This Research was Sponsored in Part by
The National Aeronautics and Space Administration
Research Grant No. NsG 110-61

NsG-110

SIMPLE SHOCK ISOLATOR SYNTHESIS WITH BILINEAR STIFFNESS AND VARIABLE DAMPING

EDC Report No. 2-65-9

by

Lucien A. Schmit, Jr.

and

Edmund F. Rybicki

June 1965

Engineering
Synthesis
Group

ABSTRACT

32885

An extension of the previously reported synthesis capability for a simple shock isolator is presented. Advances in the engineering scope and the algorithmic efficiency of the previous work are offered. A one degree of freedom system with a single package of mass M is to be protected from a multiplicity of shock pulses. Two common situations are considered. In the first type of problem a design is sought which minimizes the absolute acceleration felt by the package subject to relative displacement limitations. In the second type of problem a design is sought which minimizes the relative displacement subject to limitation on the absolute acceleration felt by the package. Three design variables are employed to characterize the bilinear spring and six additional design variables are used to represent a piecewise-linear variable damping coefficient. The synthesis technique employed is based on an implementation of the gradient projection method, with certain special additional features. Results for several numerical examples are presented. By permitting a broader class of possible designs it was found that a reduction of as much as 25% in the criterion function value, at termination of the synthesis, could be obtained in some cases.

Author

ACKNOWLEDGEMENTS

The authors express their appreciation to the following for their cooperation:

The National Aeronautics and Space Administration which sponsored the research under Grant No. NsG-110-61.

The Case Computing Center.

The Case Engineering Design Center, for providing a pleasant environment in which to work.

TABLE OF CONTENTS

SYMBOLS

$\bar{c}$	time dependent coefficient of damping dexcribed by C(0) thru C(5)
$\vec{u}_1$	component of gradient in direction normal to previous gradient
$\vec{g}$	gradient of criterion function
q_j	distance from design to j^{th} constraint
$\vec{u}$	modified gradient direction vector
x	absolute displacement of mass
$\ddot{x}A$	maximum allowable absolute acceleration
y	absolute displacement of base
z	relative displacement between mass and base
zA	absolute value of maximum allowable displacement
A_j	unit vector normal to j^{th} constraint pointing into the acceptable design region
C	constant damping coefficient
$\vec{C}$	new design to be analysis
$\vec{C1}$	best design obtained at any time
C(0) thru C(8)	variables describing piecewise linear damping with respect to time
CAL_j	allowable lower limit on j^{th} variable
CAU_j	allowable upper limit on j^{th} variable

CBT	absolute value of maximum allowable time rate of change of damping
DT	spacing in time of damping variables C(0) thru C(5)
K	constant spring constant
K(z)	force of bilinear spring system
K_1	first spring constant of bilinear spring system
K_2	sum of $K_1 + K'$
K'	second spring constant of bilinear spring system
L	maximum possible distance in given direction before encountering constraint
LC	number of pulses comprising the load condition
M	mass
$\underset{i}{\mathrm{Max}}\ \{\mathrm{Max}\lvert\ddot{x}_i\rvert\}$	maximum of maximum absolute accelerations for all load conditions
$\underset{i}{\mathrm{Max}}\ \{\mathrm{Max}\lvert z_i\rvert\}$	maximum of maximum relative displacements of given design for all load conditions
$\mathrm{Max}\ \lvert\ddot{x}_i\rvert$	maximum absolute acceleration for a given design and the ith load condition
$\mathrm{Max}\ \lvert z_i\rvert$	maximum relative displacement of given design for ith load condition
N	matrix with columns corresponding to vectors of normals to active constraints
$S_i(t)$	the ith shock pulse
δ	difference in lengths of springs

CHAPTER 1
INTRODUCTION

1.1 Relation to Previous Work

The research reported herein may be viewed as an extension in scope and algorithmic efficiency of a synthesis capability for the automated optimum design of one degree of freedom shock isolators. The previous work reported in Ref. 1 considered a constant spring stiffness and a damping coefficient as design variables to be determined by the synthesis process. Two distinct problem types were dealt with. In the first type the objective is to select the spring stiffness and the damping coefficient so as to minimize the maximum absolute acceleration of the package subject a prescribed limit on the maximum relative displacement. In the second type of problem the objective is to select the spring stiffness and the damping coefficient so as to minimize the maximum relative displacement of the package subject to a prescribed limit on the maximum absolute acceleration. The synthesis technique employed was a modification of the steep-descent alternate step methods (see Ref. 9). In the extension reported here the simple spring is replaced by a bilinear spring and the simple damper is replaced by a damping device capable of supplying a preprogrammed time dependent damping. This results in a synthesis problem having nine design variables.

The purpose of this study is to explore the influence of considering additional design variables which admit a wider class of possible designs than those previously considered.

1.2 Description of Problem

Consider the simple shock isolator of Figure 1. The spring system is comprised of two concentric springs of unequal lengths. Thus, the shorter spring is not compressed until the deflection of the mass exceeds the difference in spring lengths, δ. The characteristics of this spring system are represented by the bilinear force-displacement curve shown in Figure 2. Three design variables are required to describe the force-displacement curve K_1, K_2, and the gap, δ. Noting that K_2 is $K_1 + K^1$ from Figure 1, the force as a function displacement is given by $K(z) =$

$$K_1 z \quad \text{for } |z| < \delta$$

$$K_1 \delta + K_2(z-\delta) \quad \text{for } z > \delta$$

$$-K_1 \delta + K_2(z+\delta) \quad \text{for } |z| > \delta,\ z < 0$$

The coefficient of damping was chosen to be a piecewise linear continuous function of time. An illustration is shown in

Figure 3. This coefficient of damping has six design variables. The last value of the damping, C(5), continues on to $t \to \infty$. These six variables are termed C(0) thru C(5).

1.3 Formulation of Problem

The synthesis problem with the objective of minimizing the maximum absolute acceleration can be expressed as:

Given the preassigned values of M(M=1) and DT, find values of $C_o, \ldots, C_5, K_1, K_2$, and δ such that for $i = 1, 2, \ldots, LC$.

$$[\underset{i}{\text{Max}} \, (\text{Max} \, |\ddot{x}_i|)] \rightarrow \text{Min}$$

subject to the following side constraints,

$$\frac{|C_j - C_{j+1}|}{DT} \le CBT \qquad \text{for } j = 0, 1, 2, 3, 4$$

$$0 \le CAL_j \le C_j \le CAU_j \qquad \text{for } j = 0, 1, 2, 3, 4, 5,$$

$$0 \le CAL_6 \le K_1 \le CAU_6 \qquad K_1 \le K_2 \le CAU_7$$

$$0 \le \delta \le zA$$

behavior constraint,

$$\text{Max} \, \{z_i\} \le zA \qquad \text{for } i = 1, 2, \ldots, LC$$

and governing technology

$$\ddot{z}_i + \overline{c}\,\dot{z}_i + K(z_i) = S_i(t). \qquad i = 1, \ldots, LC$$

The synthesis problem of minimizing the maximum relative displacement can be stated in a similar form:

Given the preassigned values for M (M=1) and DT find values of $C_o, \ldots, C_5, K_1, K_2$, and δ such that for i = 1, 2, ..., LC

$$[\underset{i}{\text{Max}}\ \{\text{Max}\,|z_i|\}] \rightarrow \text{Min}$$

subject to the following side constraints,

$$\frac{|C_j - C_{j+1}|}{DT} \leq CBT \qquad \text{for } j = 0,1,2,3,4.$$

$$0 \leq CAL_j \leq C_j \leq CAU_j \qquad \text{for } j = 0,1,2,3,4,5$$

$$0 \leq CAL_6 \leq K_1 \leq CAU_6 \qquad K_1 \leq K_2 \leq CAU_7 \qquad 0 \leq \delta \leq zA$$

behavior constraint,

$$\max |\overline{c}\cdot\dot{z} + K(z)| \leq \ddot{x}A$$

and governing technology,

$$\ddot{z}_i + \overline{c}\cdot\dot{z}_i + K(z_i) = S_i(t) \qquad i = 1, \ldots, LC$$

CHAPTER 2
ANALYSIS

The equation of motion for the mass in Figure 1 is

$$M\ddot{x} = K(y - x) + \bar{c}\,(\dot{y} - \dot{x})$$

By making the substitutions $z = y - x$, $\dot{z} = \dot{y} - \dot{x}$, and $\ddot{z} = \ddot{y} - \ddot{x}$ the equation becomes

$$M\ddot{z} + \bar{c}\cdot\dot{z} + K(z) = M\ddot{y}. \qquad (1)$$

The mass, M, is taken to be 1.0, and $\ddot{y}$ is then viewed as an input acceleration, S(t). The acceleration felt by the mass is

$$\ddot{x} = \bar{c}\cdot\dot{z} + K(z).$$

Equation (1) is difficult to solve in closed form; therefore, a numerical integration (Runge-Kutta) was used. Details and examples of this method are given in Appendix I.

The analysis procedure terminates when a maximum displacement is found after the duration of the input pulse S(t). Because of the numerical approach used to solve Eq. (1), the type of pulse does not have to be confined to "square" pulses, although square pulses will be the only type employed in this paper.

CHAPTER 3
SYNTHESIS

The synthesis method answers two questions. Which direction to go from a given design point and how far to go in that direction.

3.1 Direction of Travel

If the present design is not on a constraint, the best direction to move in is the direction of the negative gradient of the criterion function (See Figure 4). The gradient of a function is a vector of the first partial derivatives of that function. It is the direction of most rapid increase in value of the function and similarly the negative gradient is the direction of most rapid decrease in value of the function. Because a closed form solution was not obtained for the equation of motion, an explicit function for the gradient was not available. A forward finite difference approximation to the gradient was obtained by increasing each of the variables individually and noting the change in criterion value per unit increase of each variable.

If the design point is constrained, then the synthesis procedure first determines whether it is advantageous to remain on the constraint or to get off of it. This is important because a constraint which is presently active may not be active at the optimum design. The result of remaining on a constraint, that is not active at the optimum design, is getting "cornered" at a vertex of constraints and never reaching the true optimum design.

This synthesis procedure avoids "cornering" by checking the inner product of the "negative gradient" and the normal vector to the active constraints. The normal to the constraint surface used here is the one which points into the acceptable design space. If the inner product is greater than zero,* then the new design is allowed to be off the constraint (See Figure 5). This test is executed for each active constraint at each point in the design path, since it is possible for the design path leading to the optimum design to travel along a constraint for a while and then leave the constraint. Looking at this another way, it is seen that a positive inner product means the angle between the two vectors is acute and the negative gradient has a component in the same direction as the constraint normal. Thus, it would be useless to move along the constraint when moving off it reaps more gain.

If the inner product described above is less than or equal to zero, it means a move in the negative gradient direction will

---

* The inner product of two vectors $\vec{A}$ and $\vec{B}$ is defined as a number equal to $\sum_i a_i b_i$.

violate the active constraint. In this case the best move (in the gradient sense) is in the direction given by the projection of the negative gradient on that constraint (See Figure 6). A method for finding this direction, $(\vec{u})$, is given in Refs. (2) and (3). This direction may be viewed as the direction of constrained steepest descent.

Fundamentally $\vec{u}$ is the vector which is the component of the gradient lying in the space orthogonal to the normals of the active constraints. That is, $\vec{u}$ is the component of the negative gradient, $-\vec{g}$, minus $N(N^T N)^{-1} N^T (-\vec{g})$ where N is a matrix composed of columns which are the unit vectors normal to the active constraints and pointing into the acceptable design region. In effect the component of the negative gradient which would pierce the unacceptable design region has been subtracted from the negative gradient to find the $\vec{u}$ direction (See Figure 6).

Figure 6 depicts the case where N is a single column, $\hat{A}_j$. Then $N^T N = \hat{A}_j \hat{A}_j = 1$, because $\hat{A}_j$ has been normalized to be a unit vector. $N^T(-\vec{g})$ is the component of $\hat{A}_j$ in the $(-\vec{g})$ direction or the component of $(-\vec{g})$ in the $\hat{A}_j$ direction or $\hat{A}_j \cdot (-\vec{g})$. The direction $\vec{u} = -\vec{g} - N(N^T N)^{-1} N^T(-\vec{g})$ is then

$$-\vec{g} - \hat{A}_j (1)^{-1} (\hat{A}_j \cdot (-\vec{g})).$$

The method of derivation of the $\vec{u}$ direction is described in detail in Appendix III.

3.2 Distance of Travel

The question how far to go is easily answered for the case of linear constraints.* The maximum distance that it is possible to go in a desired direction without entering the unacceptable region is denoted by L.

To illustrate this procedure, consider a general linear constraint

$$\sum_{i=0}^{8} a_{ij} C_i \leq b_j$$

where C_i are the variables of the vector C(0), C(1), . . . , C(8) and a_{ij} and b_j are constants. The normal to the j^{th} constraint is the vector $\hat{A}_j$ or $(a_{oj}, a_{1j}, \ldots, a_{8j})$. Figure 7 shows how $\vec{u}$, and $\hat{A}_j$ look in 2 dimensions. The perpendicular distance from the current design point to constraint j is called q_j in the sketch. The value of q_j for linear constraints is

$$b_j - \sum_{i=0}^{8} a_{ij} C_i .$$

For all the constraints the minimum of q_j divided by the absolute

* The following is taken from Ref. 2.

value of the inner product of $\vec{u}$ and $\hat{A}_j$,

$$\text{Min} \quad \frac{q_j}{|\hat{A}_j \cdot \vec{u}|} \quad ,$$

is sought to find L. Only constraints for which the inner product of $\vec{u}$ and $\hat{A}_j$ is < 0 are used because $(\vec{u}, A_j) \geq 0$ signifies no component of $\vec{u}$ will enter the unacceptable region by piercing the j^{th} constraint. The quantity L will always be > 0. Since the program is always in the acceptable region characterized by $\sum_{i=0}^{8} a_{ij} C_i \leq b_j$, q_j is always greater than or equal to zero and $|\vec{u} \cdot \hat{A}_j| > 0$.

A similar treatment employed for the nonlinear constraint by approximating it with its tangent hyperplane has been found to be very successful in the synthesis method. The procedure to find L is explained in more detail in Appendix III.

The length L does not necessarily produce an acceptable design because nonlinear constraints have been linearized to obtain it. (See Figure 8).

If the program is at point (1), which is on constraint i, the modified gradient $\vec{u}$ will be along constraint i. The associated length L from the procedure LINLEN will take the new design to point (2). Point (2), however, lies in the unacceptable region, with respect to the nonlinear constraint, but is seen to be acceptable with respect to the linearized approximation for the constraint j.

The synthesis program checks point (2) for both criterion value and acceptability. If either test is not passed, the length L is multiplied by 0.85 and a new move vector equal to 0.85 L $\vec{u}$ is used to generate a new design to be checked. This procedure is repeated until either an acceptable point with lower criterion value is found or until the length becomes less than 0.00001 of its original value (Maximum number of cycles is 52).

For an explanation of what the synthesis does if the latter occurs, the reader is referred to section 3.3.1.

3.3 Special Features

3.3.1 δ Difficulty.

As the synthesis progresses, it checks the "inlineness" of the negative gradient and the behavior constraint normal. It was found that with nine variables in the redesign cycle the inner product of the unit vectors corresponding to the negative gradient and the normal to the behavior constraint was often less than -0.999. This means the gradient and deflection bound are in a position similar to that in Figure 9.

It is seen that the component of the negative gradient which will not pierce the unacceptable region is very small. However, it was observed that the negative gradient component of the criterion with respect to the spring gap and the component of the normal to behavior function with respect to the gap were +0.999 and -0.999 respectively. The gap was then held fixed and the synthesis was carried on with the remaining eight variables. In doing this an assumption had to be made. Consider that the results of several eight dimensional optimizations, each with a distinct fixed value of δ, are available. It must be assumed that a plot of optimum criterion function values versus gap distance is unimodal (i.e. has one minimum over the allowable range of values for δ). The one dimensional search over δ is terminated when

$$|\delta_{new} - \delta_{current}| \leq 0.00001 \text{ in.}$$

In the synthesis problem where maximum absolute acceleration is the criterion function, special attention is given to the cases where $\delta_{new} > \delta_{current}$. For, as seen in Figure 10, increasing δ results in decreasing the over-all stiffness of the spring system. (i.e. choose any $x > \delta$ current and observe $K(z) |\delta_{new} < K(z)| \delta_{current}$.) The softer spring system (with $\delta = \delta_{new}$) will have a larger maximum relative displacement which may place the new design in violation of the relative displacement constraint.

If this violation occurs, a move (with the first eight variables) in the direction of the negative gradient to the behavior constraint will decrease the maximum relative displacement enough to make the design acceptable with respect to the behavior constraint. However, precautions must be taken to assure that the design is also acceptable with respect to the side constraints. This can easily be accomplished by employing the $\vec{u}$ direction (discussed in the synthesis chapter) with the negative gradient to the behavior constraint used in place of the negative gradient to the criterion function. Furthermore, it is seen that by using the criterion function as a constraint in obtaining the direction $\vec{u}$, the increase of the criterion value of the new acceptable point will not be as large as it would be if this constraint were omitted. (See Figure 11).

3.3.2 Hop Out. In the event of the $\vec{u}$ direction becoming zero, the program will examine a point a short distance from each of the active constraints to see if a non-zero $\vec{u}$ can be found. If it can, it is the new direction of travel. This is a precaution which does not have to be taken if the gradients are found exactly, because when $\vec{u}$ is a zero vector the design satisfies the Kuhn-Tucker (Appendix IV) conditions and a constrained minimum has been found (assuming the design space is convex).

3.3.3 Zig-Zag. Multiple load conditions for dynamic systems often cause the criterion function to have a discontinuous gradient. It was found that with multiple load conditions cusp areas similar to those shown in Figure 12 were encountered. Usually, no further progress can be made with the gradient method at a cusp point because the negative gradient points in a direction giving larger criterion values than that of the cusp point. (See Figure 12).

The direction of travel employed for the cusped region was obtained by performing a single step of the Schmidt orthogonalization process. This process is a method for obtaining mutually orthogonal vectors from a set of linearly independent vectors. The two linear independent vectors for the process are the negative gradients at two consecutive acceptable designs. The negative gradient of the first design is used as the base vector. Then the direction of travel becomes the component of the negative gradient at the second design which is orthogonal to the base vector. An example in two dimensions is shown in Figure 13.

This procedure may be viewed as treating the Pulse 2 contours as constraints and only the component of the negative gradient which does not pierce this "unacceptable" region is used. For more than two pulses the direction $\vec{u}_1$ may have to be further modified. If $\vec{g}_3$ is the gradient of the third pulse contour $\vec{u}_2 = -\vec{g}_3 - (\vec{g}_3 \cdot \vec{g}_2)(\vec{g}_2) - (-g_3 \cdot \vec{u}_1)\,\vec{u}_1$ where $\vec{g}_3$ is the negative gradient where pulse 3 is active. The synthesis procedure used the method whenever two consecutive designs have negative gradients which have an inner product less than (-0.70).

A description of the computer program and associated flow chart can be found in Appendix V.

CHAPTER 4
NUMERICAL EXAMPLES

4.1 Introduction

The purpose of the numerical examples is to examine the influence of increasing the number of design variables on the performance of the shock isolator with respect to the results previously reported in Ref. 1.

4.2 Test Case

First, an example case was used to test the computer synthesis program. The example considered is found in Ref. 1, page 24. This case involves only two variables - constant coefficient of damping and constant spring constant. There are two reasons why this particular case was chosen. It has the characteristic that at the optimum design the nonlinear deflection constraint is active and the normalized component of the gradient in the K direction is small. This small component invites many values of K to yield nearly the same criterion value. This means there is a long region where the deflection constraint and the criterion function contours nearly coincide. This is seen in Ref. 1, Figure 21.

The synthesis program reported herein can be used to solve the two variable problem by letting CBT = 0 and keeping $\delta = 0$, i.e. if CBT = 0, the damping coefficient is constant with respect to time. If $\delta = 0$, the spring system has only one spring constant, K_2.

The results were very similar considering that two independent methods were used in both the analysis and synthesis procedures. The load condition consisted of two pulses, the first 1000 in/sec^2 for 0.05 seconds and the second 2000 in/sec^2 for 0.01 seconds. The results are shown in Table 4.

4.3 Single Load Condition

It was suspected that there exists a region containing a large number of designs all having the same optimum criterion value. This belief is supported by the results of both single and multiple load condition cases. Three distinct starting points were used and three distinct terminal designs resulted. Each terminal design had the same criterion value associated with it. This phenomena has been experienced before in structural synthesis problems (See Refs. 8 and 9).

The three initial designs chosen for the single load condition case are listed in Table 1 under Case 1_s. The load condition was the pulse.

$$S(t) = \begin{cases} 1000.0 \text{ in/sec}^2 & t \leq 0.05 \text{ sec.} \\ 0 & t > 0.05 \text{ sec.} \end{cases}$$

The computer input data determining constraints and parameters for Case 1_s is listed in Tables 2 and 3. The terminal designs and the percent reduction in criterion values are listed in Table 4. The percent reduction in criterion value is calculated with respect to the criterion value obtained in Ref. 1 (See Table 4). It was felt that this value was a fair standard even though the case in Ref. 1 was a multiple load condition case. The reason is that the active load condition at the optimum design of Ref. 1 was the pulse S(t) defined above. It is seen from Tables 1 and 4, for Case 1_s, that these distinct starting points have terminated at three distinct designs all of which are characterized by approximately the same percent reduction in criterion value. An illustration of the reduction in criterion value versus the computer running time is shown in Figure 14.

4.4 Multiple Load Conditions

4.4.1 Introduction. A very salient characteristic of a multiple load condition synthesis problem is that the design which optimizes the system for any single load condition will not, in general, also be the optimum design for the other load conditions. For example, 3 distinct starting points were used to find the best possible design for a single load condition. The final criterion value was between 626 and 635 in/sec^2. See Table 4 and Figure 15. However, when these designs were subjected to the second load condition of pulse set II, the maximum absolute acceleration was found to be near 1000.0 in/sec^2. Thus, if the second pulse is included the value of the criterion function at this design is 1000 in/sec^2 rather than 636 in/sec^2. Adding pulses can change the form of the criterion function over major portions of the design space. If the additional pulses change the form of the criterion function in the region of the optimum design obtained ignoring the additional pulses, then the previous results are invalid.

With this in mind, it is easily seen why all load conditions must be observed as the synthesis progresses. It is also important that the constraints for all the load conditions be satisfied in order that the design be acceptable. Therefore, the problem consists of choosing the direction which best minimizes the criterion function of all load conditions and will not violate any of the constraints for all the load conditions. This means it is possible to have load condition 1 possessing the maximum absolute acceleration and load condition 2 possessing a maximum relative deflection which puts the design on the deflection constraint.

4.4.2 Finding an Advantageous Starting Design for the Multiple Load Condition Problem. To find a good starting point with minimum expenditure of computer running time the computer program used one load condition which was thought to be more critical than the others. This particular load condition was used for ten minutes of running time. Ten minutes was used because it was found that the criterion value decreased slowly after this time as shown in Figure 14. During this time all constraints of the multiple load condition problem were satisfied. After 10 minutes an acceptable design resulted and was used as the starting point for the multiple load condition problem. The method described above was found to improve efficiency with respect to computer running time.

4.4.3 Results for Multiple Load Conditions. The results obtained from two synthesis problems previously worked in Ref. 1 confirm the statement, that better or equal designs with respect to criterion values can be obtained by increasing the number of design variables. The two multiple load condition cases were Case 2_m for

$$\text{Pulse set I} = \begin{cases} S_{11}(t) = 2000.0 \text{ in/sec}^2 & 0.0 \le t \le 0.001 \text{ sec.} \\ S_{21}(t) = 200.0 \text{ in/sec}^2 & 0.0 \le t \le 0.01 \text{ sec.} \\ S_{31}(t) = 2000.0 \text{ in/sec}^2 & 0.0 \le t \le 0.01 \text{ sec.} \end{cases}$$

and case 1_m for

$$\text{Pulse set II} = \begin{cases} S_{12}(t) = 1000.0 \text{ in/sec}^2 & 0.0 \le t \le 0.05 \text{ sec.} \\ S_{22}(t) = 2000.0 \text{ in/sec}^2 & 0.0 \le t \le 0.05 \text{ sec.} \end{cases}$$

A summary of starting designs, input data specifying side and behavior constraints and terminal designs for the more sophisticated shock isolator reported herein and the shock isolator of Ref. 1 can be found in Tables 1 thru 4.

A comparison of the terminal designs and criterion values for Cases 1_m and 1_s reveal the effect of adding another load condition to a single load condition synthesis problem.

4.5 Displacement Results

The results of treating the maximum absolute acceleration as a behavior constraint and the minimum maximum relative displacement as the criterion function did not yield significant improvement with respect to the percent reduction of criterion value for the one case available in Ref. 1. Two initial starting points were used. The same terminal design was obtained for each case. A summary of the initial design, constraints and terminal designs for Ref. 1 and the increased design variable cases are given in Tables 5 and 6.

A two dimensional graph of the design space near the terminal design for the displacement problem is shown in Figure 16. It is felt that the upper bound constraint on the coefficient of damping was placed at a value too low to allow the damping to become anything but a constant. As the computer program progressed, all the damping variables increased. This is logical because a stiffer system will produce a smaller maximum deflection. One by one the damping variables reached the upper bound. Since increasing damping was not allowed, the modified gradient direction focused all attention on the spring system as the main design variables. The final design resembles that of Ref. 1, except for the gap and K_2 as is seen from Figure 17. If the spring system consisted of only one spring constant and no gap variable, the resulting design would have been identical to that of Ref. 1.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATIONS

The results have revealed that more desirable shock isolator performances can be obtained by allowing the coefficient of damping to be a piecewise linear continuous function of time and replacing the single spring with a bilinear spring system. The percent reduction in criterion value with respect to the results of Ref. 1 was chosen to be indicative of the degree to which a shock isolator performance was judged more desirable. A summary of percent reduction in criterion values and final designs appears in Table 4. Associated terminal designs are depicted in Figures 18 thru 21. The percent reduction in criterion values for the Case 2_m were not as significant as those for Case 1_m. It was felt that this was due to the shorter time duration in the load pulses of Case 2_m.

