525 DO 526 I = 2, N
      IR = NP1 - I
      I2 = IR + 1
      DO 526 J = I2, N
  526 VI(IR) = VI(IR) - B(J,IR)*VI(J)
      DO 527 L = 1, N
      I2 = IROW(L,1)
  527 VR(I2) = VI(L)
      DO 528 L = 1, N
  528 VI(L) = VR(L)
  529 IF(RCOTI) 615, 656, 615
C
C        FACTOR MATRIX.
C
  700 ICC = 0
      SW1=1.0D72
      DO 701 LL = 1, N
  701 IROW(LL,1) = LL
      DC 708 K = 1, N1
      AMAX=DABS(B(K,K))
      IMAX = K
      K1 = K + 1
      DO 702 I = K1, N
      IF(AMAX.GT.DABS(B(I,K))) GO TC 702
      AMAX=DABS(B(I,K))
      IMAX = I
  702 CONTINUE
      IF(AMAX .LT. SW1) SW1 = AMAX
      IF(AMAX.GE.1.0D-25) GO TC 723
      B(K,K)=0.0D0
      ICC = ICC + 1
      GO TO 7C8
  723 IF(IMAX .EC. K) GO TC 704
      DC 703 J = 1, N
      AMAX = B(K,J)
      B(K,J) = B(IMAX,J)
  703 B(IMAX,J) = AMAX
      I2 = IROW(K,1)
      IROW(K,1) = IROW(IMAX,1)
      IROW(IMAX,1) = I2
  704 DO 707 I = K1, N
      IF(B(I,K)) 705, 707, 705
  705 B(I,K) = B(I,K)/B(K,K)
      DO 706 J = K1, N
  706 B(I,J) = B(I,J) - B(K,J)*B(I,K)
  707 CONTINUE
  708 CONTINUE
      AMAX=DABS(B(N,N))
```

```
1031         IF(AMAX-1.0D-25) 712,712,713
1032    712  B(N,N)=0.0D0
1033         SW1=C.0D0
1034         ICC = ICC + 1                                                      ES
1035         GO TC 709                                                          ES
1036     713 IF(AMAX .LT. SW1) SW1 = AMAX                                       ES
1037     709 IF(ICC .LE. ISW) GC TG 710                                         ES
1038         IF(MP) 105C,1C50,1051
1039    1051 WRITE(IO3,1C2) ICC
1040         COUNTE = C                                                         ES
1041         RETURN                                                             ES
1042.   1050 WRITE(IO3,1052) ICC
1043     710 DO 711 LL = 1, N                                                   ES
1044         I2 = IRCW (LL,1)                                                   ES
1045     711 IROW(I2,2) = LL                                                    ES
1046         IF(RCOTI) 6C7, 652, 607                                            ES
1047    1052 FORMAT(///23H ****** WARNING ******     ,'   SUBROUTINE EIGVEC HAS ES
            1FOUNC AN EIGENVALUE CF APPARENT MULTIPLICITY',                     ES
            1                                          I4,/23X,'  COMPUTATION OF EIES
            2GENVECTOR(S) CONTINUES AT USER S OPTION'//)                        ES
1048     101 FORMAT(38HOMORE THAN 15 LCCPS FOR EIGENVECTOR OF,2E12.4,           ES
            2    14H DIFFERENCE CF,E12.4)                                       ES
1049     102 FORMAT(16HO****WARNING**** , I4, 71H ZERCS ON DIAGONAL OF FACTOREDES
            1 MATRIX.  CHECK FOR MULTIPLE EIGENVALUES./20X,                     ES
            2' SUBROUTINE EIGVEC WILL NOT PERFORM COMPUTATION FCR THIS EIGENVECES
            3TOR '//)                                                           ES
1050         END
      /DATA
```