The synthesis method employed consisted of three types of moves in the design space: (1) moving in the negative gradient direction if no constraints were active, (2) deciding whether to remain on active constraint or to move off it, and (3) when remaining on an active constraint finding the direction of travel containing the largest component of the negative gradient.

The results showed distinct terminal designs with the same criterion value. It is felt that this phenomena, which has been observed in previous synthesis problems (See Refs. 8 and 9), can be attributed to a common characteristic of the terminal design, such as perhaps energy absorbtion.

There are several things that have been investigated in this study that represent advances beyond the two design variable system reported in Ref. 1. Because of the numerical integration technique employed in the analysis, the program has the capability of working with any type of pulse. The pulses do not have to be definable in terms of a function. A series of points from a recorder could be used.

All of the developments in the modified gradient direction can be applied to a general N dimensional design space. The improved shock isolator synthesis program can be specialized to take the form of the two design variable case by letting the rate of change of damping with respect to time be zero and letting $\delta = 0$.

The method of using normals to the constraints to keep from entering the unacceptable region lends itself quite easily to linear constraints. A linear approximation to the nonlinear behavior constraint proved successful. A situation where normals to active constraints cannot be employed successfully arises when an unreasonable amount of computer time is required to calculate them.

This capability could be extended further by adding more variables such as the time between damping positions (See Fig. 3). The number of damping variables, spring constants and gaps could also be increased. Further extensions could include applications of variable damping and bilinear spring systems to problems with more than one degree of freedom.

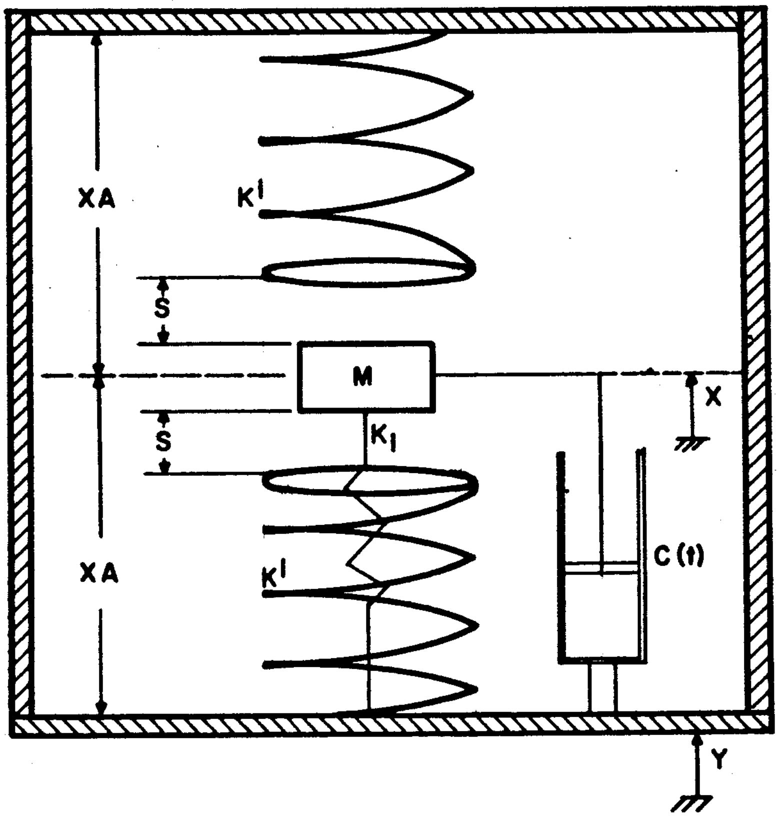


FIGURE I. SHOCK ISOLATOR

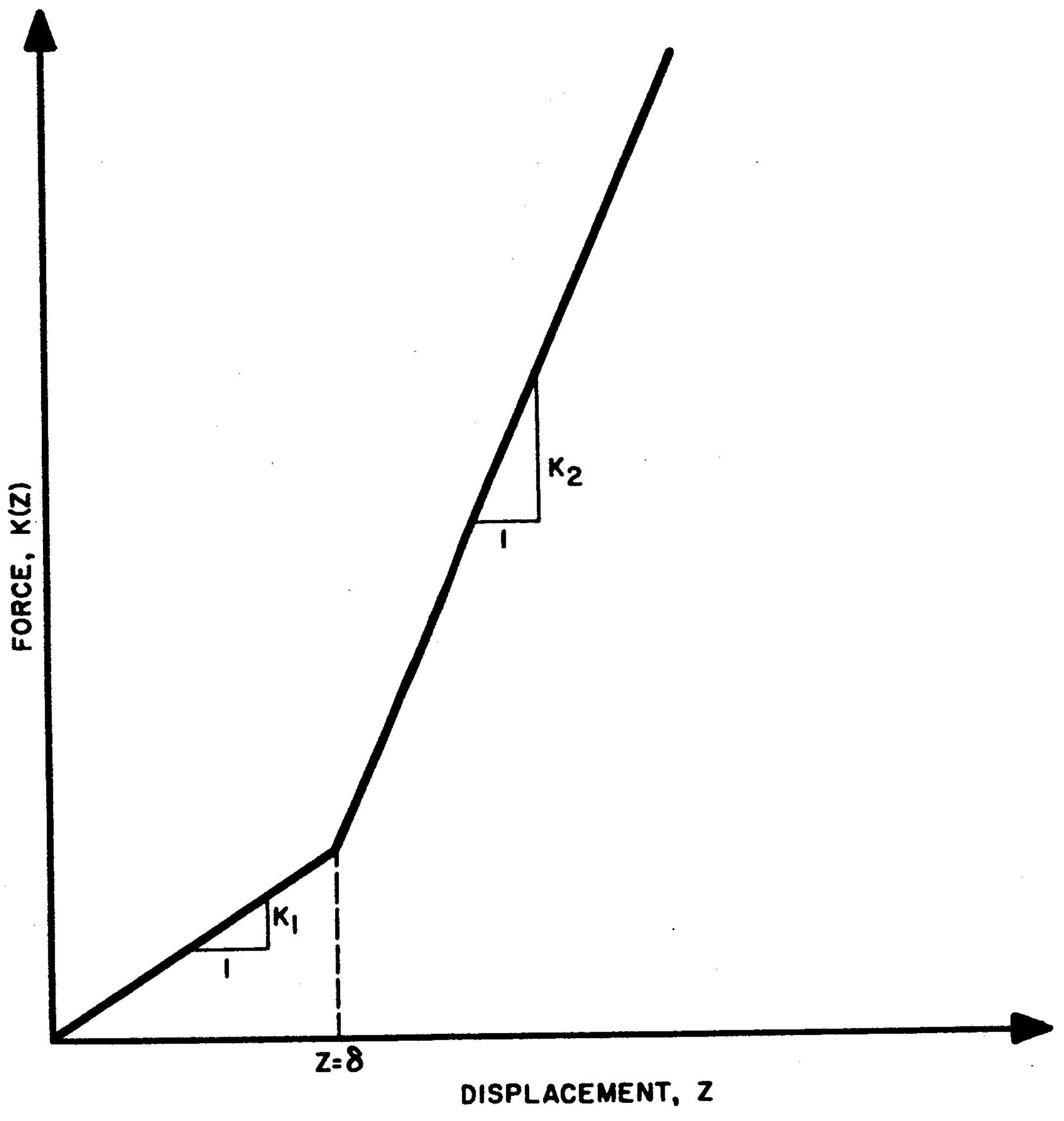


FIGURE 2. FORCE VERSUS DISPLACEMENT FOR BI-LINEAR SPRING

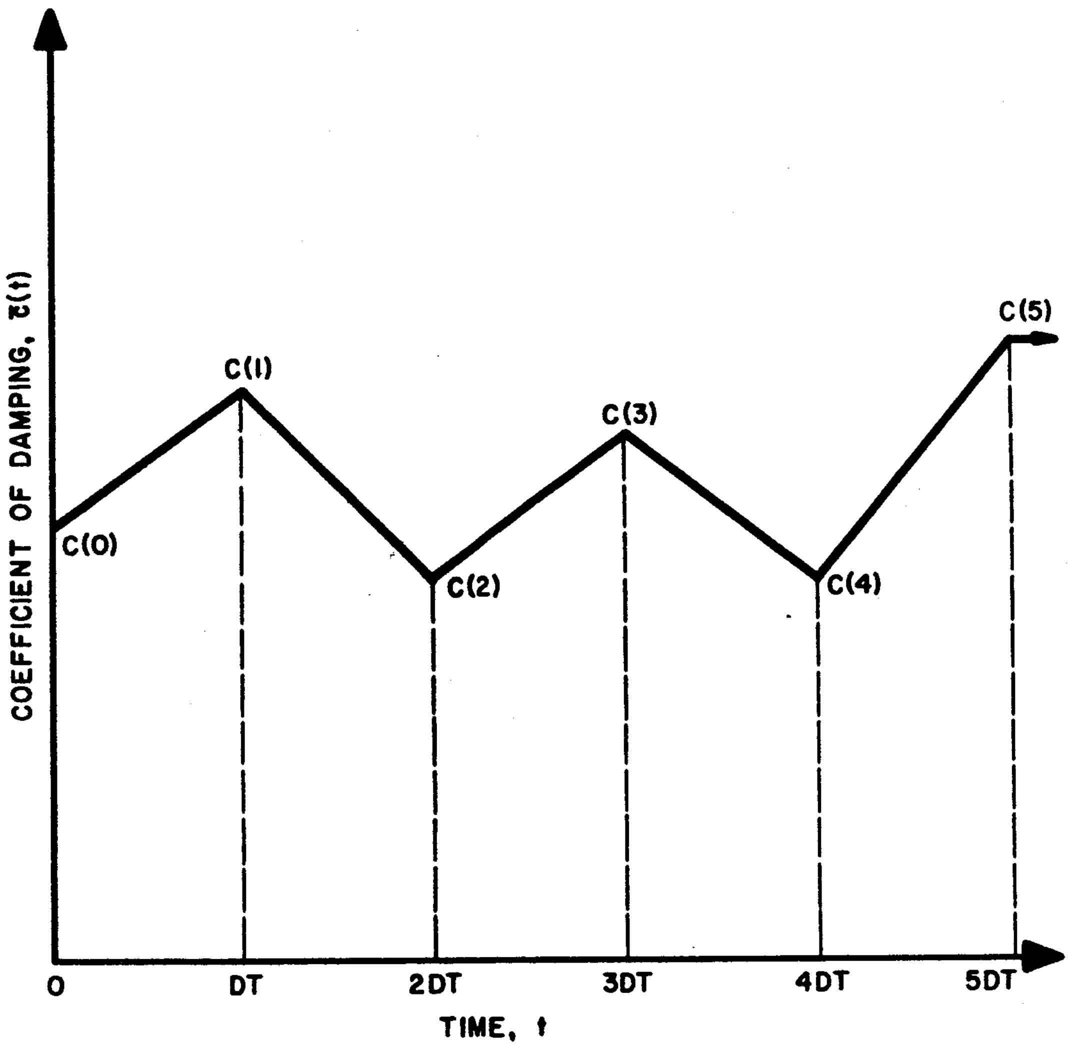


FIGURE 3. TIME DEPENDENT, PIECE WISE LINEAR, CONTINUOUS DAMPING

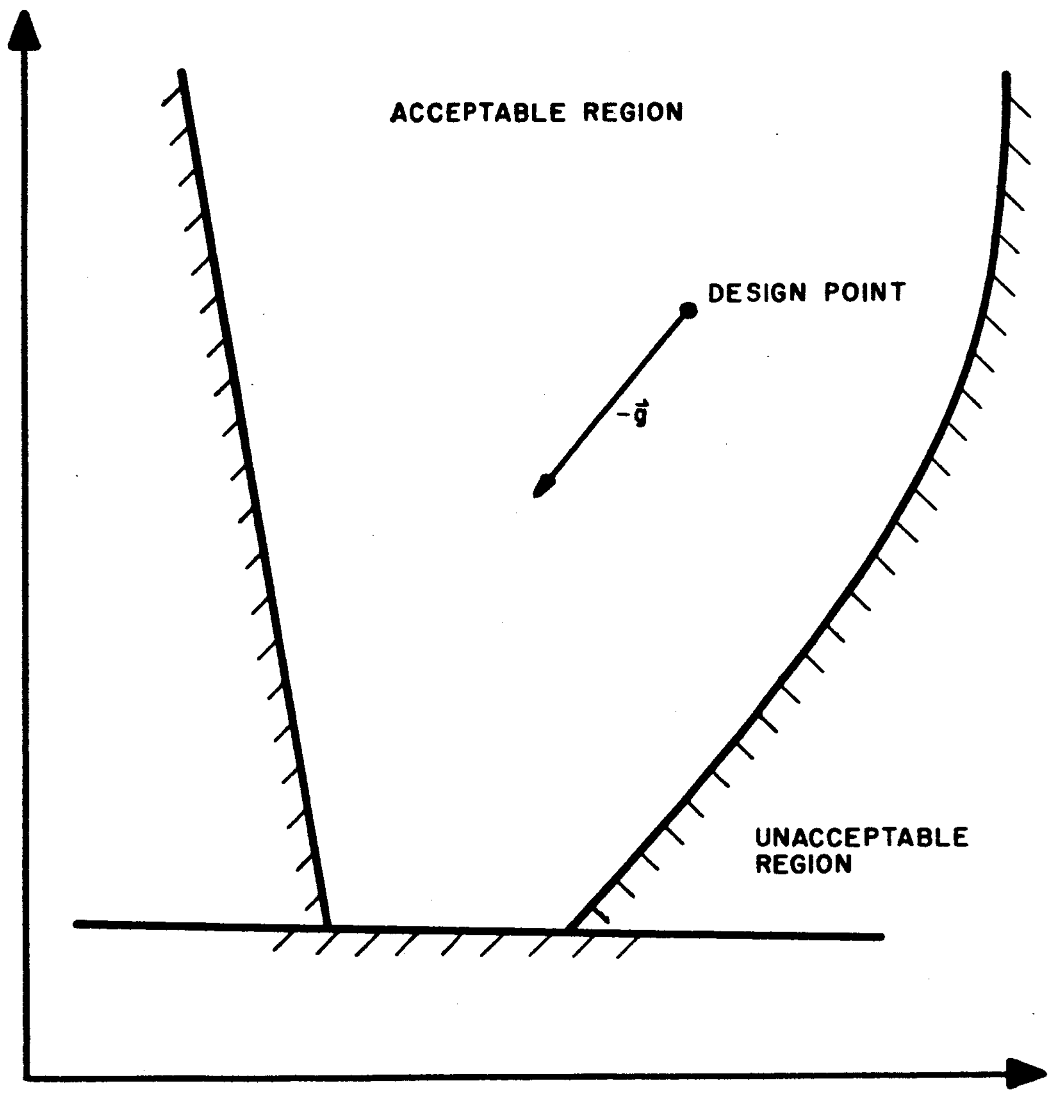


FIGURE 4. FREE DESIGN POINT

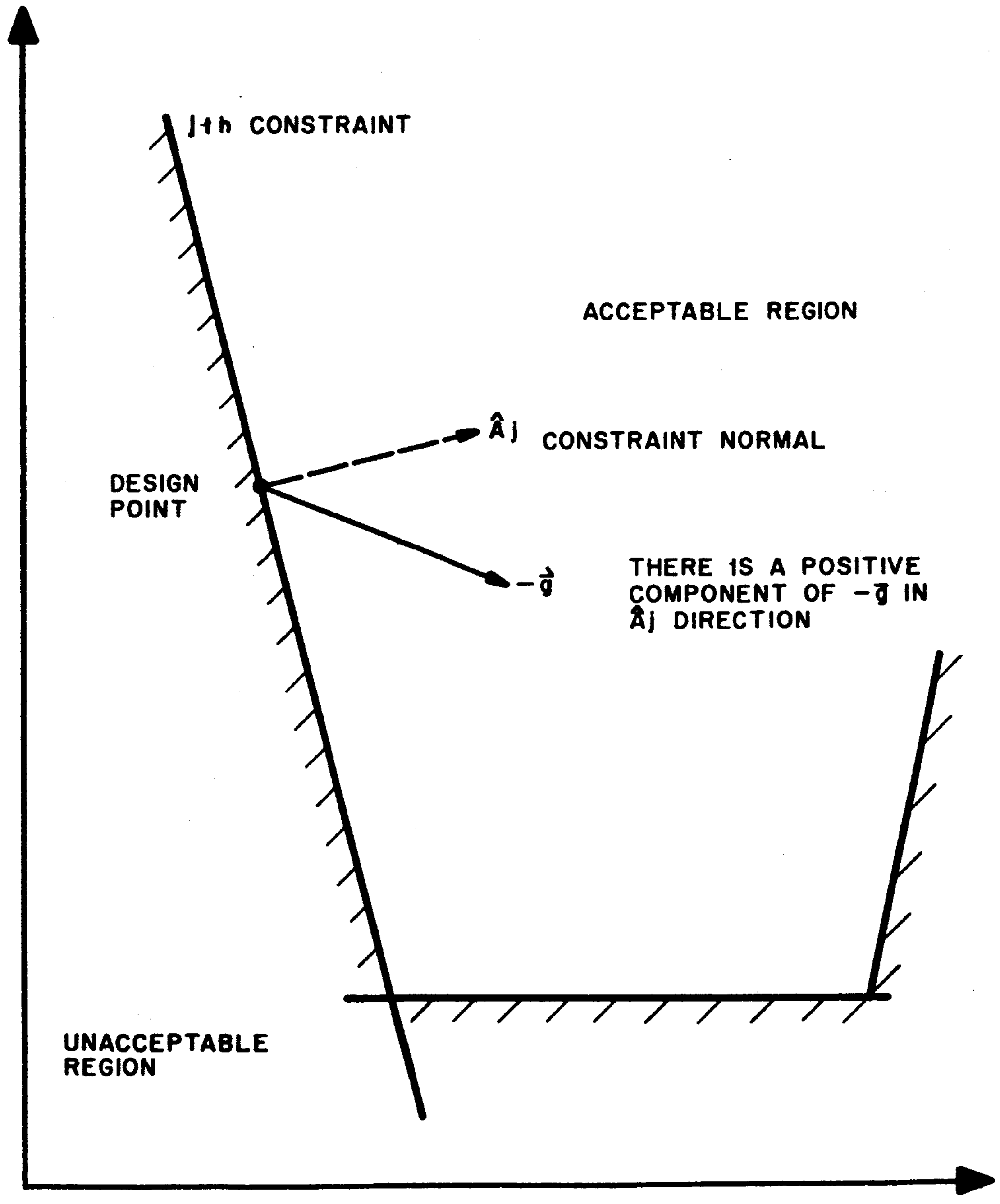

j-th CONSTRAINT
ACCEPTABLE REGION
$\hat{A}_j$ CONSTRAINT NORMAL
DESIGN POINT
$-\vec{g}$
THERE IS A POSITIVE COMPONENT OF $-\vec{g}$ IN $\hat{A}_j$ DIRECTION
UNACCEPTABLE REGION

FIGURE 5. CONSTRAINED DESIGN POINT

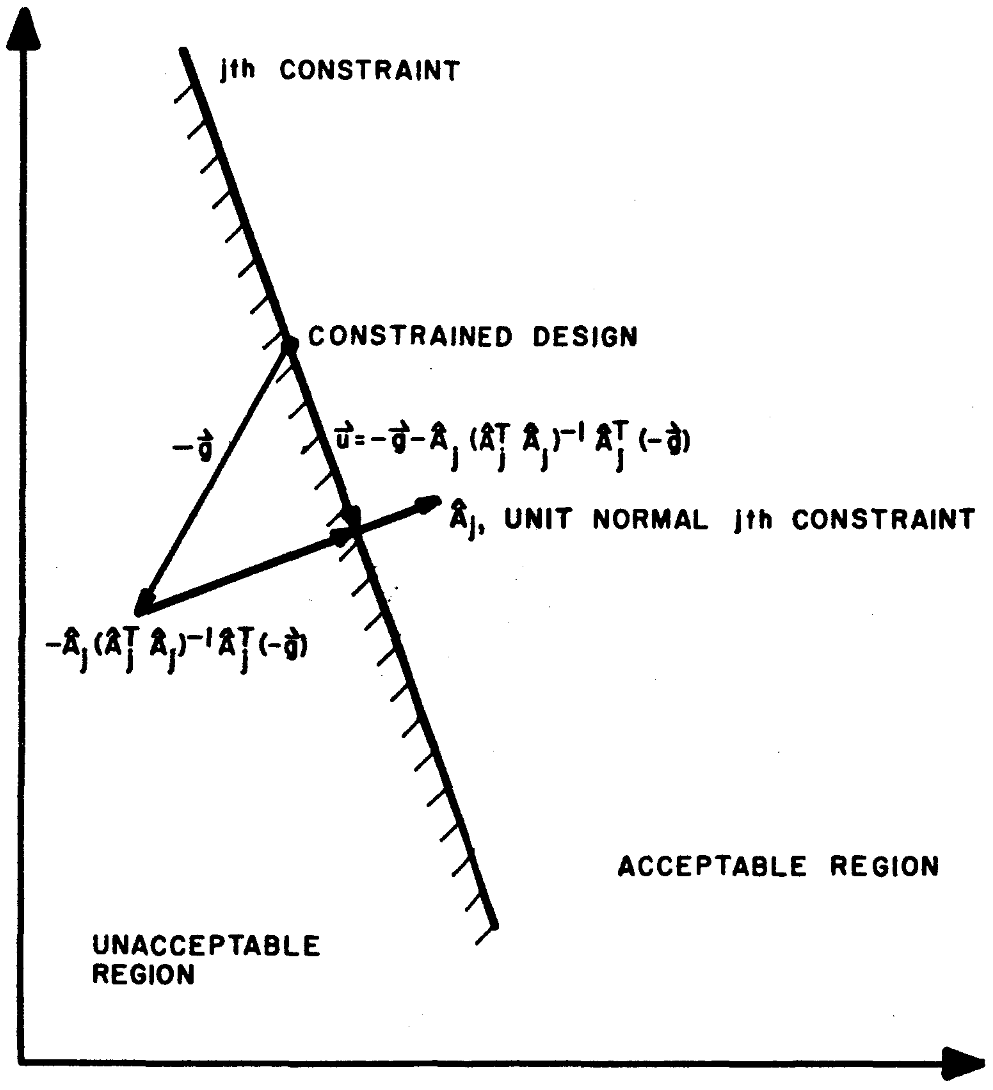

jth CONSTRAINT
CONSTRAINED DESIGN
$-\vec{g}$
$\vec{u}=-\vec{g}-\hat{A}_j(\hat{A}_j^T\hat{A}_j)^{-1}\hat{A}_j^T(-\vec{g})$
$\hat{A}_j$, UNIT NORMAL jth CONSTRAINT
$-\hat{A}_j(\hat{A}_j^T\hat{A}_j)^{-1}\hat{A}_j^T(-\vec{g})$
ACCEPTABLE REGION
UNACCEPTABLE REGION

FIGURE 6. DIRECTION OF CONSTRAINED STEEPEST DESCENT

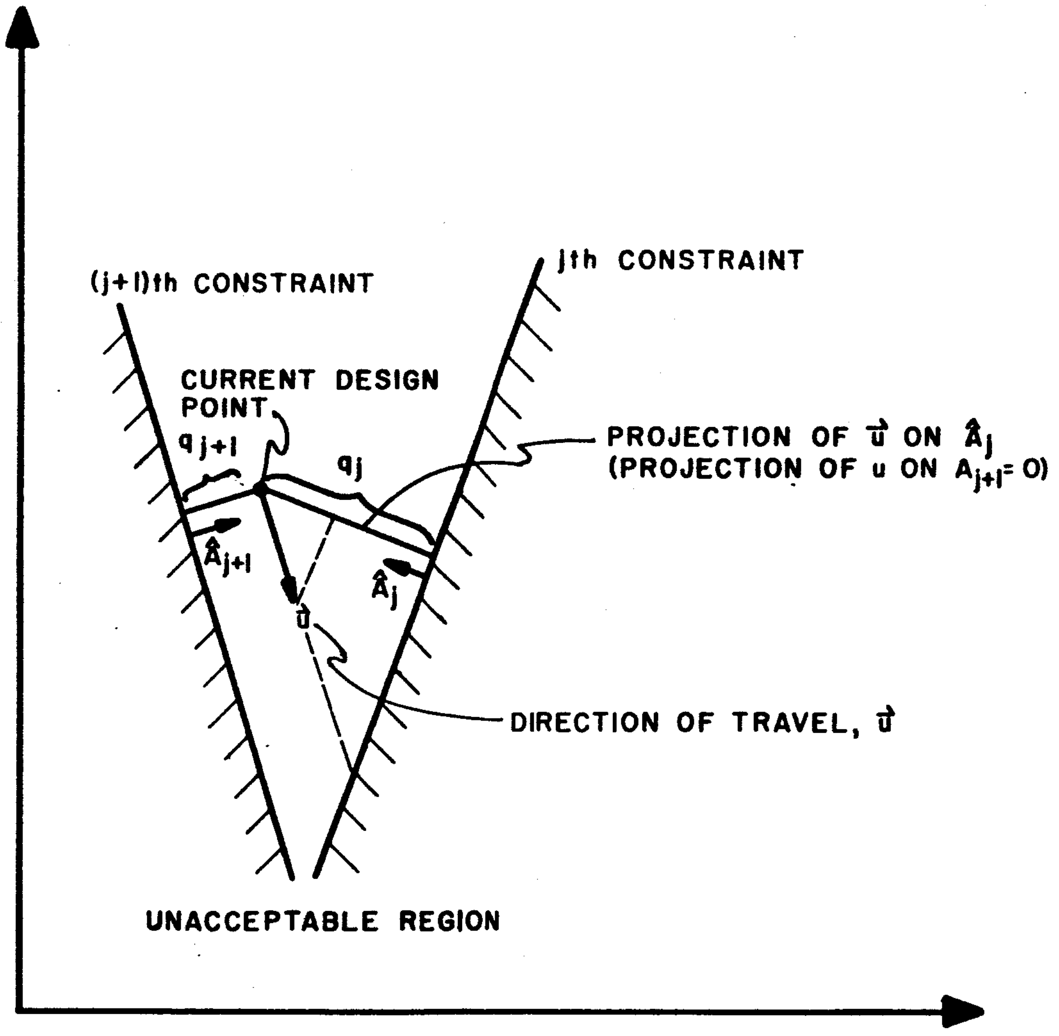


FIGURE 7. DETERMINING HOW FAR TO GO BEFORE CONSTRAINT IS ENCOUNTERED

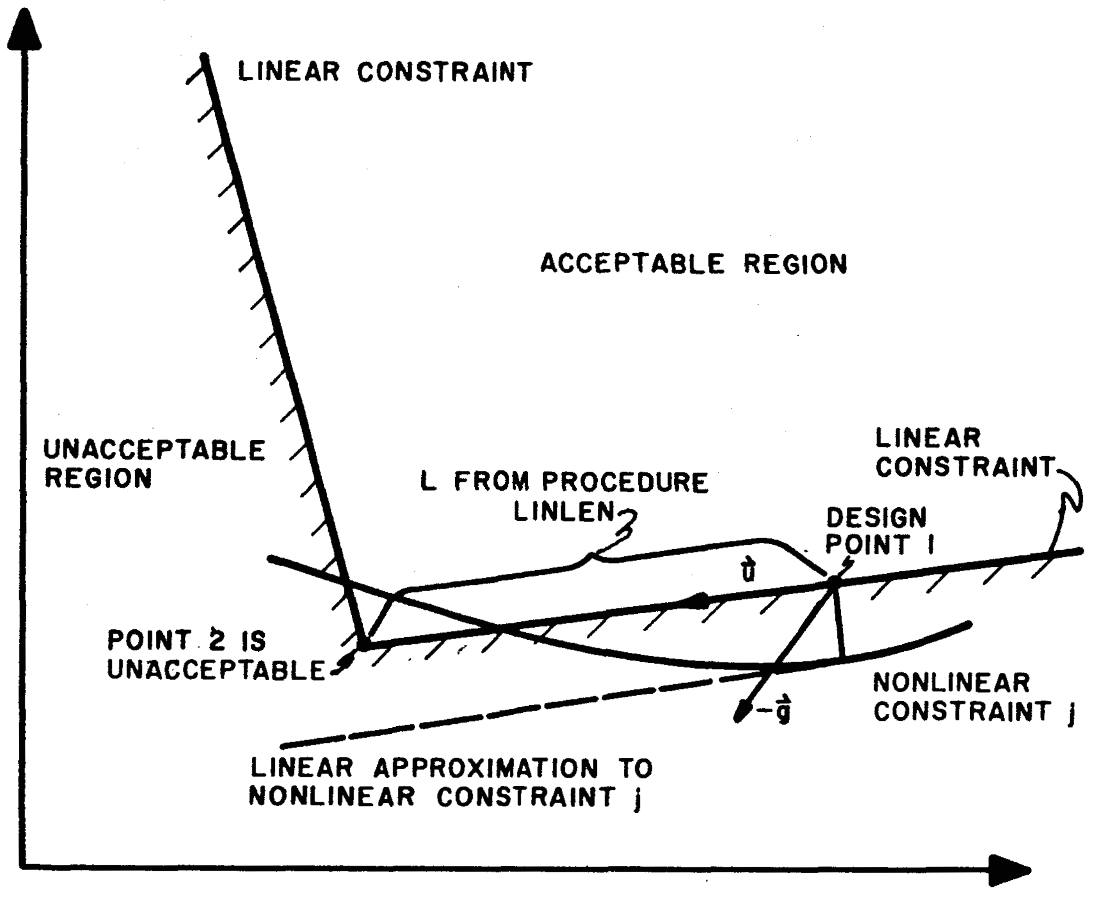


FIGURE 8. L FOUND BY LINEAR APPROXIMATION TO NONLINEAR CONSTRAINT

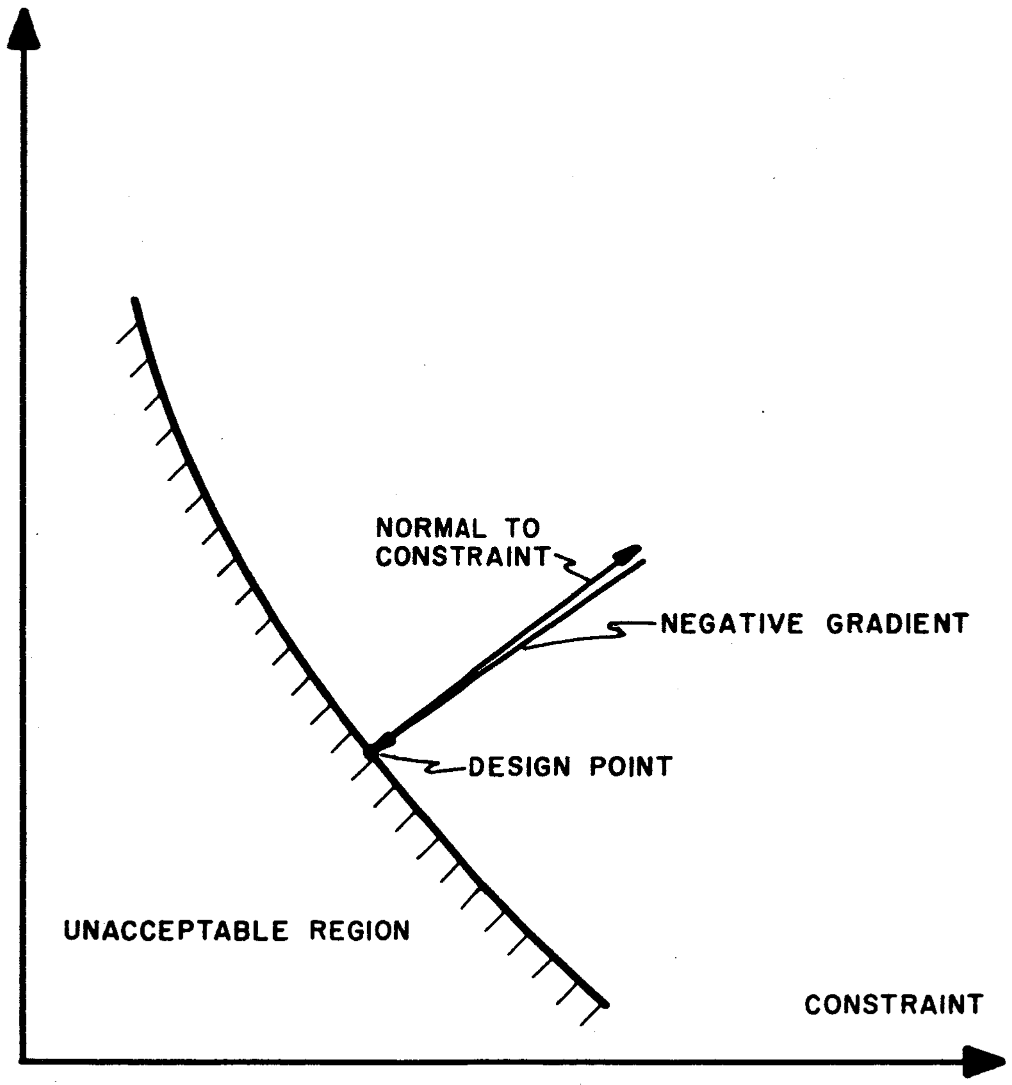


FIGURE 9. NEGATIVE GRADIENT AND NORMAL TO ACTIVE CONSTRAINT WITH INNER PRODUCT LESS THAN -0.999

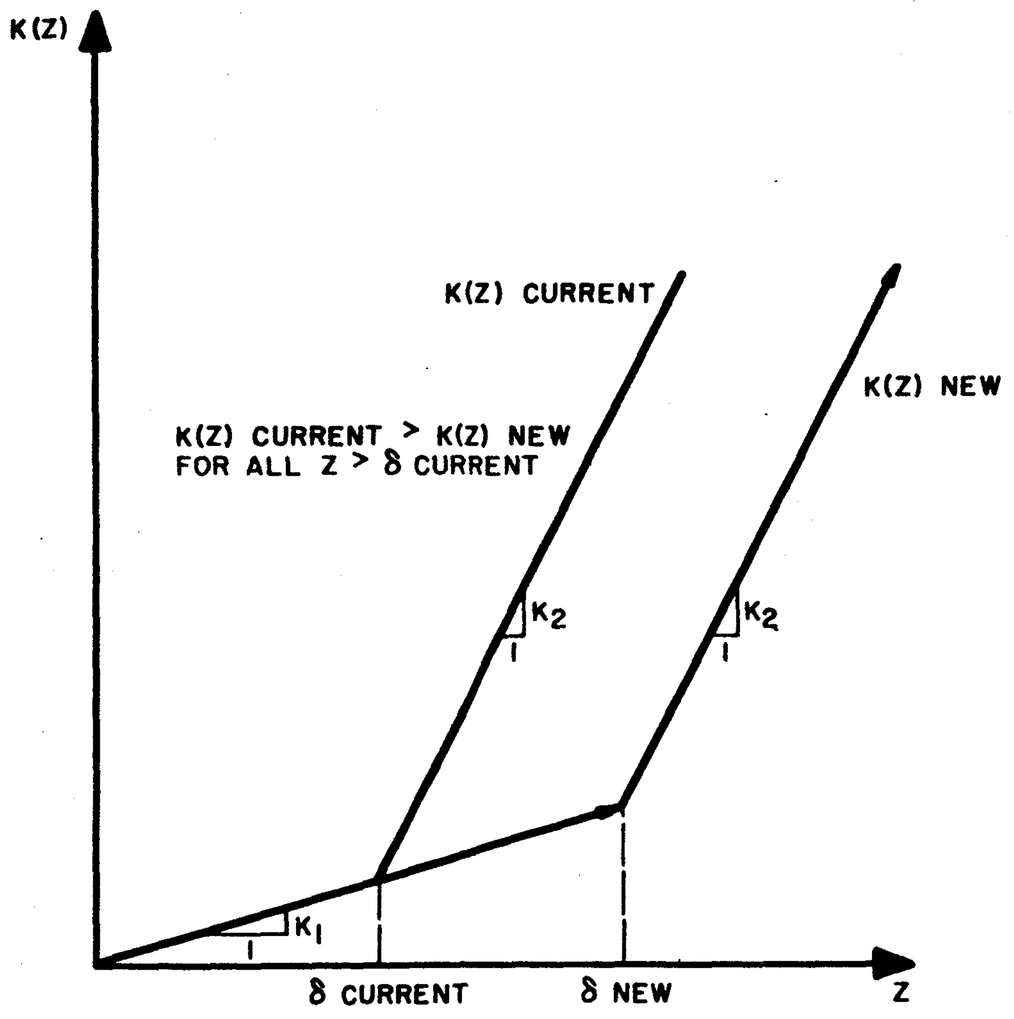


FIGURE 10. EFFECT ON STIFFNESS OF SPRING SYSTEM WHEN S IS INCREASED

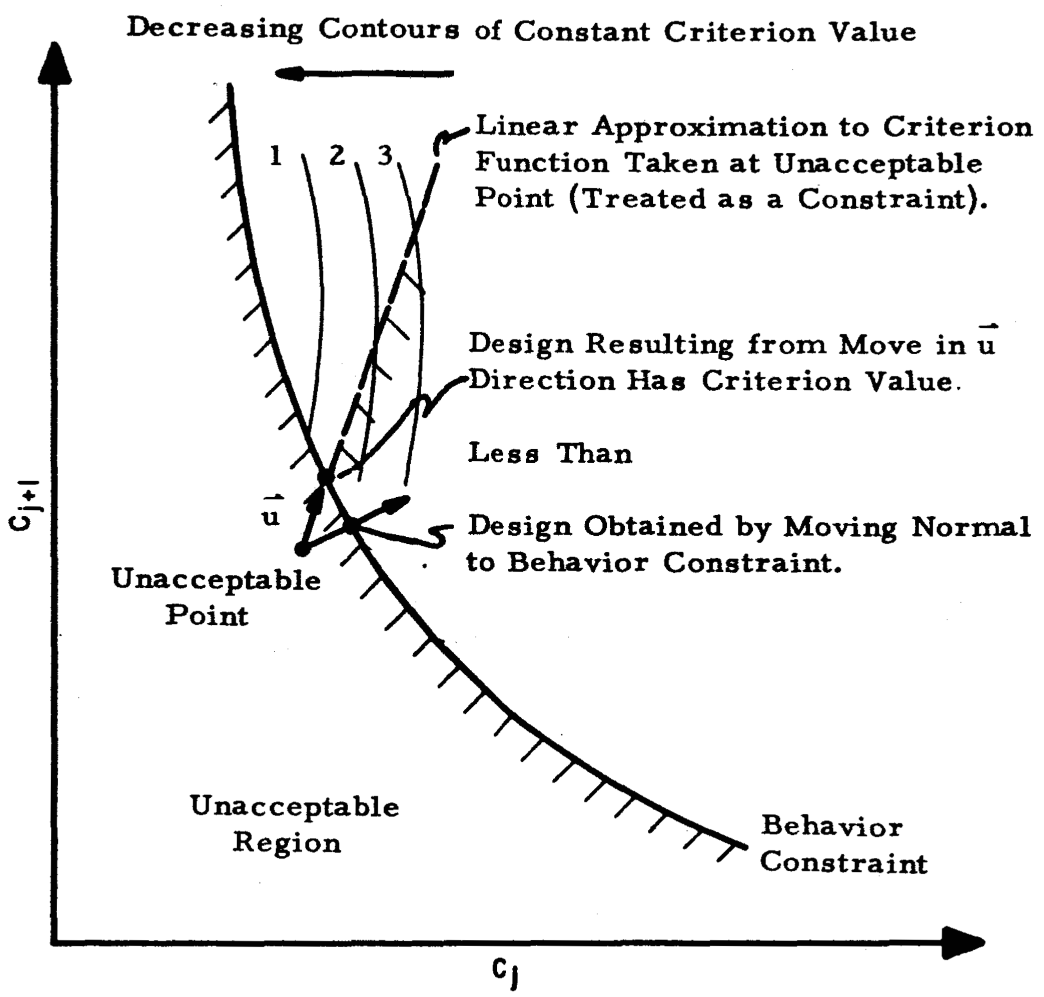

Decreasing Contours of Constant Criterion Value
1
2
3
Linear Approximation to Criterion Function Taken at Unacceptable Point (Treated as a Constraint).
Design Resulting from Move in $\vec{u}$ Direction Has Criterion Value.
Less Than
$\vec{u}$
Design Obtained by Moving Normal to Behavior Constraint.
Unacceptable Point
Unacceptable Region
Behavior Constraint
c_{j+1}
c_j

FIGURE II. ENTERING ACCEPTABLE REGION WHEN BEHAVIOR CONSTRAINT IS VIOLATED

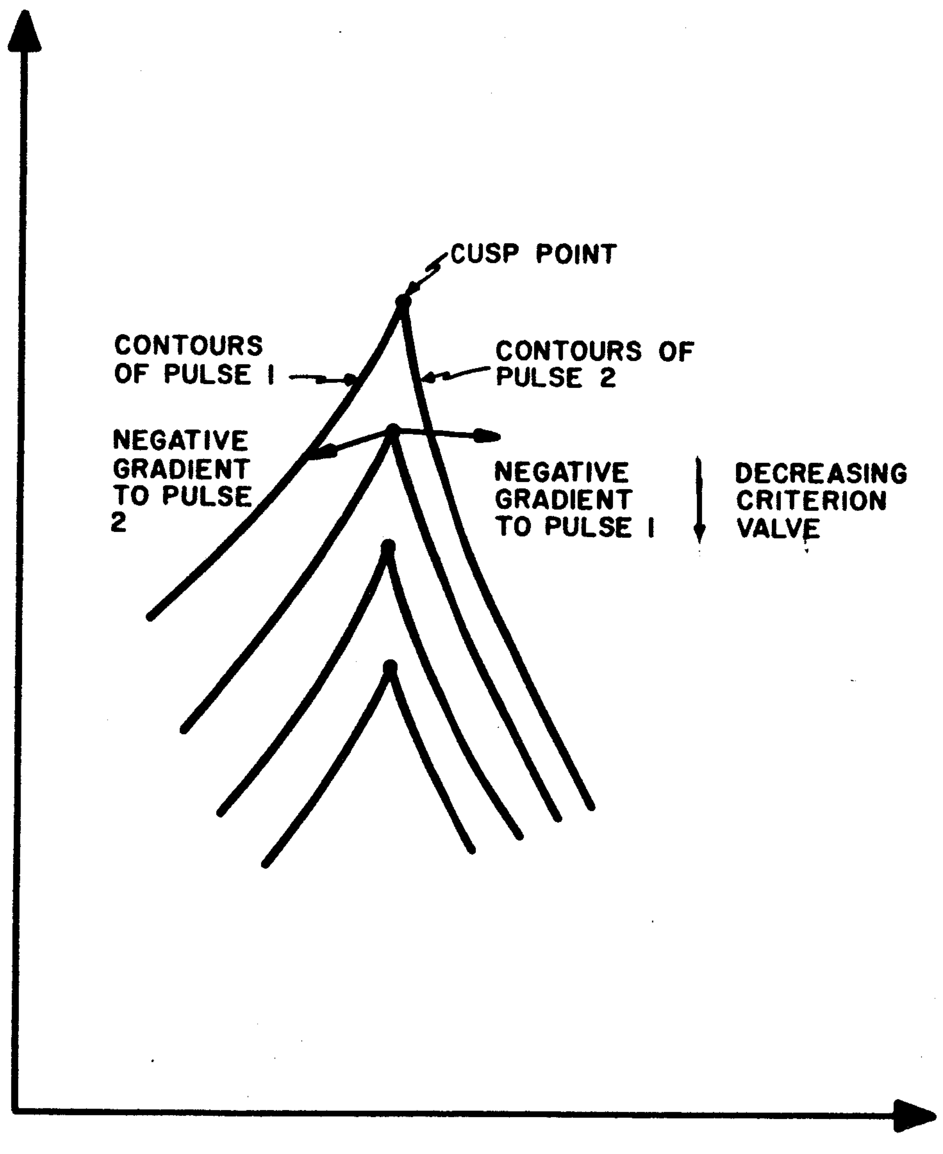

CUSP POINT
CONTOURS OF PULSE 1
CONTOURS OF PULSE 2
NEGATIVE GRADIENT TO PULSE 2
NEGATIVE GRADIENT TO PULSE 1
DECREASING CRITERION VALVE

FIGURE 12. TWO DIMENSIONAL REPRESENTATION OF CUSPS IN CRITERION CONTOURS

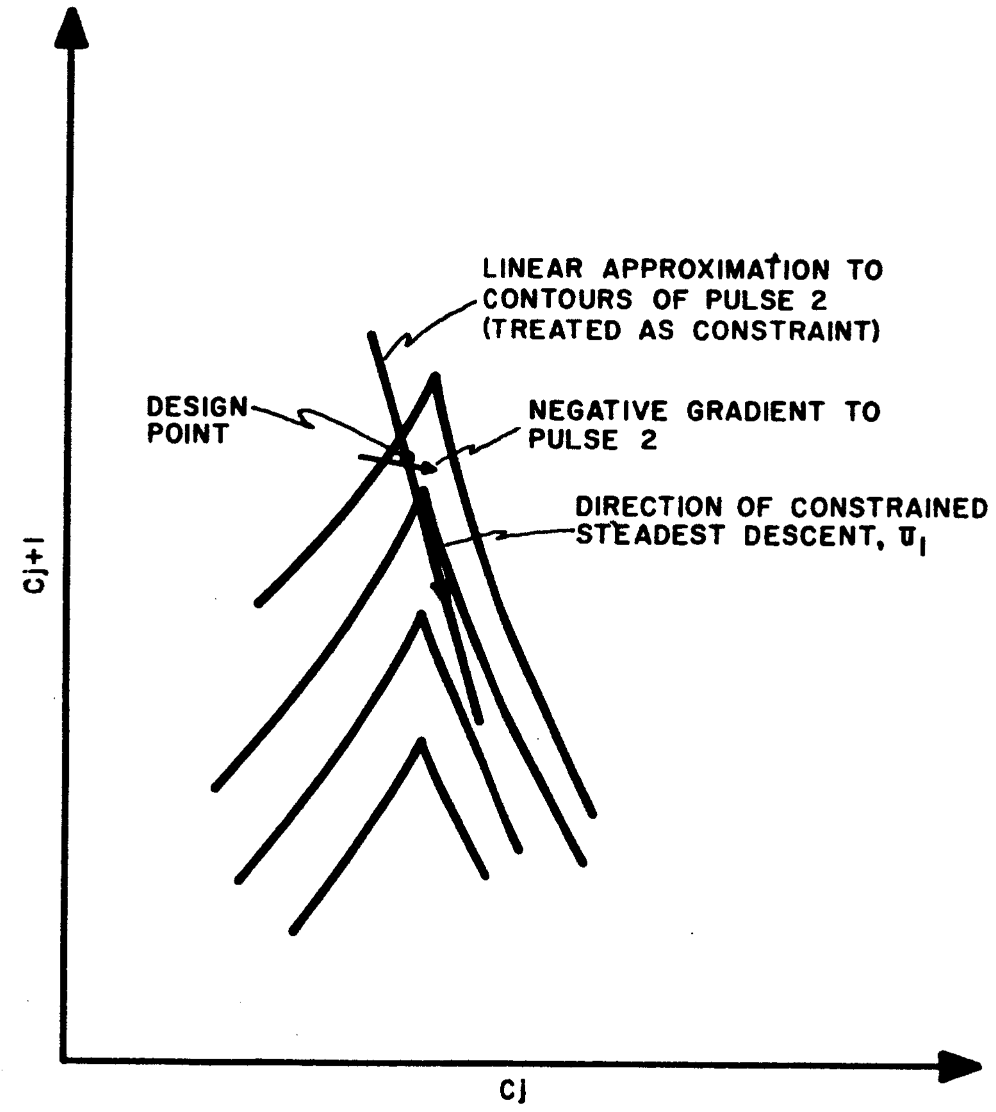
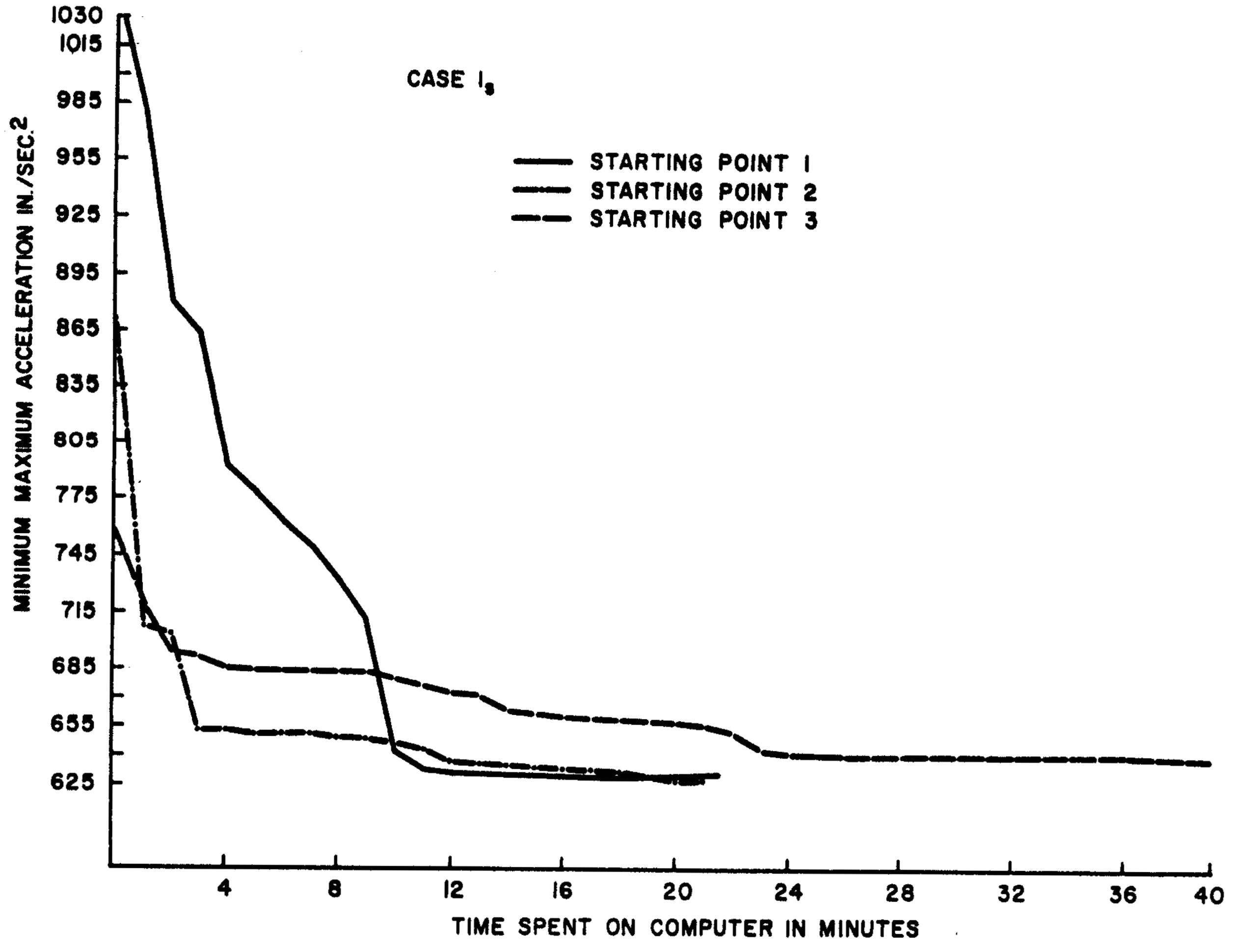

LINEAR APPROXIMATION TO CONTOURS OF PULSE 2 (TREATED AS CONSTRAINT)
DESIGN POINT
NEGATIVE GRADIENT TO PULSE 2
DIRECTION OF CONSTRAINED STEADEST DESCENT, $\overline{U}_1$
C_{j+1}
C_j

FIGURE 13. METHOD OF TRAVEL IN CUSPED REGION



CASE I_s
STARTING POINT 1
STARTING POINT 2
STARTING POINT 3
MINIMUM MAXIMUM ACCELERATION IN./SEC.^{2}
1030
1015
985
955
925
895
865
835
805
775
745
715
685
655
625
4
8
12
16
20
24
28
32
36
40
TIME SPENT ON COMPUTER IN MINUTES

FIGURE 14.

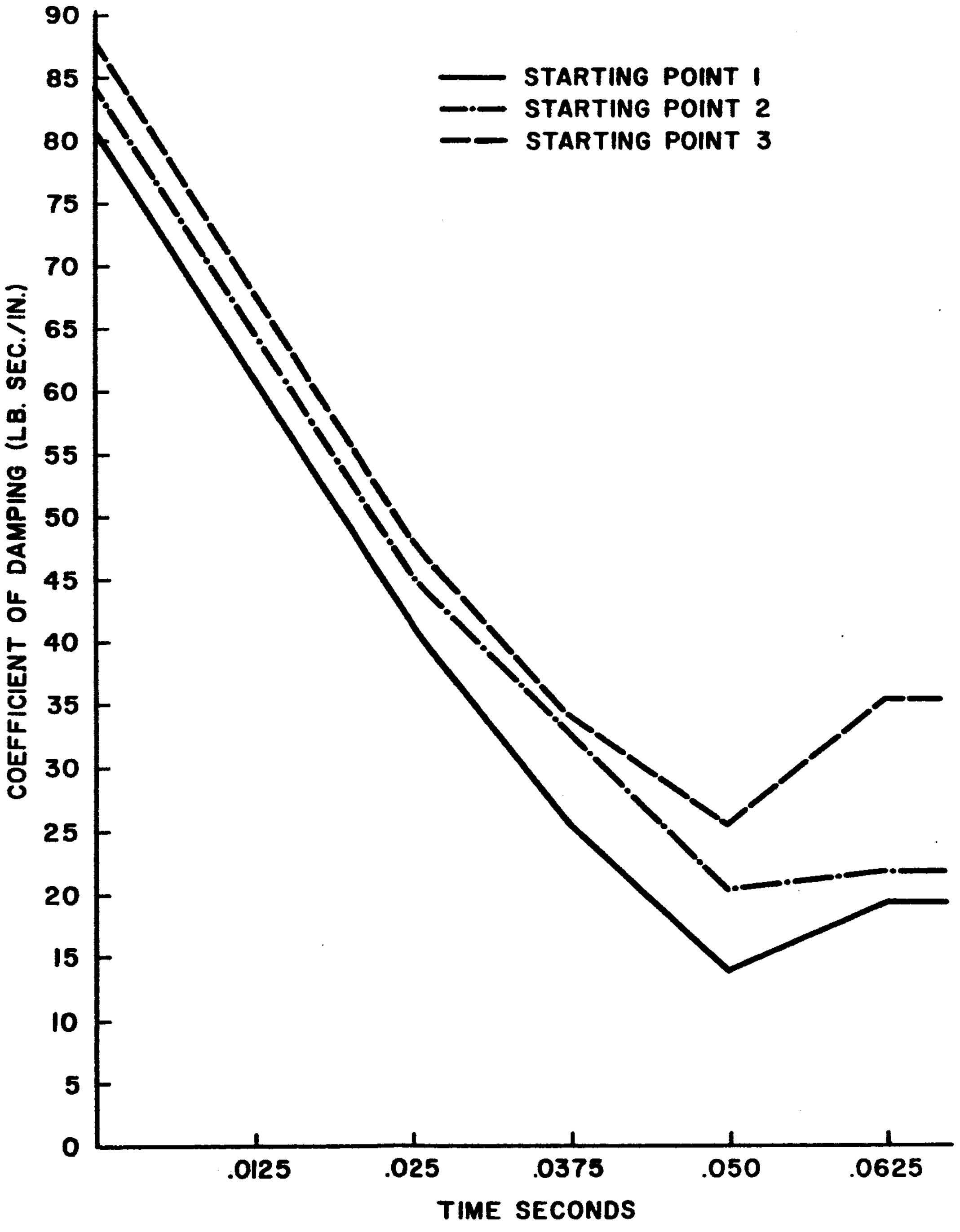


FIGURE 15. COEFFICIENT OF DAMPING VERSUS TIME FOR TERMINAL DESIGNS OF CASE I_S

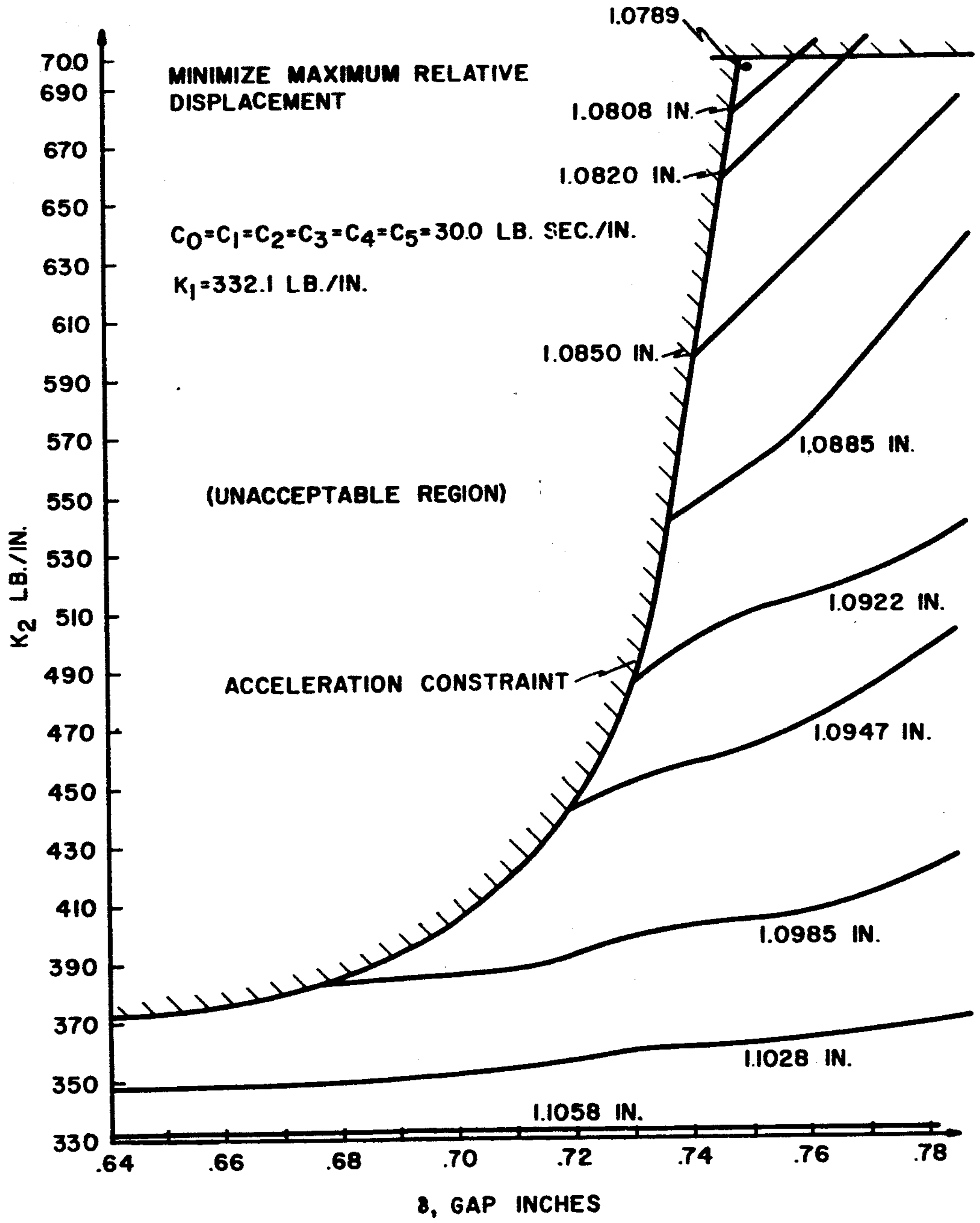
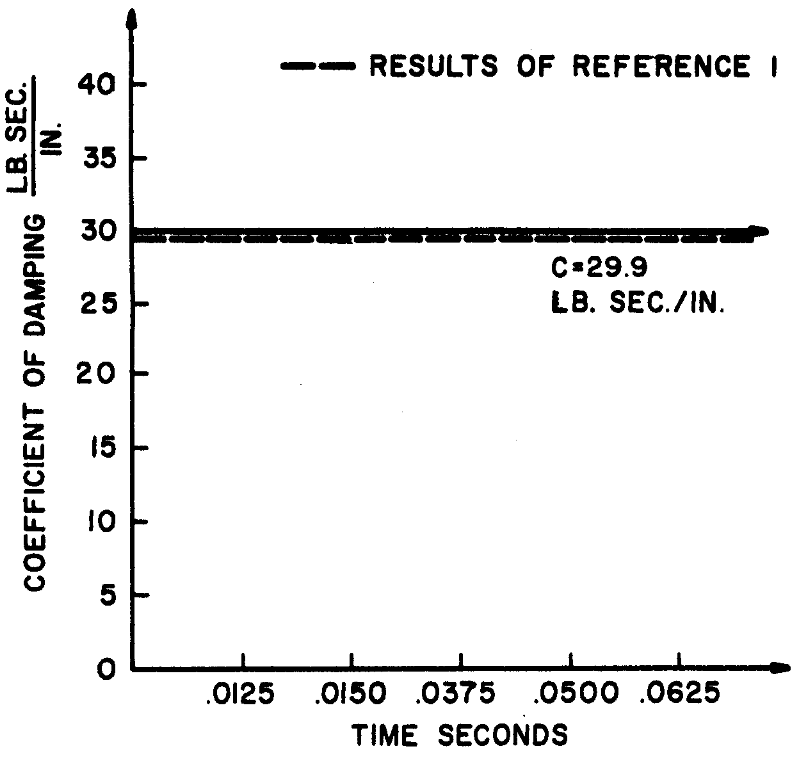


FIGURE 16. K_2-δ DESIGN-SPACE WITH REMAINING SEVEN VARIABLES FIXED



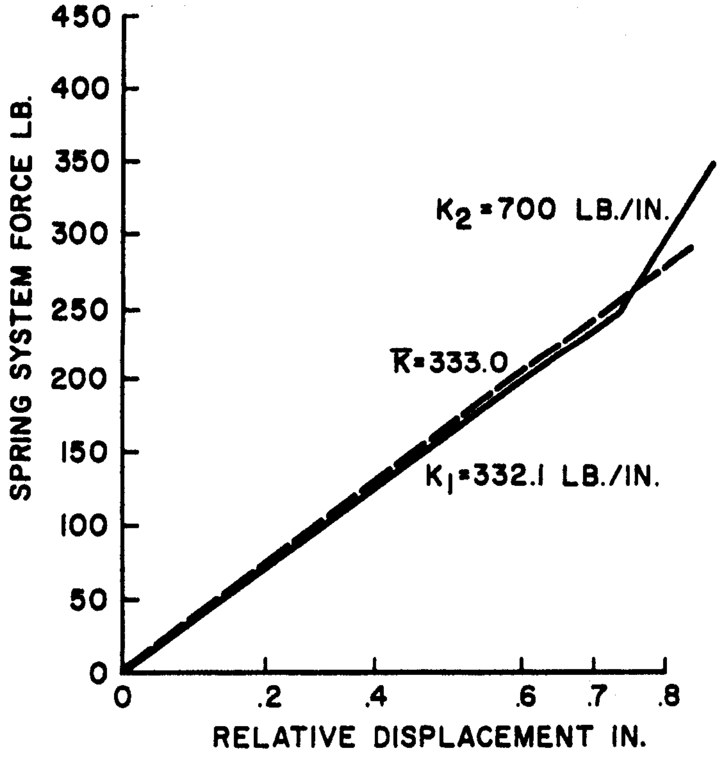
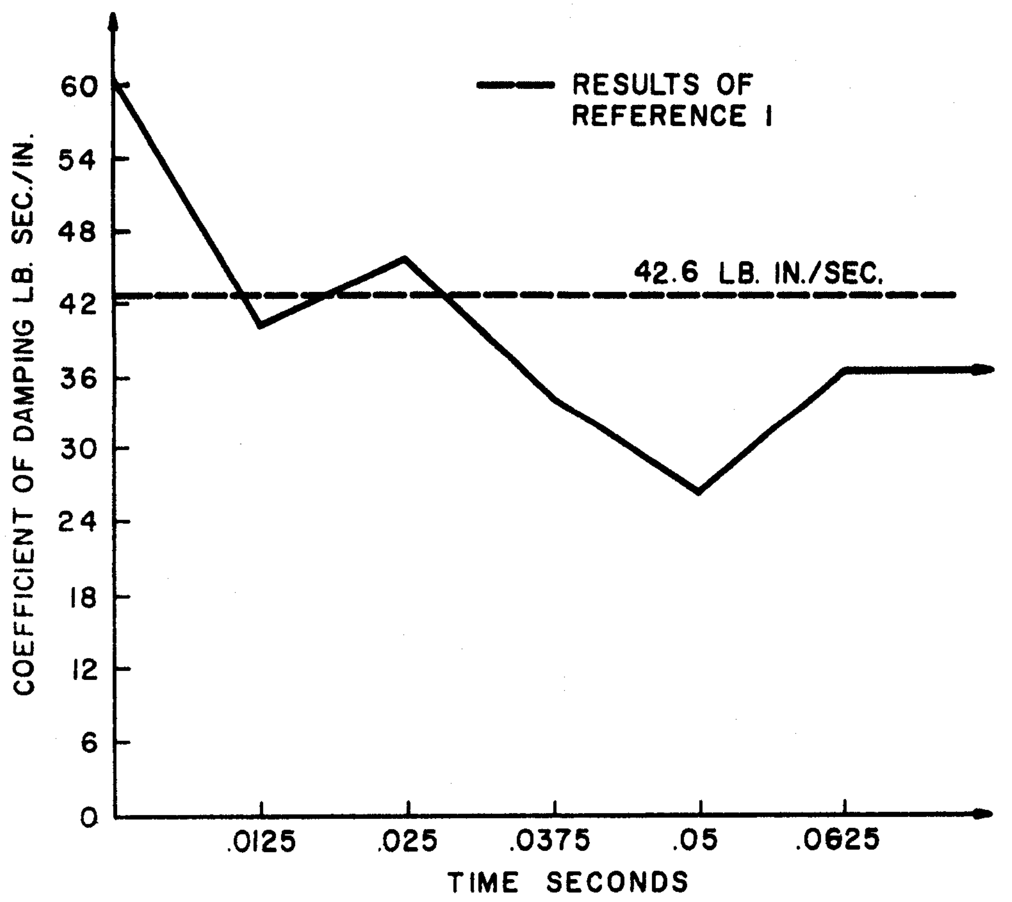


FIGURE 17. COMPARISON OF TERMINAL DESIGN FOR DISPLACEMENT CASE 3_m AND RESULTS OF REFERENCE 1



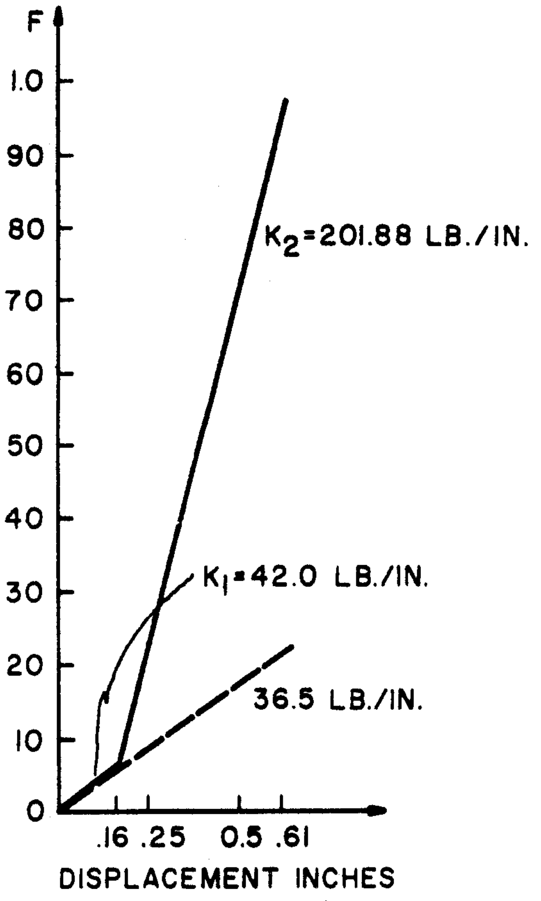


FIGURE 18. TERMINAL DESIGN FOR CASE I_m, PATH I.

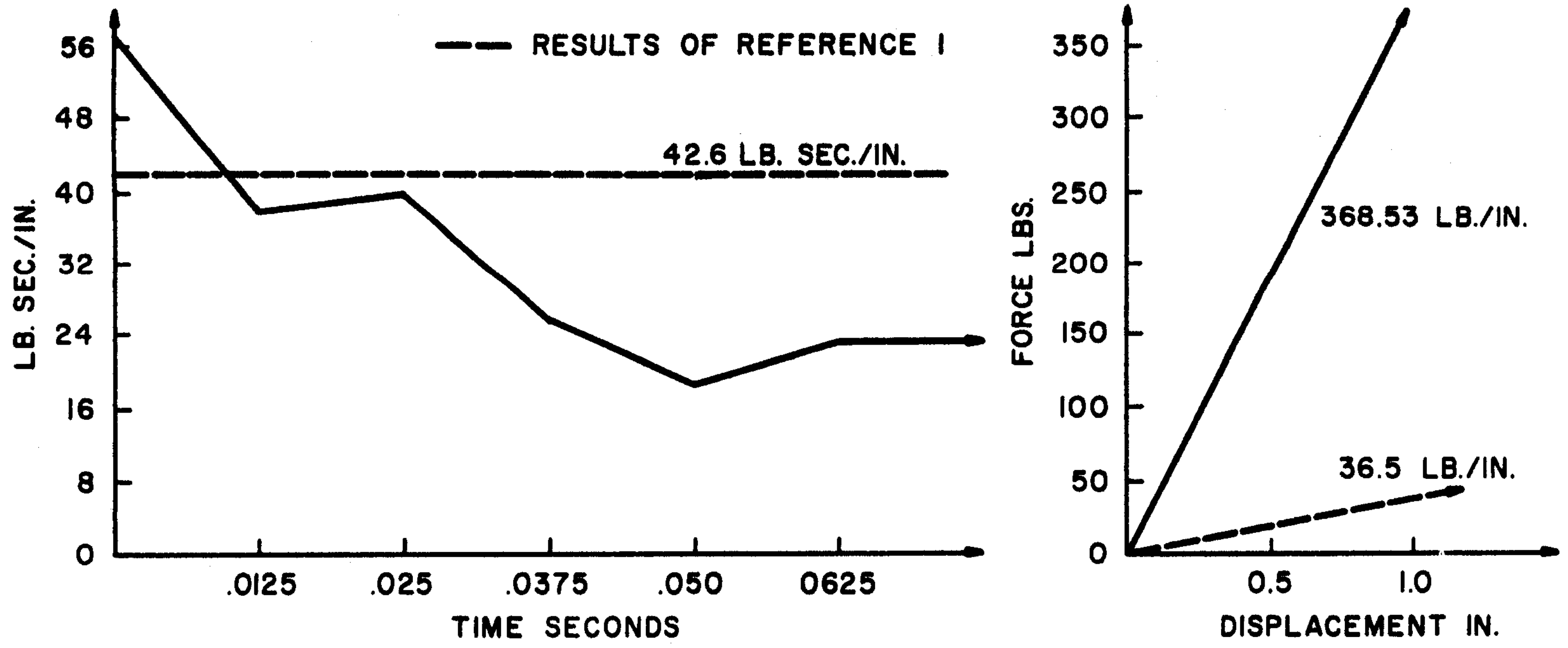


FIGURE 19. TERMINAL DESIGN FOR CASE I_m, PATH 2

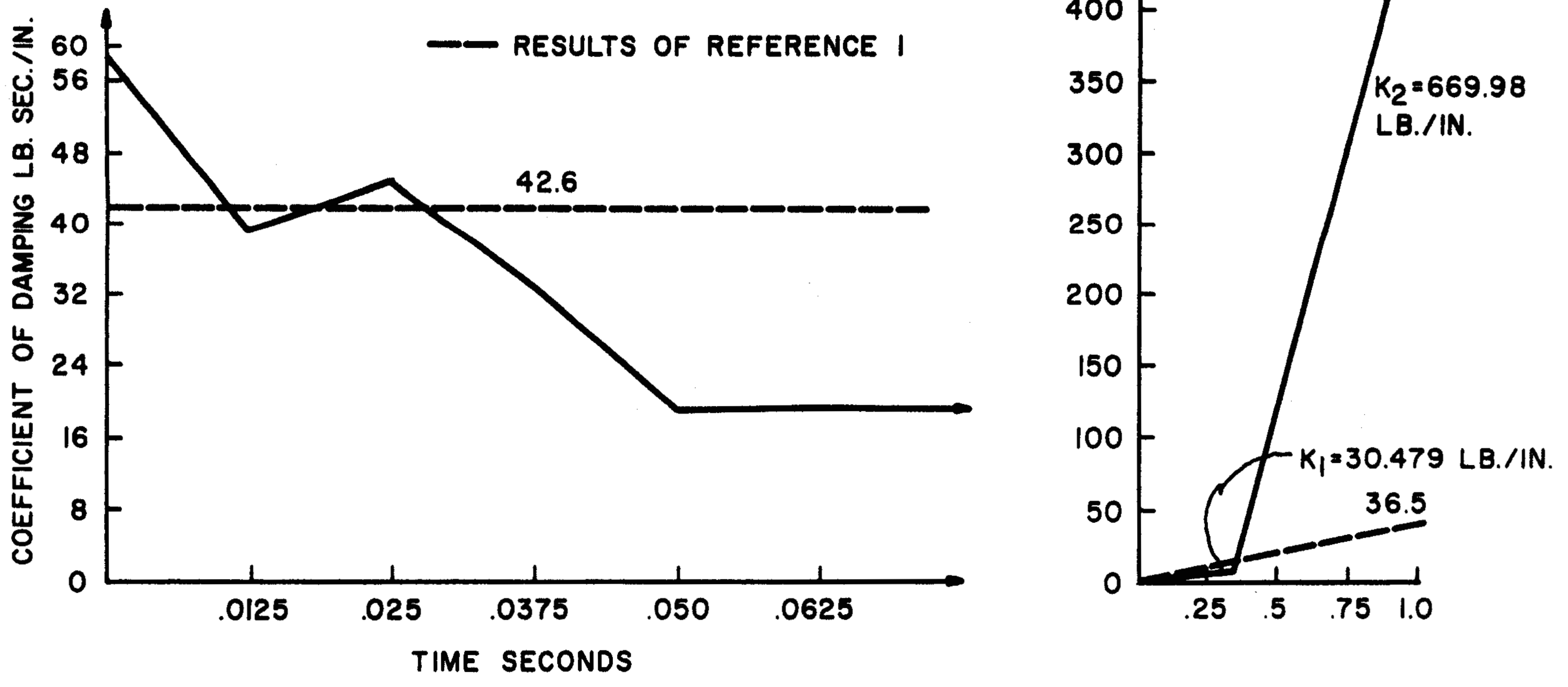


FIGURE 20. TERMINAL DESIGN FOR CASE I_m, PATH 3

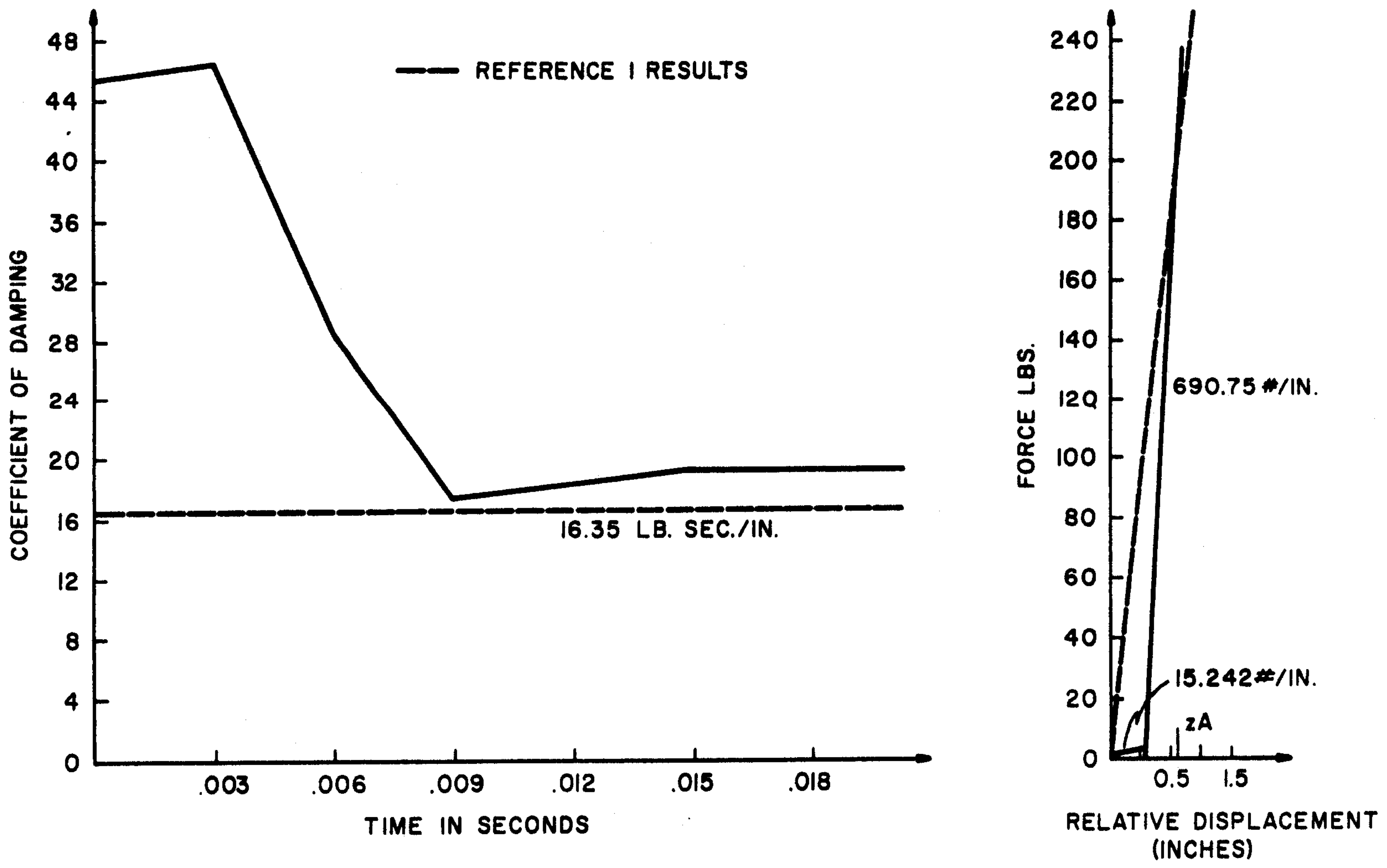


FIGURE 21. TERMINAL DESIGN FOR CASE 2_m PATH 1 AND 2

TABLE 1 MINIMIZE MAXIMUM ABSOLUTE ACCELERATION

SUMMARY OF STARTING DESIGNS

			lb. sec. /in.					
Case	Load Condition	Path	C_0	C_1	C_2	C_3	C_4	C_5
1_s	S (t) = 1000.0 in. / sec.^{2} for 0.05 sec.	1	65.0	65.0	65.0	65.0	65.0	65.0
		2	49.16	56.07	48.59	28.50	18.59	34.00
		3	47.16	54.69	46.59	38.65	29.50	47.74
1_m	Pulse II	1	88.2	68.2	48.2	34.158	25.6	35.655
		2	81.688	61.688	41.688	25.400	14.594	19.765
		3	84.900	64.900	45.337	33.295	21.477	22.079
2_m	Pulse I	1	45.0	45.0	45.0	45.0	45.0	45.0
		2	45.0	45.0	45.0	35.25	27.25	21.0
Ref 1	Pulse II	1	24.0	24.0	24.0	24.0	24.0	24.0
Ref 1	Pulse II	2	39.0	39.0	39.0	39.0	39.0	39.0
Test Case	Pulse II	1	32.0	32.0	32.0	32.0	32.0	32.0
Ref 1	Pulse I	1	3.0	3.0	3.0	3.0	3.0	3.0

TABLE 1, (continued)

lb./in.			in./sec.^{2}	in.
K_1	K_2	δ	Criterion Function Value	Behavior Function Value
370.0	370.0	0.25	1028.6	0.62931
30.0	670.0	0.145	861.7	0.933
40.0	200.0	0.30	759.45	1.022
42.369	201.79	0.50499	1008.9	1.1
368.39	368.39	0.0	965.56	1.1
30.809	670.10	0.49956	976.15	1.1
15.0	690.5	0.0	616.2	.29
15.0	690.0	0.18	537.4	0.503
----	1000.0	0	1165.0	-----
----	1000.0	0	1159.0	-----
----	750.0	0	1089.0	0.92
----	1000.0	0	580.0	-----

TABLE 2 MINIMIZE MAXIMUM ABSOLUTE ACCELERATION

SUMMARY OF INPUT DATA

		$\frac{\text{lb. sec.}}{\text{in. sec.}}$	lb. sec. /in.			
Case	Path	CBT	CAL_0	CAL_1	CAL_2	CAL_3
1_s	1	1600.0	0.0	0.0	0.0	0.0
	2	1600.0	0.0	0.0	0.0	0.0
	3	1600.0	0.0	0.0	0.0	0.0
1_m	1	1600.0	0.0	0.0	0.0	0.0
	2	1600.0	0.0	0.0	0.0	0.0
	3	1600.0	0.0	0.0	0.0	0.0
2_m	1	6000.0	0.0	0.0	0.0	0.0
	2	12000.0	0.0	0.0	0.0	0.0
Ref 1	1-II	0.0	0.0	0.0	0.0	0.0
Ref 1	2-II	0.0	0.0	0.0	0.0	0.0
Test Case	1	0.0	0.0	0.0	0.0	0.0
Ref 1	1-I	0.0	0.0	0.0	0.0	0.0

TABLE 2, (continued)

lb. sec. /in.		lb. /in.		in.	lb. sec. /in.	
CAL_4	CAL_5	CAL_6	CAL_7	CAL_8	CAU_0	CAU_1
0.0	0.0	4.5	K_1	0.0	100.0	100.0
0.0	0.0	4.5	K_1	0.0	100.0	100.0
0.0	0.0	4.5	K_1	0.0	100.0	100.0
0.0	0.0	4.5	K_1	0.0	100.0	100.0
0.0	0.0	4.5	K_1	0.0	100.0	100.0
0.0	0.0	4.5	K_1	0.0	100.0	100.0
0.0	0.0	4.5	K_1	0.0	100.0	100.0
0.0	0.0	4.5	K_1	0.0	100.0	100.0
0.0	0.0	0.0	0.0	---	100.0	100.0
0.0	0.0	0.0	0.0	---	100.0	100.0
0.0	0.0	---	4.5	0.0	100.0	100.0
0.0	0.0	---	0.0	0.0	-----	-----

TABLE 3 MINIMIZE MAXIMUM ACCELERATION
SUMMARY OF INPUT

		lb. sec. /in.				lb. /in.		in.		sec.
Case	Path	CAU_2	CAU_3	CAU_4	CAU_5	CAU_6	CAU_7	CAU_8	zA	DT
1_s	1	100.0	100.0	100.0	100.0	700.0	700.0	1.1	1.1	.0125
	2	100.0	100.0	100.0	100.0	700.0	700.0	1.1	1.1	.0125
	3	100.0	100.0	100.0	100.0	700.0	700.0	1.1	1.1	.0125
1_m	1	100.0	100.0	100.0	100.0	700.0	700.0	1.1	1.1	.0125
	2	100.0	100.0	100.0	100.0	700.0	700.0	1.1	1.1	.0125
	3	100.0	100.0	100.0	100.0	700.0	700.0	1.1	1.1	.0125
2_m	1	100.0	100.0	100.0	100.0	700.0	700.0	0.6	0.6	.003
	2	100.0	100.0	100.0	100.0	700.0	700.0	0.6	0.6	.003
Ref 1	1-II	100.0	100.0	100.0	100.0	-----	-----	---	1.1	----
Ref 1	2-II	100.0	100.0	100.0	100.0	-----	-----	---	1.1	----
Test Case	1	100.0	100.0	100.0	100.0	-----	-----	0.0	1.1	.01
Ref 2	1-I	-----	-----	-----	-----	-----	-----	0.0	0.6	----

TABLE 4 MINIMIZE MAXIMUM ACCELERATION

SUMMARY OF TERMINAL POINTS

			lb. sec./in.					
Case	Path	Fig.	C_0	C_1	C_2	C_3	C_4	C_5
1_s	1	18	81.689*	81.689*	41.689*	25.401	14.595	19.766
	2	18	84.899*	64.983*	45.339	33.297	21.499	22.089
	3	18	88.205*	68.205*	48.205*	34.160	25.610	35.657
1_m	1	19	60.942	41.165	45.826	34.434	26.374	36.397
	2	20	57.830	38.036	39.987	27.607	18.655	23.087
	3	21	59.277	39.395	44.843	32.259	18.763	18.903
2_m	1	22	45.477	46.695	28.707	17.738	18.355	19.394
	2	22	45.295	46.853	27.788	19.891	15.663	18.643
Ref 1	1-II	--	42.6	42.6	42.6	42.6	42.6	42.6
Ref 1	2-II	--	41.9	41.9	41.9	41.9	41.9	41.9
Test Case	1	--	42.4	42.4	42.4	42.4	42.4	42.4
Ref 1	1-I	--	15.9	15.9	15.9	15.9	15.9	15.9

* Denotes active constraint for damping time rate of change.

TABLE 4, (continued)

lb./in.		in.	in./sec.^{2}	in.	Percent Reduction in Criterion Value with Reference to Ref 1
K_1	K_2	δ	Criterion Function Value	Behavior Function	
368.46	368.46	0.0	635.31	1.1$^+$	28.9
30.81	670.12	0.49958	626.35	1.1$^+$	29.9
42.37	201.81	0.5057	639.76	1.1$^+$	28.4
42.04	201.88	0.16446	712.73	1.1$^+$	20.3
368.39	368.53	0.0	697.33	1.1$^+$	21.95
30.479	669.98	0.36007	690.34	1.1$^+$	22.75
15.242	690.75	0.26556	316.08	0.6$^+$	8.65
14.731	689.89	0.24599	315.56	0.6$^+$	8.66
------	36.5	0.0	893.5	1.1$^+$	-----
------	48.3	0.0	893.6	1.1$^+$	-----
------	38.0	0.0	893.5	1.1$^+$	-----
------	409.9	0.0	346.4	0.6$^+$	-----

$^+$ Denotes active behavior constraint.

TABLE 5 MINIMIZING MAXIMUM RELATIVE DISPLACEMENT

Initial Designs

			lb.in./sec.					
Case	Path	Pulse	C_0	C_1	C_2	C_3	C_4	C_5
3_m	1	II	10.0	10.0	10.0	10.0	10.0	10.0
	2	II	20.0	20.0	20.0	20.0	20.0	20.0
Ref 1-D	1	II	0.0	0.0	0.0	0.0	0.0	0.0

Input Data

		lb. in.	lb.in./sec.					
Case	Path	CBT	CAL_0	CAL_1	CAL_2	CAL_3	CAL_4	CAL_5
3_m	1	1600.0	0.0	0.0	0.0	0.0	0.0	0.0
	2	1600.0	0.0	0.0	0.0	0.0	0.0	0.0
Ref 1-D	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Input Data

		lb.in./sec.			lb./in.		in.	in./sec^2	sec.
Case	Path	CAU_3	CAU_4	CAU_5	CAU_6	CAU_7	CAU_8	xA	DT
3_m	1	30.0	30.0	30.0	K_2	700.0	1.1	932.4	0.0125
	2	30.0	30.0	30.0	K_2	700.0	1.1	932.4	0.0125
Ref 1-D	1	30.0	30.0	30.0	-----	-----	0.0	932.4	------

TABLE 5, (continued)

Initial Designs

lb./in.		in.	in.	in./sec.^{2}
K_1	K_2	δ	Criterion Function Value	Behavior Function
100.0	300.0	1.0	2.78	458.2
200.0	400.0	1.0	1.55	781.2
-----	50.0	0.0	7.0	-----

Input Data

lb./in.		in.	lb.in./sec.		
CAL_6	CAL_7	CAL_8	CAU_0	CAU_1	CAU_2
0.0	4.5	K_1	0.0	30.0	30.0
0.0	4.5	K_1	0.0	30.0	30.0
0.0	-----	0.0	0.0	30.0	30.0

TABLE 6 MINIMIZE MAXIMUM RELATIVE DISPLACEMENT

SUMMARY OF TERMINAL DESIGNS

		lb. sec. /in.					
Case	Path	C_0	C_1	C_2	C_3	C_4	C_5
3_m	1	30.0*	30.0*	30.0*	30.0*	30.0*	30.0*
	2	30.0*	30.0*	30.0*	30.0*	30.0*	30.0*
Ref 1-D	1	29.9*	29.9*	29.9*	29.9*	29.9*	29.9*

lb. /in.		in.	in.	in. /sec.	Percent Reduction 2 in Criterion Value with Reference to
K_1	K_2	δ	Criterion Function Value	Behavior Function	Ref 1
332.1	700.0*	0.74510	1.0792	932.38$^+$	2.42
332.1	700.0*	0.74510	1.0792	932.38$^+$	2.42
-----	333.0	0.0	1.1	932.38$^+$	----

$^+$ Denotes active behavior constraint.

* Denotes active upper bound on damping.

REFERENCES

1. Schmit, L. A. and Fox, R. L., "Synthesis of a Simple Shock Isolator," Engineering Design Center, EDC Report No. 2-63-4, Case Institute of Technology, Oct., 1961.

2. Best, G. C., "Completely Automatic Weight-Minimization Method for High-Speed Digital Computers," Journal of Aircraft Vol. 1, No. 3, May-June 1964.

3. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints," Journal of Society of Applied Mathematics, Vol. 8, No. 1, March 1960.

4. Scarborough, J. B., Numerical Mathematical Analysis, Johns Hopkins Press, Baltimore, Maryland, 4th Edition, p. 317, 1958.

5. "Communications of the Aids to Computational Machinery," Vol. 7, No. 1, January 1964.

6. Collatz, L., The Numerical Treatment of Differential Equations, Springer-Verlag, 3rd Edition, Berlin, Germany, pp. 60-70, 1960.

7. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part II. Nonlinear Constraints," Journal of Society of Applied Mathematics, Vol. 9, No. 4., December 1961.

8. Schmit, L. A. and Kinser, David E., "The Synthesis of a Laminated Plate for High Temperature Application," EDC Report No. 2-64-7, Case Institute of Technology.

9. Schmit, L. A. and Mallett, Robert H., "Design Synthesis in Multi-dimensional Space with Automated Material Selection," EDC Report, Case Institute of Technology.

10. Kuhn, H. W. and Tucker, A. W., "Nonlinear Programming," Proc. 2nd Berkeley Symp. on Mathematical Statistics and Probability, University of California Press, pp. 481-492, 1951.

[illegible]

APPENDIX I

The equation of motion was obtained by Figure 1, using Newton's law $\Sigma F = M \cdot \ddot{x}$, where M = mass and $\ddot{x}$ = acceleration. The forces acting on the mass are

$$+ \bar{\bar{c}} \cdot (\dot{y} - \dot{x}) + K (y - x)$$

Then $M\ddot{x}$ is

$$M\ddot{x} = + \bar{c} (\dot{y} - \dot{x}) + K (y - x) \qquad \text{(I-1)}$$

Letting $z = y - x$ then $\dot{z} = \dot{y} - \dot{x}$ and $\ddot{x} = \ddot{y} - \ddot{z}$ putting this in (I-1) gives

$$M\ddot{z} + \bar{c}\dot{z} + \bar{K}(z) = M\ddot{y} \qquad \text{(I-2)}$$

Letting M = unity and replacing $\ddot{y}$ with S(t), the input acceleration, (I-2) becomes

$$\ddot{z} + \bar{c}\dot{z} + K(z) = S(t) \qquad \text{(I-3)}$$

The acceleration that the mass 'experiences' is equal the acceleration of the mass with respect to a fixed point. That is absolute acceleration $\ddot{x} = \ddot{y} - \ddot{z}$.

From (I-3) it is seen that the acceleration felt by the mass is

$$S(t) - \ddot{z} = \bar{c} \cdot \dot{z} + K(z).$$

It is difficult to solve (I-3) explicitly because of the characteristics of $\bar{c}(t)$ and $K(z)$. A numerical integration technique, the Runge-Kutta method has been chosen to obtain the unknown displacements, velocities, and accelerations. The Runge-Kutta method is cumbersome if hand calculations are used, but it lends itself quite easily to automated computation. The method is accurate and efficient with respect to computer storage space for only information pertaining to the previous point is needed to obtain the next point. The Runge-Kutta method is of order h^4.

In order to use the Runge-Kutta method equation (I-3) had to be transformed into two first order simultaneous equations.

Let

$$\frac{dz}{dt} = y$$

and $$\dot{y} = \ddot{z} = -\bar{c} \cdot \dot{z} - K(z) + S(t)$$

The general formula for two simultaneous ordinary differential equations is shown below(4).

Let $dz/dt = f_1(t, z, y)$

and

$$\frac{dy}{dt} = f_2(t, z, y)$$

also

$$K_1 = f_1(t_o, z_o, y_o) \; \Delta t$$

$$L_1 = f_2(t_o, z_o, y_o) \; \Delta t$$

$$K_2 = f_1(t_o + \frac{1}{2} \Delta t, \; z_o + \frac{1}{2} K_1, \; y_o + \frac{1}{2} L_1) \; \Delta t$$

$$L_2 = f_2(t_o + \frac{1}{2} \Delta t, \; z_o + \frac{1}{2} K_1, \; y_o + \frac{1}{2} L_1) \; \Delta t$$

$$K_3 = f_1(t_o + \frac{1}{2} \Delta t, \; z_o + \frac{1}{2} K_2, \; y_o + \frac{1}{2} L_2) \; \Delta t$$

$$L_3 = f_2(t_o + \frac{1}{2} \Delta t, \; z_o + \frac{1}{2} K_2, \; y_o + \frac{1}{2} L_2) \; \Delta t$$

$$K_4 = f_1(t_o + \Delta t, \; z_o + K_3, \; y_o + L_3) \; \Delta t$$

$$L_4 = f_2(t_o + \Delta t, \; z_o + K_3, \; y_o + L_3) \; \Delta t$$

Then going from the point (z_o, y_o, t_o) to $(z_o + \Delta z, \; y_o + \Delta y, \; to + \Delta t)$, where Δt is specified, Δz and Δy are found from the formulas below.

$$\Delta z = \frac{1}{6} (K_1 + 2K_2 + 2K_3 + K_4)$$

$$\Delta y = \frac{1}{6} (L_1 + 2L_2 + 2L_3 + L_4).$$

For this case,

t = time

z = displacement

y = velocity

Furthermore, from (I-3)

$$f_1\ (t,\ z,\ y) = \frac{dz}{dt} = y$$

and

$$f_2\ (t,\ z,\ y) = \frac{dy}{dt} = -\overline{c}(t) \cdot \dot{z} - K(z) + S(t)$$

This method may easily be extended to N equation for dx_1/dt, $dx_2/dt, \ldots, dx_n/dt$ and put in matrix form[(5)]. However, it was found to be too time consuming for the 2 x 2 matrices resulting from this second order equation.

The next step is to obtain a feasible error analysis and thereby control the step size so that one may place a tolerance at any point on the unknowns z and y. The Runge-Kutta method has error of $0\ (h^4)$. Max Lotkin in reference (3) gives an error bound for the i^{th} unknown as

$$|E_i| \leq \frac{973}{720}\ M\ L^4\ h^5,$$

where $h = \Delta t$

$$|f_i\ (t,\ x_1,\ \ldots,\ x_n)| \geq M$$

and

$$\left| \frac{\partial^{p+q+r}\ f_i\ (t, x_1,\ \ldots,\ x_N)}{\partial\ t^p\quad \partial x^q\quad \partial y^r} \right| \leq \frac{L^{p+q+n}}{M^{q+r+1}}$$

for $p+q+r \leq 4$.

Recall that $f_1(t, x, y) = dz/dt$ and $f_2(t, x, y) = dy/dt$. In order to obtain a non-zero error bound, r was set equal to 1 and p = q = 0.

Then $|y| \leq M, \quad |df_1/dy| = 1.0 = L/M^2.$

or, $L = M^2$ and $|E_1|$ that is the error of y is

$\leq (\frac{973}{720})\ M \cdot (M^2)^4\ h^5.$

In terms of physical quantities this means that the error is proportional to M^9 or y^9 or the velocity to the ninth power. For $M > 1.0$ this is an intolerable error. To check the validity of this error analysis an example equation was solved both exactly and by Runge-Kutta method and the difference at each solution point was recorded along with the error bounds given by the formulation above.

The equation $x'' + x' + x = (t-1) \exp(-t) + \cos(t)$ has the solution $x = t \cdot \exp(-t) + \sin(t)$ for the initial conditions $x(0) = 0$ and $\dot{x}(0) = 2$.

The predicted error was found to be as large as 10^4 times the actual error. It was concluded that the error analysis was too conservative to be used for step size control.

A rule of thumb for step size control is given in Reference 6. The rule states that the ratio $(K_2-K_3)/(K_1-K_2)$ from equation (I-a) should be less than 2%. Upon examining this, it is seen that this method would not be efficient if the system were large and the K's for each unknown had to be checked at each point. It was concluded that a predetermined step size, obtained by observing the convergence of the solution as the step size decreased was not out of order for this problem.

A test analysis case using the proposed Runge-Kutta method was done for a spring, mass, damper system which has constant spring stiffness equal to 36.5 lb/in, mass of one and

$$S(t) = \begin{cases} 1000.0 & t \leq 0.05 \text{ seconds} \\ 0.0 & t > 0.5 \text{ seconds} \end{cases}$$

The exact solution found in Reference 1 and that obtained by the numerical technique are respectively: maximum acceleration 893.6 in/sec.^{2}, 893.68 in/sec.^{2} maximum deflections 1.1 in. and 1.1025 in.

Both of these maximum quantities occur at the same time. It was concluded that the Runge-Kutta formulation would be accurate and efficient enough to use for the analysis.

APPENDIX II
CONVERGENCE

A valuable but conservative convergence criteria is given in Reference 7. However, operating computer time is an important factor in optimization problems. It was thus deemed worthwhile to further explore this termination criteria in order to (1) place a less conservative relation between the true optimum merit and the best merit obtained by the synthesis program, and (2) reduce the costly computing time spent trying to lower the merit value when it is already within a prescribed tolerance of a local optimum design.

The tolerance ϵ, where ϵ is greater than zero, is defined as the difference between the merit at design point $\vec{C}$ and the true focal optimum $\vec{C}_o$, or global minimum if the acceptable design space and criterion function are convex. According to the ϵ sign convention used here, ϵ would be less than zero if a maximum were sought.

The reference proves that if $\vec{C}_o$ is the optimum design and $M(\vec{C}_o)$ the optimum merit, then at any acceptable design point, $\vec{C}$,

$$M(\vec{C}) - M(\vec{C}_o) \leqq \epsilon \qquad \text{(II-1)}$$

provided the inequality below holds.

$$\beta(\vec{C}) \leqq \epsilon/2ML\alpha \qquad \text{(II-2)}$$

Reference 7 states that if $\beta(\vec{C})$ goes to zero this condition is equivalent to the Kuhn-Tucker conditions. (See Appendix IV)

The quantities L, M, and α are defined below.

M = Number of design variables defining the space.

L = The maximum "distance" between two acceptable points in the design space. A value for L is obtained from

$$L^2 = \sum_{i=1}^{M} (C_i^{upper} - C_i^{lower})^2.$$ C_i^{upper} and C_i^{lower} are the bounds of the variables such that all the acceptable points are enclosed in the rectangular space of dimensions

$$\{C_o^{upper} - C_o^{lower}, C_1^{upper} - C_1^{lower}, \ldots, C_8^{upper} - C_8^{lower}\}$$

$$\alpha = \text{SQRT}\,(\sum_{i=0}^{8} (V_q\vec{g})_i^{\,2}) / \text{SQRT}\,(\sum_{i=0}^{8} g_i^{\,2})$$

where g_i are the components of the gradient $\vec{g}$ and V_q is the matrix $(NTC^T\, NTC)^{-1}$.

The contention here is that (II-2) really implies

$M(\vec{C}) - M(\vec{C}_o) \leq K' \cdot \epsilon$ where $K' < 1$. Starting with

$$M(\vec{C}) - M(\vec{C}_o) \cong (\vec{C} - \vec{C}_o)^T \vec{g},$$

the reference shows that the gradient, $\vec{g}$, may be rewritten for convenience as

$$\vec{g} = P_q(\vec{C})\,\vec{g} + \sum_{i=1}^{q} \gamma_i U_i \qquad \text{(II-4)}$$

where γ_i are the scalar components of $V_q(\vec{C})\, NTC^T(\vec{C})\, \vec{g}$ equal to $\{\gamma_1, \gamma_2, \ldots, \gamma_q\}$ and q is the number of active constraints.

$$P_q(\vec{C}) = I - NTC \cdot V_q \cdot (NTC)^T$$

u_i in (II-4) are the normalized vectors spanning the subspace defined by the independent vectors of NTC or those unit normals to the active constraints. Let $(C - C_o)^T$ be denoted by $\vec{y}'$ for simplicity. Then $\vec{y}'\,\vec{g}$ of (II-4) becomes

$$\vec{y}^T\vec{g} = y^T\vec{P}(C)\,\vec{g} + y^T \sum_{i=1}^{q} \gamma_i u_i$$

or since γ_i's are scalars

$$\vec{y}^T\vec{g} = y^T P(\vec{C})\,\vec{g} + \sum_{i=1}^{q} \gamma_i (y^T \vec{u}_i) \qquad \text{(II-5)}$$

It is given that $\beta(\vec{C}) \leq \epsilon/2ML$.

$$\beta 1(\vec{C}) = \operatorname*{Max}_i \{ \tfrac{1}{2} \gamma_i v_{ii}^{-1/2} \} \qquad i = 1, 2, \ldots M \qquad \text{(II-6)}$$

v_{ii} are the diagonal elements of V_q. Furthermore,

$$\beta(\vec{C}) = \text{Max} \{ || Pq(\vec{C}) \vec{g} ||, \beta 1(\vec{C}) \} \qquad \text{(II-7)}$$

Thus from (II-2) $\beta 1(\vec{C}) \leq \epsilon/2LM\alpha$ and solving the inequality (II-6)

$$\frac{1}{2} \gamma_i v_{ii}^{-1/2} \leq \epsilon/2LM\alpha$$

or

$$\gamma_i \leq \epsilon v_{ii}^{1/2} /2mL\alpha$$

Using $\sum_{i=1}^{q} v_{ii}^{1/2}$ and knowing $|y^T u_i| \leq |y^T| |u_i|$ and $|u_i| \equiv 1$ with the fact that $|y^T|$ cannot be greater than L, the term $\sum_{i=1}^{q} \gamma_i (y^T u_i)$ of (II-5) is $\leq L\epsilon \sum_{i=1}^{q} v_{ii}^{1/2} / mL\alpha$.

Using (II-7) again,

$$|| Pq(\vec{C}) \vec{g} || \leq \epsilon/2ML\alpha$$

thus

$$y^T P_q (\vec{C}) \vec{g} \leq |y^T| \cdot |P_q (\vec{C}) \vec{g}| \leq L\epsilon/2Lm\alpha. \qquad \text{(II-8)}$$

Then (II-1) becomes

$$M(\vec{C}) - M(\vec{C}_o) \leq \epsilon(\sum_{i=1}^{q} v_{ii}^{1/2} + \frac{1}{2})/m\alpha.$$

Since $\alpha^2 = \sum_{j}^{q} \sum_{i}^{q} v_{ij}^2$ the quantity

$$\sum_{i=1}^{q} \frac{v_{ii}^{1/2}}{a} + \frac{1}{2a} \leq M$$

Since each of the terms are less than 1 (M=9)

$$(\sum_{i=1}^{9} v_{ii}^{1/2} + \frac{1}{2}/Ma < 1$$

and is the previously sought quantity K'.

In (II-8) the right hand side can be replaced with ϵ', the new tolerance. Then a relation in terms of $K\epsilon$ is obtained to be placed in (II-2) which is the test of the validity of (II-8). Now (II-2) becomes

$$\beta\,(\vec{C}) \leq \frac{\epsilon\,Ma}{\sum_{i=1}^{q} v_{ii} + \frac{1}{2}} / \,2MLa \quad \text{or} \quad \beta\,(\vec{C}) \leq \frac{\epsilon}{(2\sum_{i=1}^{q} v_{ii}^{V_2} + 1)L}$$

(II-9)

APPENDIX III
DIRECTION AND STEP SIZE

The constraints which any acceptable design must obey are 29 in number. Twenty-two are placed in the coefficient of damping.

$C_j - C_{j+1} \leq CBT \cdot DT \qquad \text{for } j = 0, 1, \ldots, 4 \qquad (5)$

(A)

$C_j - C_{j+1} \leq CBT \cdot DT \qquad \text{for } j = 0, 1, \ldots, 4 \qquad (5)$

$C_j \leq C_j \text{ upper} \qquad j = 0, 1, \ldots, 5 \qquad (6)$

(B)

$C_j \geq C_j \text{ lower} \qquad j = 0, 1, \ldots, 5 \qquad (6)$

Five are placed on the spring system.

$C_6 = K_1 \geq K_1 \text{ lower}$

$C_7 = K_2 \geq K_1$

(5) (C)

$C_7 \leq K_2 \text{ upper}$

$C_8 = \text{gap} \leq \text{allowable deflection}$

$C_8 = \text{gap} \geq 0$

Two are placed on the relative deflection.

Maximum deflection $\leq$ XA

(2) (D)

Minimum deflection $\geq$ -XA where XA > 0

TOTAL 29

The matrix, referred to as NTC, is composed of columns which are the normal vectors of the active constraints printing 'into' the acceptable design region. The candidates for NTC are stored in an array denoted by R(I, J) for I = 0, 1, . . . , 8; J = 0, 1, . . . , 29. R is generated in the program. The first two columns must be redetermined every time R is needed, because they represent the normal to the deflection or acceleration constraints as the case may be. The remaining columns are constant and need be formulated once. The R matrix is shown in Figure III-1. The odd number rows 1 thru 31 refer to the lower bounds and the even numbered rows to the upper bounds. Rows 3 thru 12 are divided by $\sqrt{2}$ to be normalized. These rows represent constraint set A. The remaining constraints sets B, C, and D are represented in columns 13 thru 24, 25 thru 31, and 1 thru 2 respectively.

If no constraints are active, the move in the design space is the gradient direction. However, if one or more constraints are in violation at the jth design, it is desirable to find the largest component of the gradient having the property that it does not point into a constraint 'wall.' Before proceeding, it should be recalled

R MATRIX

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	grad (Defl.)	-grad (Defl.)	1	-1									1	-1																	
1			-1	1	1	-1									1	-1															
2					-1	1	1	-1									1	-1													
3							-1	1	1	-1									1	-1											
4									-1	1	1	-1									1	-1									
5											-1	1											1	-1							
6																									1						-1
7																												-1			1
8																													1	-1	

FIGURE III-1

that when the j th design point is on a constraint the most advantageous moves are not always along that constraint. Thus, it is desirable to have the ability to move off of any constraint at any time during the redesign process and also the ability to determine when it is desirable to leave the constraint and when to remain on it.

Where $\overrightarrow{NTC_j}$ is the normal of the j th constraint in violation and $\overrightarrow{ACG}$ is the gradient, if $\overrightarrow{ACG}$ has a positive component in the $\overrightarrow{NTC_j}$ direction the result is to move off the constraint. If the inner product is negative, the component of the gradient in the plane of the constraint j is subtracted from the gradient resulting in remaining on the constraint surface.

The direction sought is termed $\vec{u}$. The vector $\vec{u}$ has the effect of removing the component of $\overrightarrow{ACG}$ which will violate the j th constraint. A more regerous development of $\vec{u}$ is shown below. The development is taken from Reference 2. Derivation of move direction to max $u^T g(\bar{x})$ with $N_i(\bar{x})$ for $i = 1, 2, \ldots Q \leq M-1$ and $\bar{x} = (x_1, x_2, \ldots x_M)$. The N_i denoting the active constraint normals.

Denote the gradient by $\vec{g}$, and

$$[\ N_1, N_2, \ldots N_Q\ ] \text{ by NTC.}$$

Then the allowable direction $\vec{u}$ must be orthogonal to all constraint normals in order to lie in their tangent plane or $NTC^T \vec{u} = 0$. For convenience, let the magnitude of $\vec{u}$ be 1 or $\vec{u}^T\vec{u} = 1$.

The problem may be solved by the method of Lagrangian multipliers for constraints satisfied as equalities. That is, maximize ϕ with $\vec{u}^T\vec{u} = 1$ and $NTC^T\vec{u} = \vec{0}$. Thus,

$$\phi = \vec{g}^T\vec{u} + \vec{\lambda}_1{}^T\ NTC^T\vec{u} + \lambda_2 (1 - \vec{u}^T\vec{u}) \tag{1}$$

where λ_1 is column vector of the Lagrangian multipliers λ_{1j} and λ_2 is a single multiplier to be found.

Setting

$$\frac{\partial\phi}{\partial u_j} = 0 \text{ gives } (g^T)^T + (\vec{\lambda}_1{}^T NTC^T)^T + 2\vec{\lambda}_2 (u^T)^T = 0 \tag{2}$$

$$\text{for } j = (1, 2, \ldots M), \text{ or } g + NTC\ \vec{\lambda}_1 + 2\lambda_2\ \vec{u} = \vec{0} \tag{3}$$

Then using the fact that $NTC^T u = \vec{0}$ and multiplying (2) by NTC^T gives

$$NTC^T\vec{g} + (NTC^T\ NTC)\ \vec{\lambda}_1 = 0 \tag{4}$$

The inverse of $(NTC^T\ NTC)$ exists because the columns of NTC are independent. Let $NTC^T\ NTC)^{-1} = VQ$

$$\lambda_1 \text{ is then } - VQ\, NTC^T \vec{g} \qquad (5)$$

$\vec{u}$ is found in terms of λ_2 from equation (3) to be

$$\vec{u} = \frac{1}{2\lambda_2} \{\vec{g} - NTC\cdot VQ\cdot NTC^T \vec{g}\}$$

λ_2 is found by requiring $\vec{u}^T\vec{u} = 1$

$$u^T = \frac{1}{2\lambda_2} [\{I-NTC\cdot VQ\cdot NTC^T\}g]^T = \frac{1}{2\lambda_2} [g^T[I-NTC\cdot VQ\cdot NTC^T]^T]$$

$$\vec{u}^T u = 1 = \frac{1}{4\lambda_2^2} \vec{g}^T [I - NTC\cdot VQ \cdot NTC^T]^T \cdot [I-NTC\cdot VQ\cdot NTC^T] \vec{g}$$

$$\lambda_2^2 = \frac{1}{4} \vec{g}^T [I - NTC\cdot VQ\cdot NTC^T] \cdot [I - NTC\cdot VQ\cdot NTC^T] \vec{g}$$

Thus the direction of $\vec{u}$ is $\vec{g} - NTC\cdot VQ\cdot NTC^T \vec{g}$

Again it is emphasized that if any of the columns of NTC have a positive inner product with gradient that column is deleted, thus allowing freedom to move off of the constraint.

The process for determining the first step size, L, is derived for strictly linear constraints. However, with the corrective length process described and due to the nature of the constraints, the procedure applies itself well near the one nonlinear constraint.

After $\vec{u}$ is found, the difference between the allowable bound $\bar{B}_i$, and the value of the bound function $B(\vec{C})$ is determined for every constraint.

That is, $\Delta \bar{B}_i = e_1^i (\bar{B}_i - b_i(x))$.

The term e_1^i is +1 if i refers to an upper bound and -1 for lower bounds. Thus $\Delta \bar{B}_i > 0$ indicates an acceptable region and $\Delta\bar{B}_i < 0$ an unacceptable region. When $\Delta \bar{B}_i = \epsilon$ the constractive is said to be active.

The rate of change of $\Delta \bar{B}_i$ in the $\vec{u}$ direction is found. With this linear estimation the length of $\vec{u}$ to render constraint i active is found. $\Delta \bar{B}_i$ changes with $\vec{u}$ as the inner product of the i^{th} column of the R matrix and $\vec{u}$.

This is easily seen by realizing that the i column of the R matrix (defined in the first part of this Appendix) is a vector orthogonal to the i^{th} constraint. The component of $\vec{u}$ in the R_i direction or $\Delta\bar{B}_i$, is $(\vec{R}_i, \vec{u})$ for $|\vec{u}| = 1$. Thus, for $L \cdot \vec{u}$, where L is the LINLEN length, $\Delta\bar{B}_i$ can be forced as close to ϵ as desired.

$$L_i = \frac{\Delta \bar{B}_i}{(\vec{R}_i, \vec{u})}$$

The minimum L_i found from testing all the constraints is used. The general formula for L is

$$L = \epsilon_1^i \epsilon_2^i \text{MIN} \quad \frac{\Delta \bar{B}_i}{|(\vec{R}_i, \vec{u})|}$$

$\epsilon_2^{\,i}$ is the sign of $(\vec{R}_i, \vec{u})$

There is one restriction on allowable L's "That is any L which is negative and $\Delta\bar{B}_i$ is positive should be ignored." The reason for this restriction is because positive $\Delta\bar{B}_i \nRightarrow$ an acceptable design and negative L is the opposite direction of $\vec{u}$ which means an increase in merit rather than a decrease. The increase would be permissible if $\Delta\bar{B}$ is negative or the program is trying to return to the acceptable region.

APPENDIX IV
KUHN-TUCKER CONVERGENCE CONDITIONS

Before describing the Kuhn-Tucker[10] convergence conditions it will be helpful to make several definitions.

A function f(x) is convex if

$$(1-\theta)\, f(\vec{x}') + \theta f(\vec{x}) \geq f\{(1-\theta)\, \vec{x}' + \theta\, \vec{x}\} \qquad \text{(IV-1)}$$

for all $0 \leq \theta \leq 1$. All $\vec{x}$ and $\vec{x}'$ must be in region such that $f(\vec{x})$ and $f(\vec{x}')$ are defined.

A function $f(\vec{x})$ is concave is $-f(\vec{x})$ is convex; that is,

$$(1 - \theta)\, f(\vec{x}') + \theta\, f(\vec{x}) \leq f\{\, (1-\theta)\, \vec{x}' + \theta\, x\} \qquad \text{(IV-2)}$$

for all $0 \leq \theta \leq 1$. Again all $\vec{x}$ and $\vec{x}'$ must be in the region for which f(x) is defined.

The convergence theorem states that at a local maximum if one or more constraints are satisfied as equalities, then the negative gradient of the criterion function will be a nonnegative linear combination of the gradients to the constraints.

Let the constraints be of the form

$g_i\,(\vec{x}) \geq 0$ for i = 1, 2, ... M. and the criterion be $C(\vec{x})$ to be minimized.

The minus gradient of the criterion lies in the convex cone of the gradients of the active constraints.

To test whether $\vec{x}^o$ is a local minimum, solve the equation IV-3 for a_1 and a_2.

$$a_1 \nabla g_1(\vec{x}_1) + a_2 \nabla g_2(\vec{x}^o) = - \nabla C(\vec{x}^o) \qquad \text{(IV-3)}$$

where

$\nabla g_1(\vec{x}^o)$, $\nabla g_2(\vec{x}^o)$ and $\nabla C(\vec{x}^o)$ are vectors. If a_1 and a_2 are nonnegative, the point $\vec{x}^o$ is a local minimum.

The conditions for the above to be valid are that the design space be convex or satisfy (IV-1) and that the criterion function be convex at least in the region for which (IV-3) is checked. In general it is not know whether the conditions above are true. In this case if (IV-3) is satisfied, further time spent optimizing can be termed "confidence time."

APPENDIX V
GLOBAL SYMBOLS OF COMPUTER PROGRAM

RS	Vector containing one component for each constraint. Components have integer value 1 if constraint is active and program wants to remain on constraint; 2 if constraint is active and program wants to get off; and 0 if constraint is not in violation.
INTEGERS	
I	Values from 0 to 8, used in analysis procedure to denote gradient components and -1 denotes best criterion value at present time.
N	Number of steps required to analyse a design.
P	Indexing integer.
J	Indexing integer.
K	Indexing integer.
COL	Number of active constraints which program does not want to get off (corresponds to number of 1's in RS)
F	Has value 7 if C(8) is held fixed, 8 otherwise.
FS	
WEDGE, VALY	Value 0, 1, 2 when moving normal to nonlinear constraint.

CS8, CS9	Input variables allowing program to work with first 8 variable for CS8 steps and all 9 for CS9 steps.
LC	Input; number of load condition.
ALC	Number of active load condition.

BOOLEAN VARIABLES; VALUE TRUE OR FALSE

GRD	True in analysis procedure when determining gradients.
ONE, CDA, CUSP PK, FO	True when moving normal to nonlinear constraint.
EXAM, PS	Suppresses unwanted printout when moving normal to nonlinear constraint.
PPG	True when no constraints are active and gradient method gets stuck in 'cusp.'
SB1	True when PPG is true for two consecutive steps.

Real Variables

T	Time at each step of analysis procedure.
H	Step size in analysis procedure.
XA	Maximum allowable deflection or acceleration.
DT	Time between damping coefficient variables.
Q, Q1, Q2, Q3, T1	Temporary storage locations.
CBT	Absolute value of maximum allowable time late of change of damping.
CBM	Upper bound on damping.

EP	Tolerance used for constraints.
EPL	Tolerance used in convergence check.
L	Maximum length of allowable move which does not violate constraints.
MS	Set at 10^6 used when moving normal to nonlinear constraint.
XE	Displacement at which maximum acceleration occurs.
KLB	Lower bound on spring constants.
KUB	Upper bound on spring constants.
FS11	Denotes amount variable C(8) is changed when C(8) is held fixed.

Storage Arrays:

C1( )	Vector containing best possible design at present time.
C( )	Vector containing design to be compared with C1( ).
TM( )	Vector containing times of points used in analysis.
D( )	Vector containing displacements associated with TM( ).
V( )	Vector containing velocities associated with TM( ).
ACCT( )	Vector containing acceleration associated with TM.
DC( )	Vector containing variations of variables used in finding gradient.

ACGR	Matrix containing gradient values of previous steps.
MAC(-1)	Best criterion value at present from design C1( ).
MD(-1)	Nonlinear constraint value associated with MAC(-1).
MAC( ), MD( )	Vector index from 0 to 8 stores values associated with criterion and constraint functions respectively.
VQ(,)	Array storing inverse of outer product of normals to constraints = $(NTC\ NTC^T)^{-1}$.
CAL( )	Input vector of allowable lower values for variables.
CAU( )	Input vector of allowable upper values for variables.
ACG( )	Vector of gradient components.
DG( )	Vector normal to nonlinear constraints.
V2( ), V3( ), V4( ), V5( ), DC1( ), DC2( ), S1(,) S( ), U1( ), U( )	Temporary storage vectors.
SMX( )	Input vector of load pulses.
TL( )	Input vector of load pulse times.
NTC(,)	Array storing vector of normals to active constraints.
R(,)	Matrix storing normals to linear constraints.

Procedure Names:

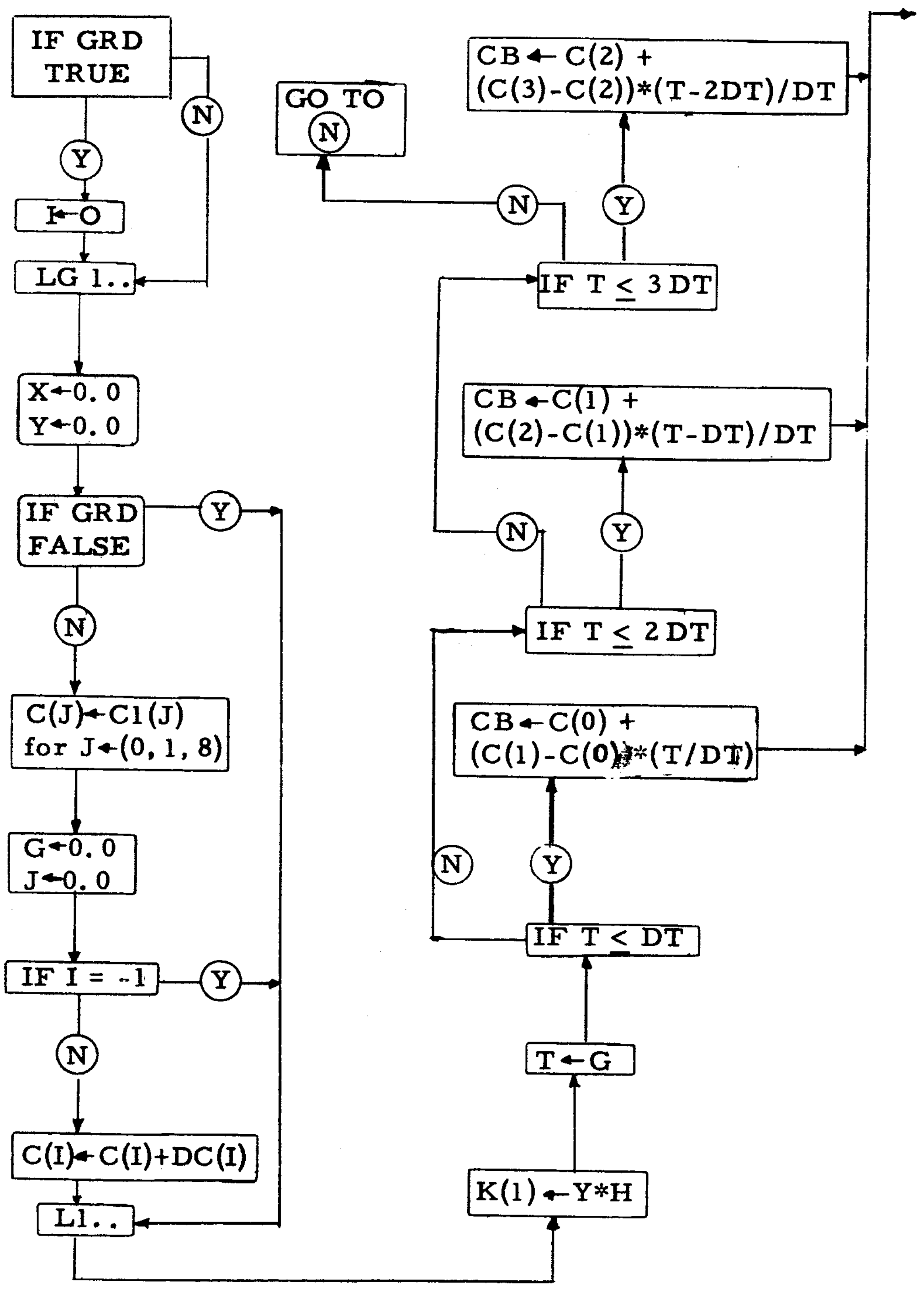
ANL	Analyses given design, gives maximum acceleration, maximum displacement,

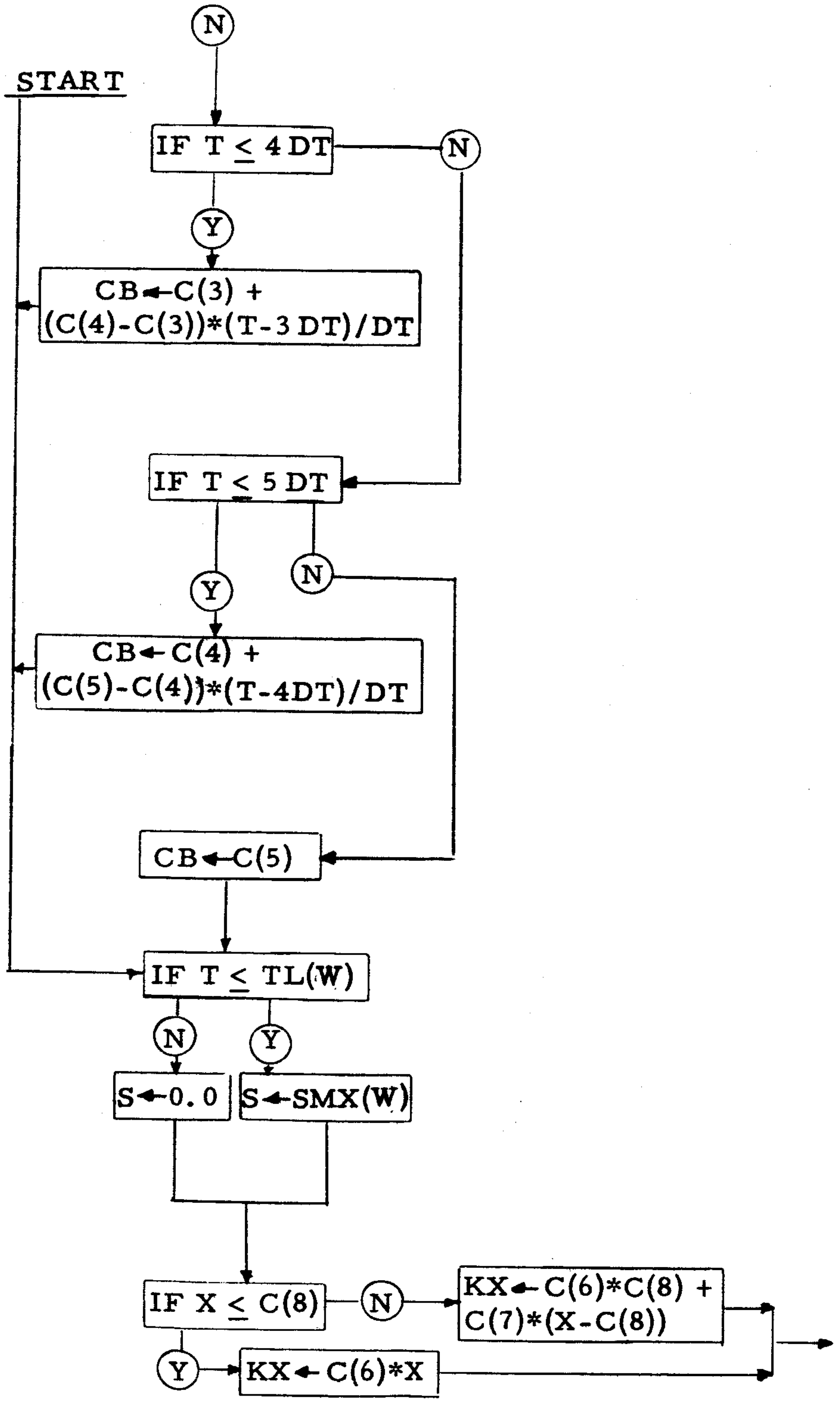
	active load condition and position of maximum acceleration.
GRA	Computes gradient.
INV	Computes inverse of matrix.
LINLEN	Computes L, maximum allowable length of move vector U( ) which will not enter unacceptable region.

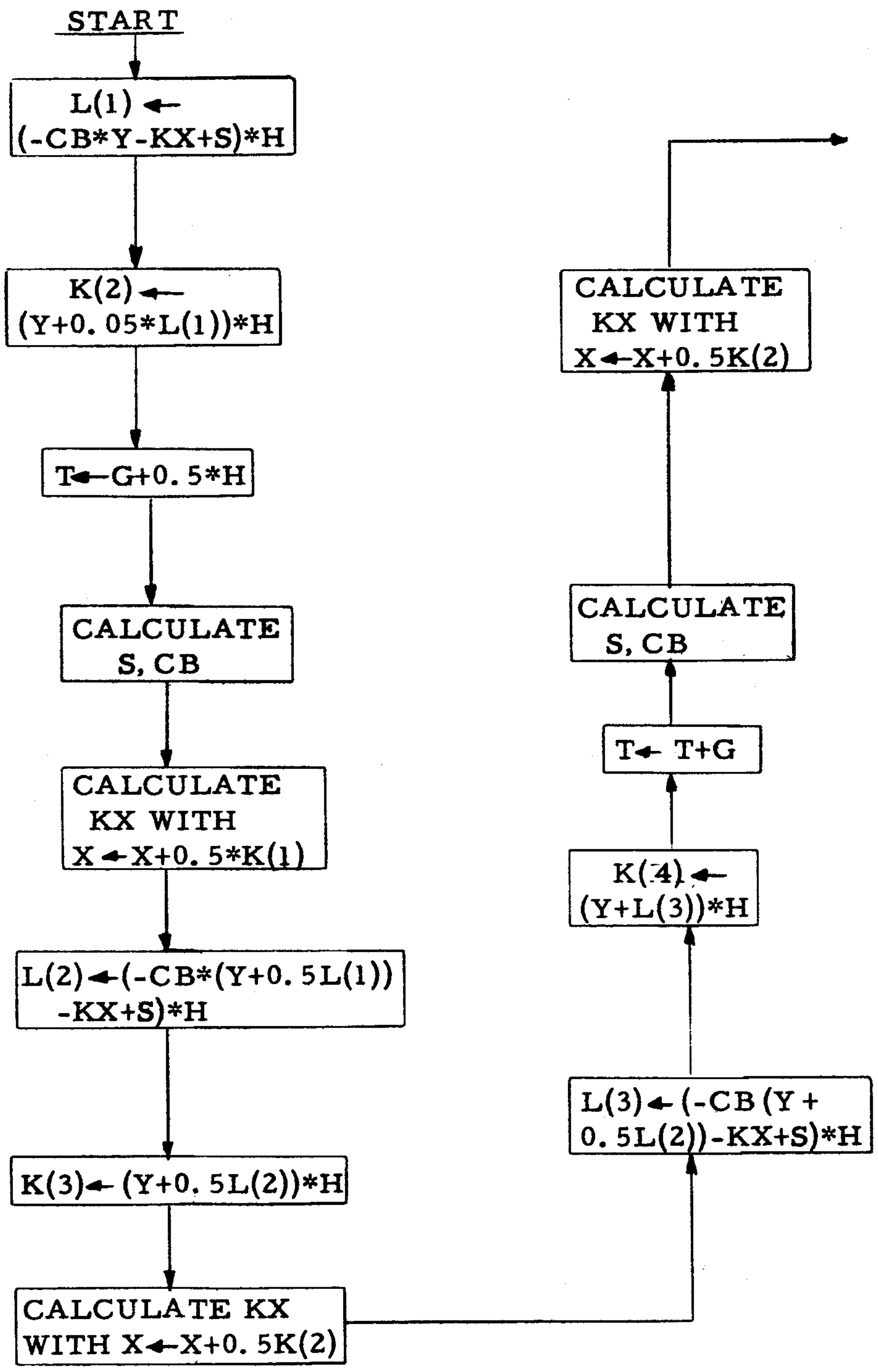
Input for computer program consists of an initial design which is acceptable, initial values for displacement and velocity, a stepsize for the analysis, the time interval between successive damping variables, the variation of each variable used in computing the gradient, absolute value of maximum allowable time rate of change of damping, maximum absolute value of nonlinear constraint function, upper and lower bounds for the variables, tolerances for constraints and convergence test, values of CS8, CS9, and FS11, values of load conditions and respective time durations.

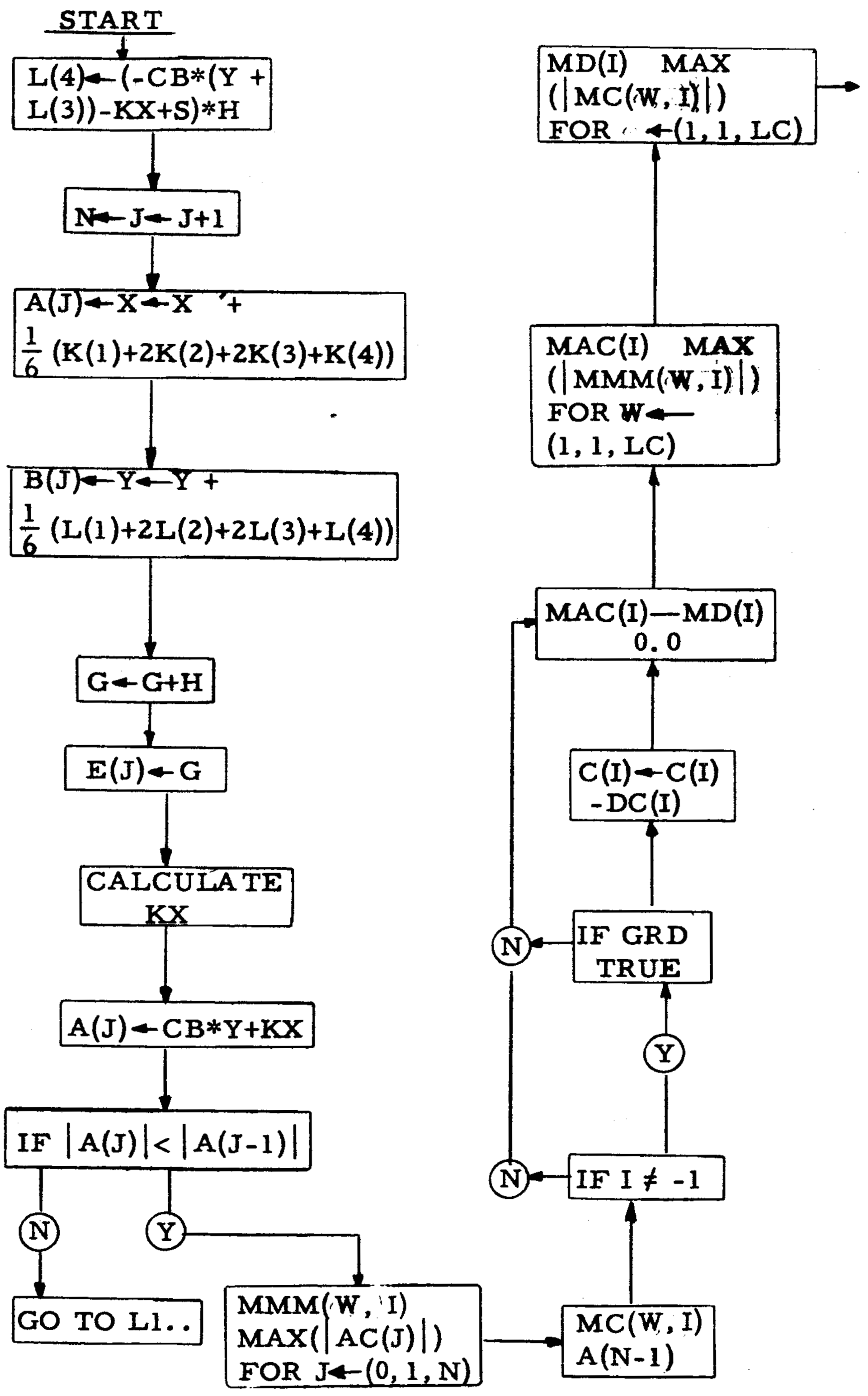
A duplication of the computer program written in ALGOL 60 and run on a UNIVAC 1107 follows with flow chart.

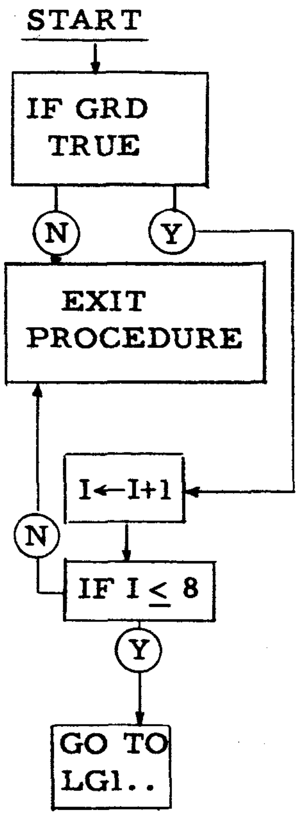
ANALYSIS PROCEDURE; ANL



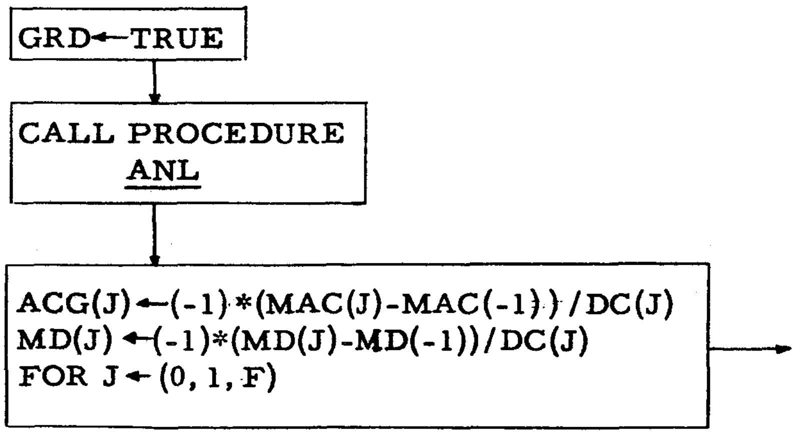

N
START
IF T ≤ 4 DT
N
Y
CB ← C(3) + (C(4)-C(3))*(T-3 DT)/DT
IF T ≤ 5 DT
Y
N
CB ← C(4) + (C(5)-C(4))*(T-4DT)/DT
CB ← C(5)
IF T ≤ TL(W)
N
Y
S ← 0.0
S ← SMX(W)
IF X ≤ C(8)
N
KX ← C(6)*C(8) + C(7)*(X-C(8))
Y
KX ← C(6)*X

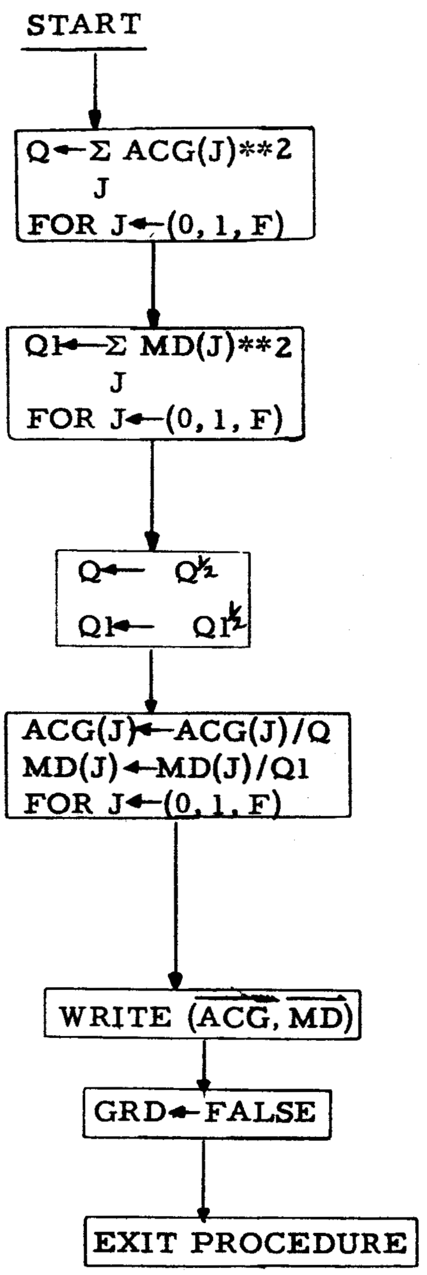

START
L(1) ← (-CB*Y-KX+S)*H
K(2) ← (Y+0. 05*L(1))*H
T← G+0. 5*H
CALCULATE S, CB
CALCULATE KX WITH X ← X+0. 5*K(1)
L(2) ← (-CB*(Y+0. 5L(1)) -KX+S)*H
K(3) ← (Y+0. 5L(2))*H
CALCULATE KX WITH X ← X+0. 5K(2)
L(3) ← (-CB (Y + 0. 5L(2))-KX+S)*H
K(4) ← (Y+L(3))*H
T ← T+G
CALCULATE S, CB
CALCULATE KX WITH X ← X+0. 5K(2)


START
L(4)←(-CB*(Y + L(3))-KX+S)*H
N←J←J+1
A(J)←X←X ʹ+ $\frac{1}{6}$ (K(1)+2K(2)+2K(3)+K(4))
B(J)←Y←Y + $\frac{1}{6}$ (L(1)+2L(2)+2L(3)+L(4))
G←G+H
E(J)←G
CALCULATE KX
A(J)←CB*Y+KX
IF |A(J)| < |A(J-1)|
N
Y
GO TO L1..
MMM(W, I) MAX(|AC(J)|) FOR J←(0, 1, N)
MC(W, I) A(N-1)
IF I ≠ -1
N
Y
IF GRD TRUE
N
C(I)←C(I) -DC(I)
MAC(I)—MD(I) 0.0
MAC(I) MAX (|MMM(W, I)|) FOR W← (1, 1, LC)
MD(I) MAX (|MC(W, I)|) FOR ←(1, 1, LC)

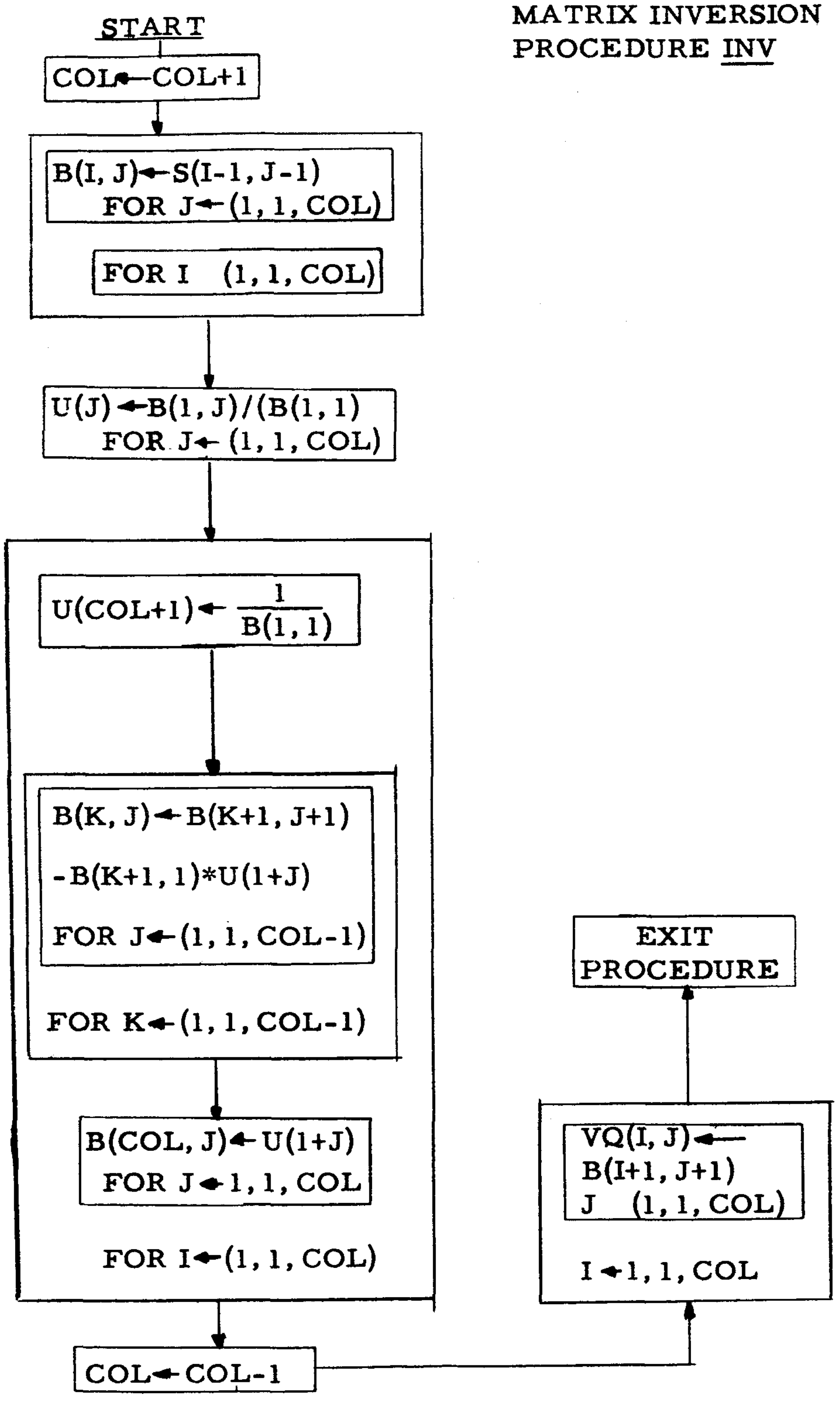


GRADIENT PROCEDURE GRA

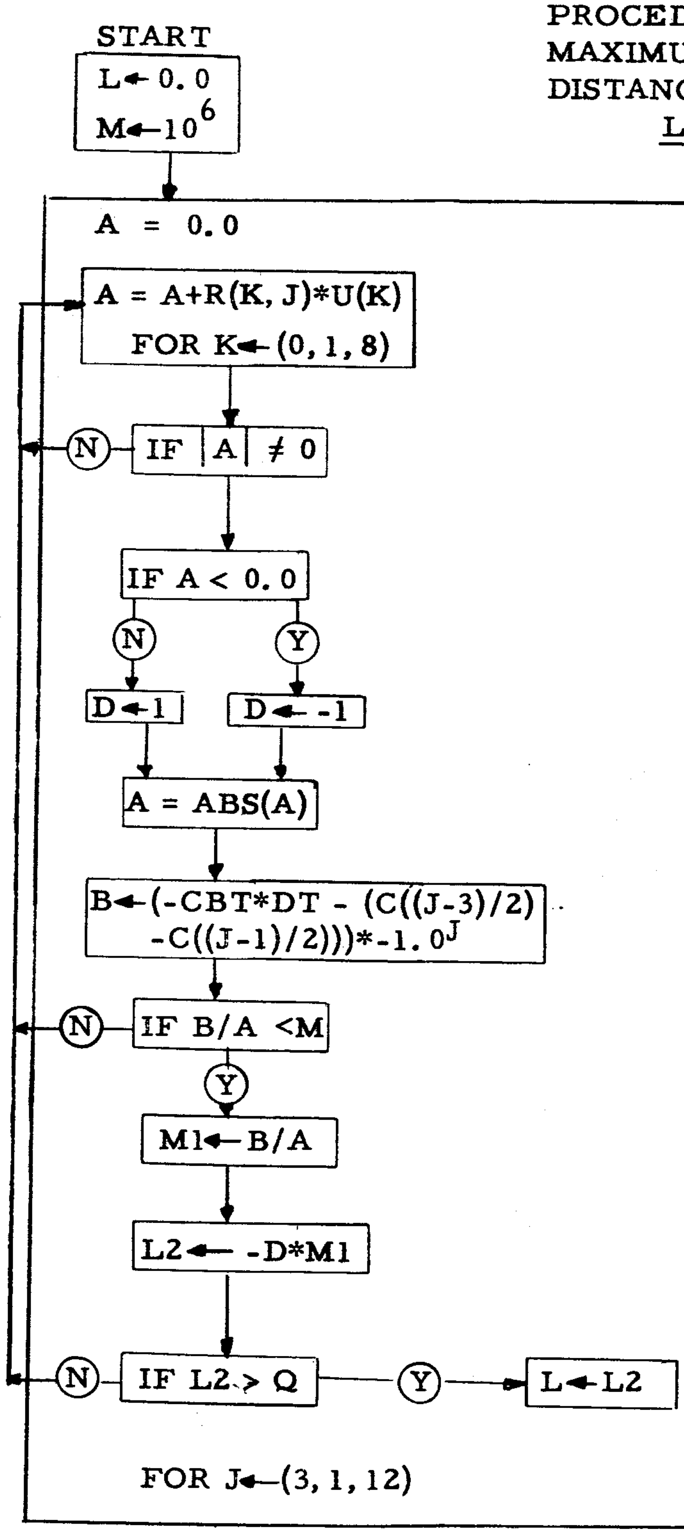

GRD←TRUE
CALL PROCEDURE ANL
ACG(J)←(-1)*(MAC(J)-MAC(-1))/DC(J)
MD(J)←(-1)*(MD(J)-MD(-1))/DC(J)
FOR J←(0, 1, F)

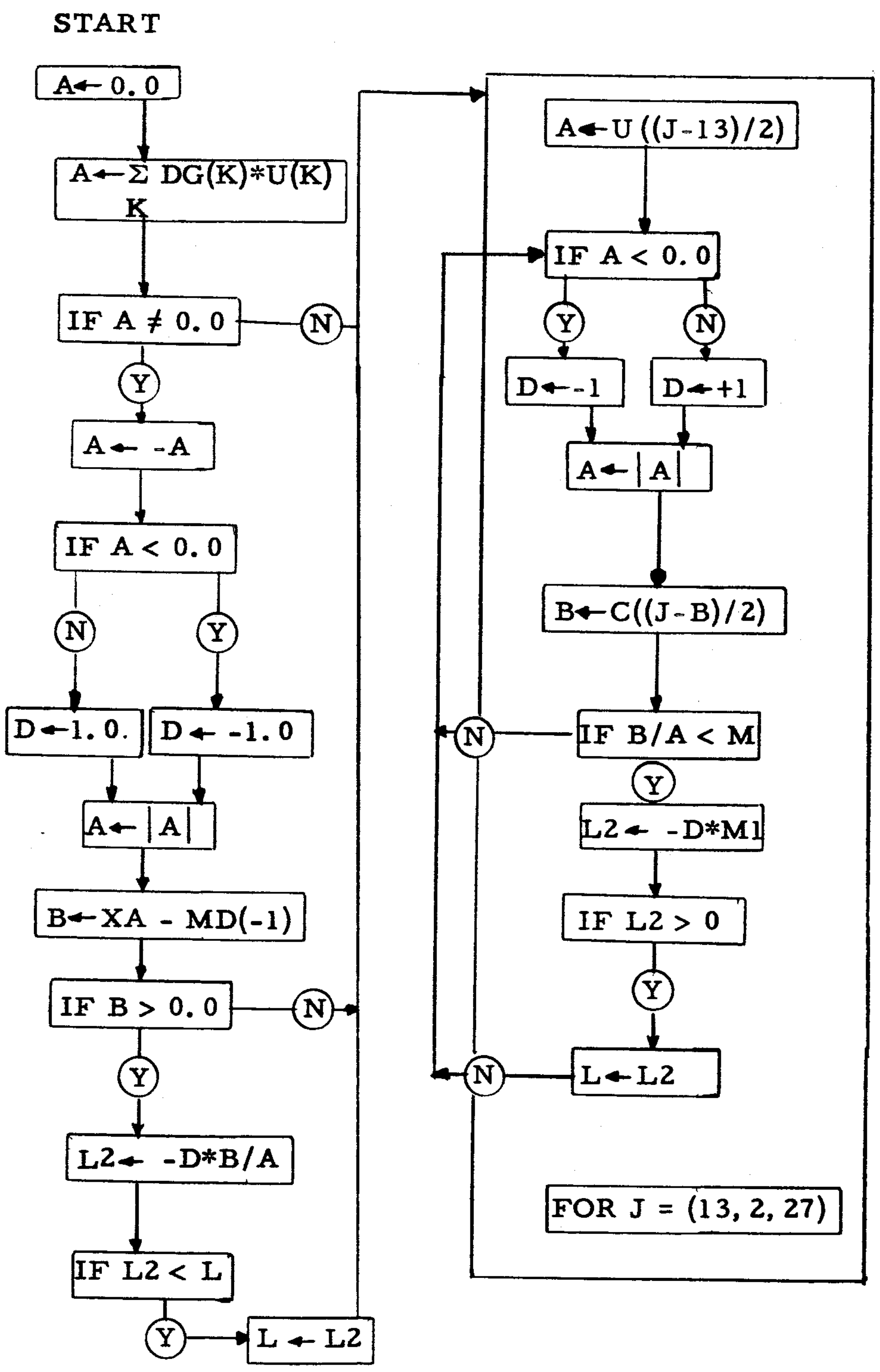

START
Q←Σ ACG(J)**2 J FOR J←(0, 1, F)
Q1←Σ MD(J)**2 J FOR J←(0, 1, F)
Q← Q½
Q1← Q1½
ACG(J)←ACG(J)/Q
MD(J)←MD(J)/Q1
FOR J←(0, 1, F)
WRITE (ACG, MD)
GRD←FALSE
EXIT PROCEDURE

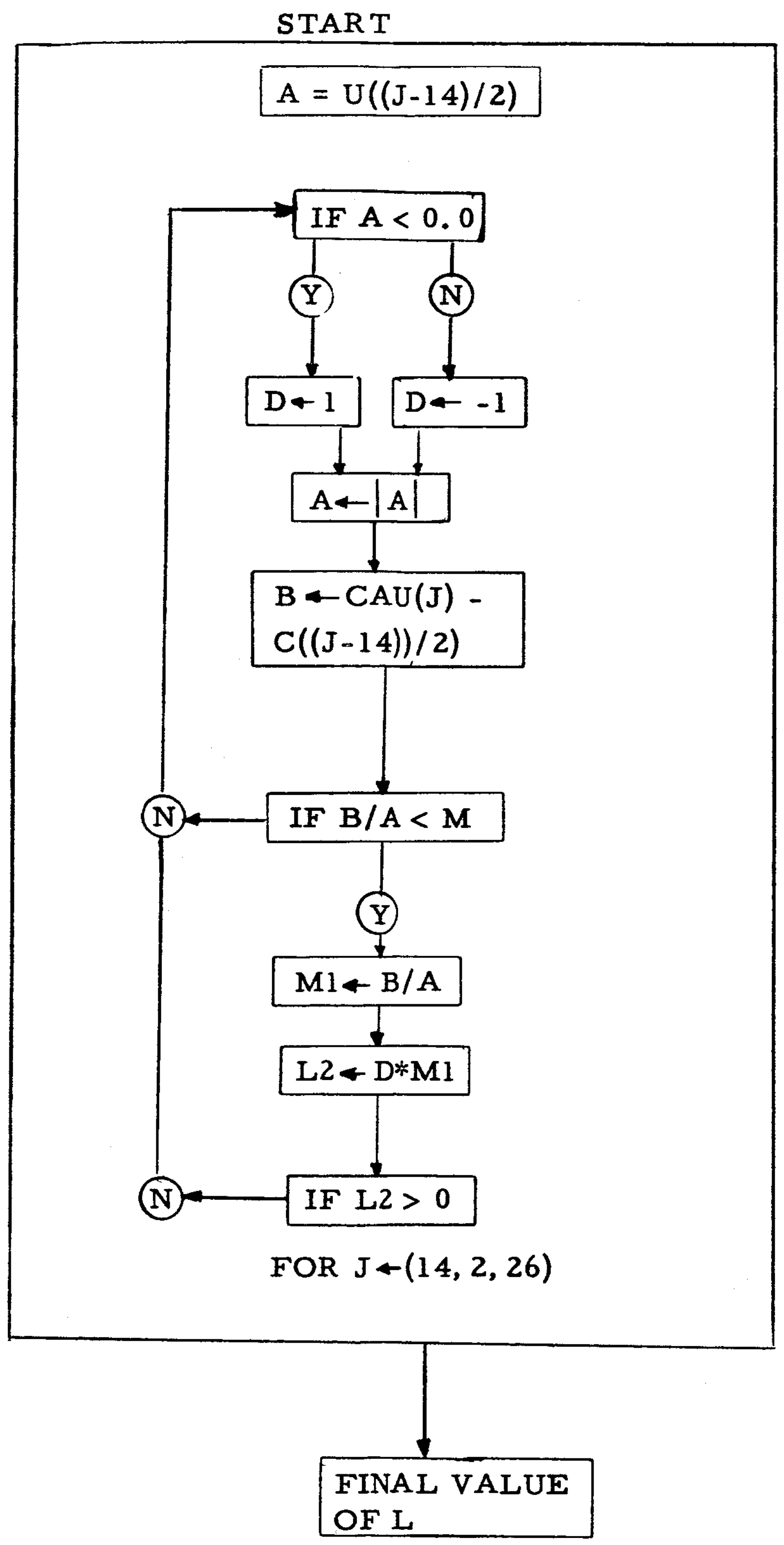
MATRIX INVERSION
PROCEDURE INV

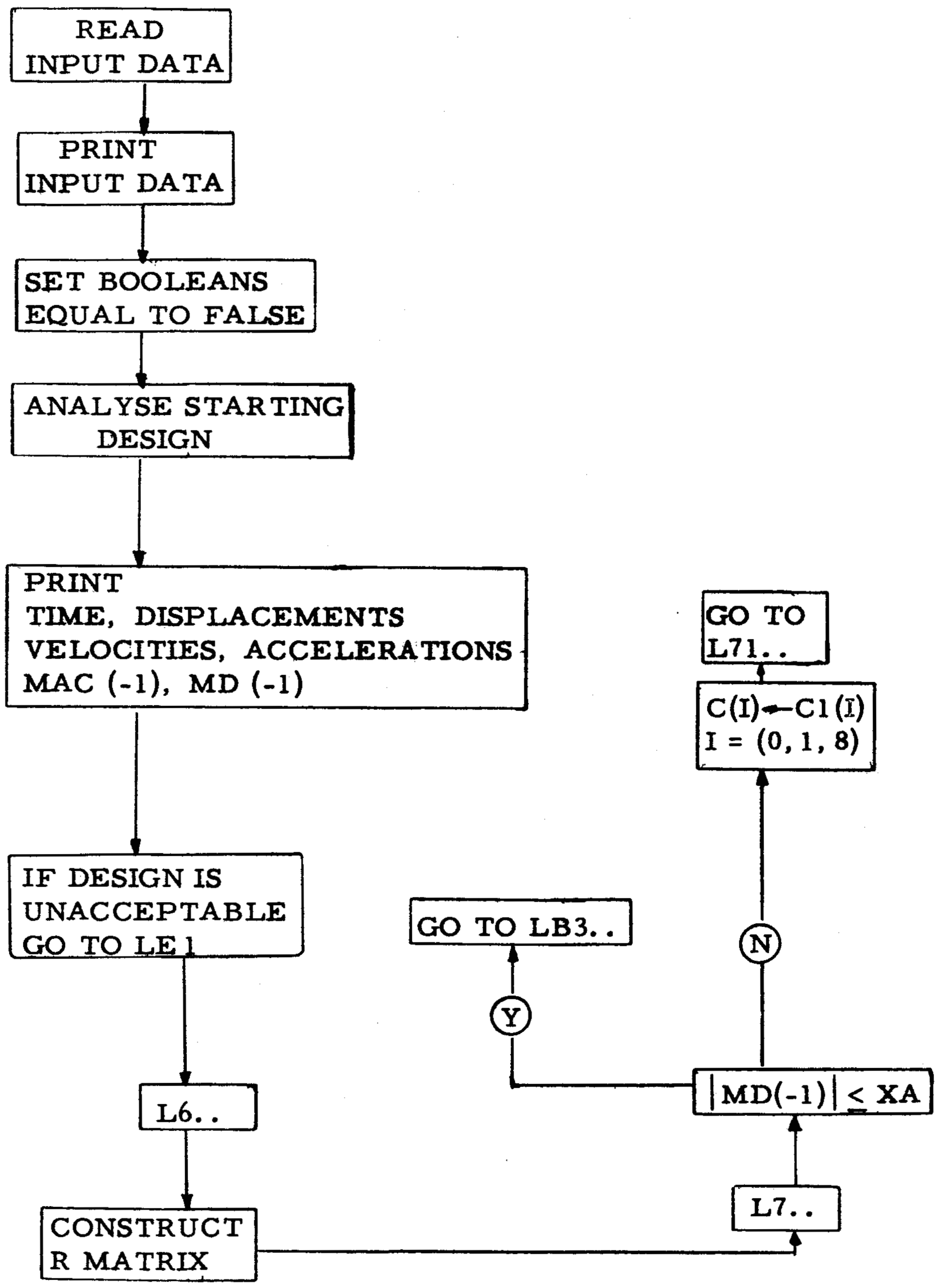


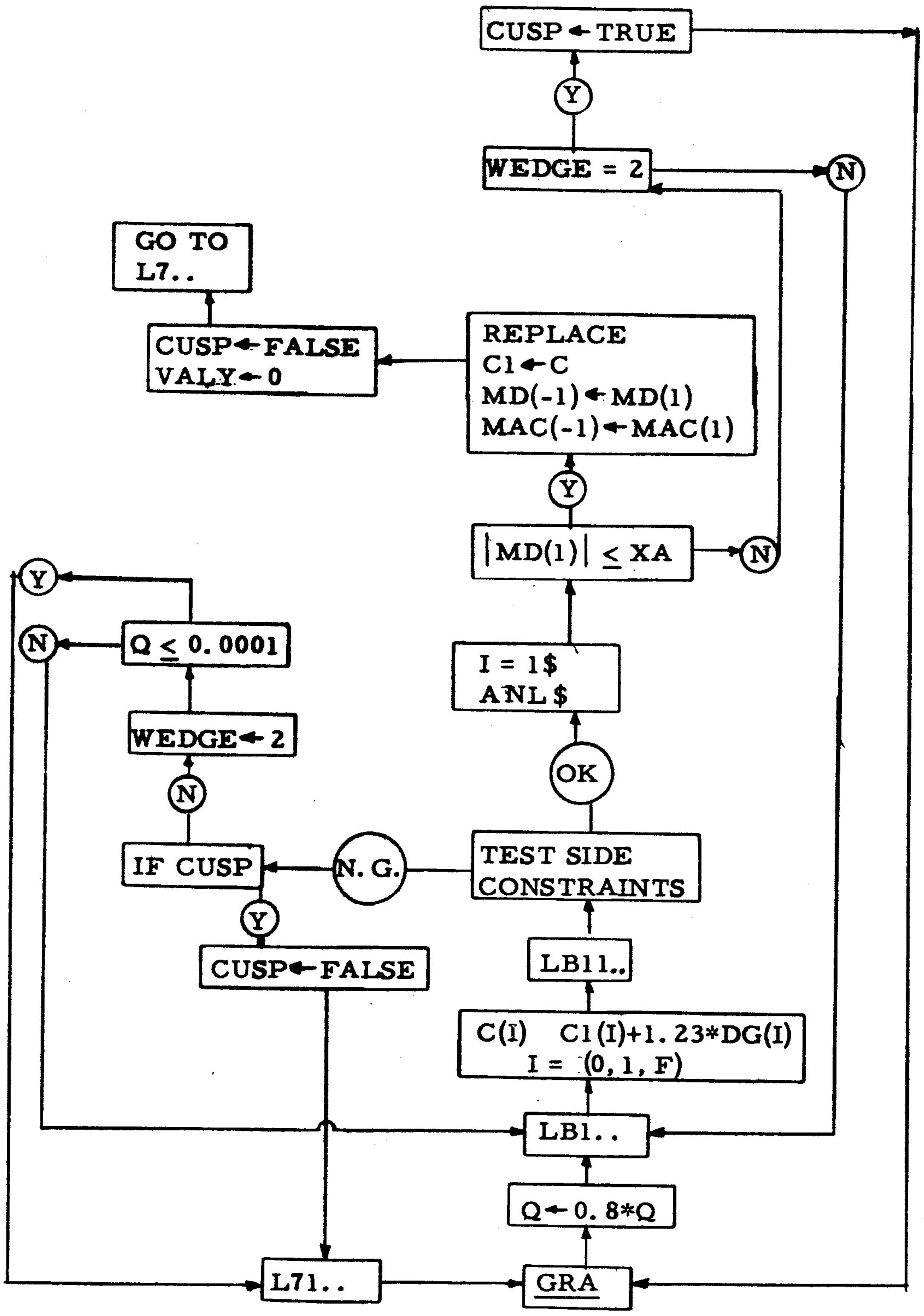
PROCEDURE CALCULATING
MAXIMUM ALLOWABLE
DISTANCE OF TRAVEL
LINLEN

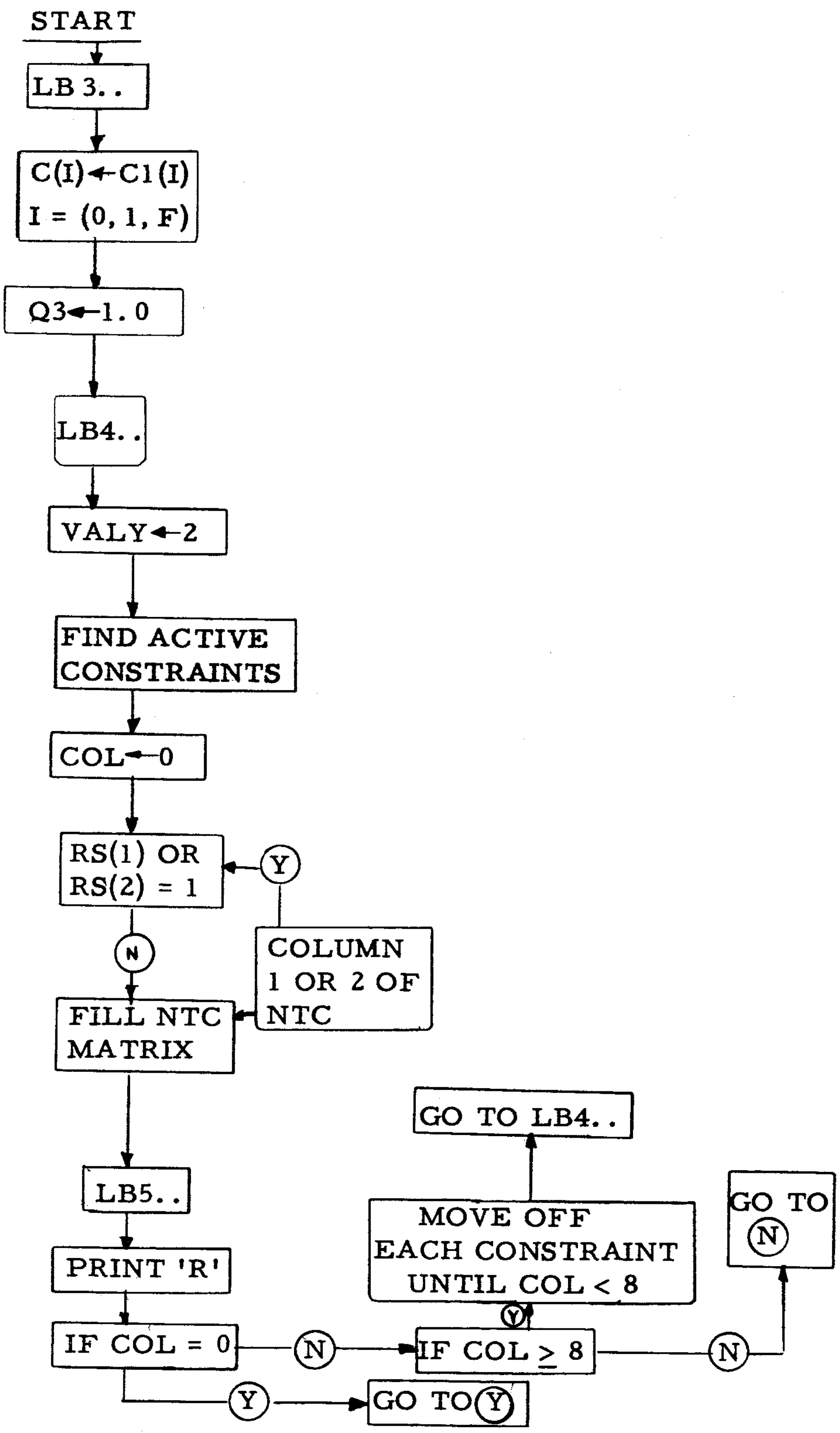


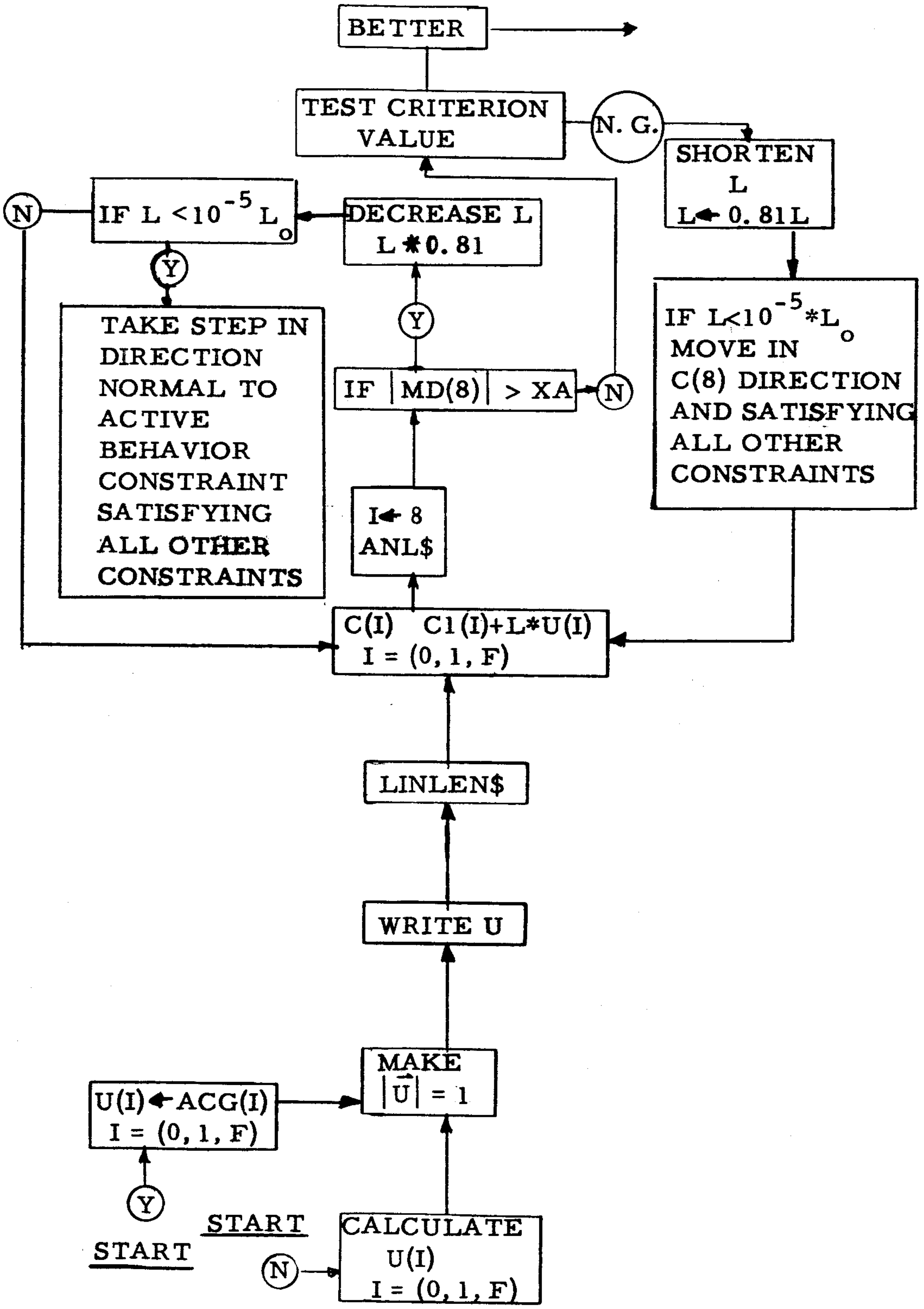

START
A← 0.0
A← Σ DG(K)*U(K) K
IF A ≠ 0.0
N
Y
A← -A
IF A < 0.0
N
Y
D←1.0.
D← -1.0
A← |A|
B←XA - MD(-1)
IF B > 0.0
N
Y
L2← -D*B/A
IF L2 < L
Y
L ← L2
A←U ((J-13)/2)
IF A < 0.0
Y
N
D←-1
D←+1
A← |A|
B←C((J-B)/2)
IF B/A < M
N
Y
L2← -D*M1
IF L2 > 0
Y
L←L2
N
FOR J = (13, 2, 27)

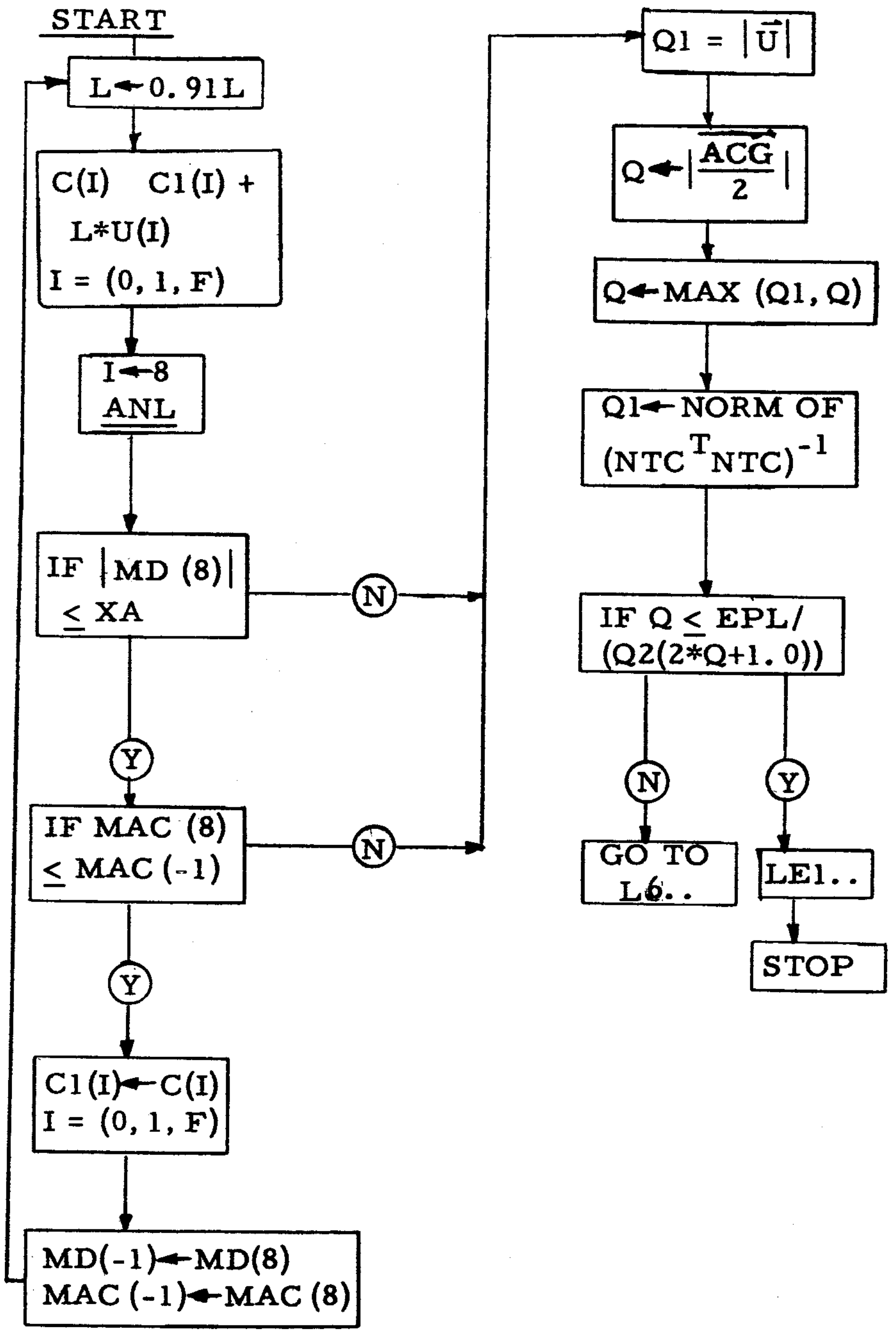

START
A = U((J-14)/2)
IF A < 0.0
Y
N
D ← 1
D ← -1
A ← |A|
B ← CAU(J) - C((J-14))/2)
IF B/A < M
N
Y
M1 ← B/A
L2 ← D*M1
IF L2 > 0
N
FOR J ←(14, 2, 26)
FINAL VALUE OF L


READ INPUT DATA
PRINT INPUT DATA
SET BOOLEANS EQUAL TO FALSE
ANALYSE STARTING DESIGN
PRINT TIME, DISPLACEMENTS VELOCITIES, ACCELERATIONS MAC (-1), MD (-1)
IF DESIGN IS UNACCEPTABLE GO TO LE1
L6..
CONSTRUCT R MATRIX
L7..
|MD(-1)| ≤ XA
Y
GO TO LB3..
N
C(I)←C1(I) I = (0, 1, 8)
GO TO L71..


CUSP ← TRUE
Y
WEDGE = 2
N
GO TO L7..
CUSP ← FALSE
VALY ← 0
REPLACE
C1 ← C
MD(-1) ← MD(1)
MAC(-1) ← MAC(1)
Y
|MD(1)| ≤ XA
N
Y
N
Q ≤ 0.0001
WEDGE ← 2
N
I = 1$
ANL $
OK
TEST SIDE CONSTRAINTS
N. G.
IF CUSP
Y
CUSP ← FALSE
LB11..
C(I) C1(I)+1.23*DG(I)
I = (0, 1, F)
LB1..
Q ← 0.8*Q
L71..
GRA


START
LB 3..
C(I)←C1(I)
I = (0, 1, F)
Q3←1.0
LB4..
VALY←2
FIND ACTIVE CONSTRAINTS
COL←0
RS(1) OR RS(2) = 1
Y
N
COLUMN 1 OR 2 OF NTC
FILL NTC MATRIX
LB5..
PRINT 'R'
IF COL = 0
N
Y
GO TO Y
IF COL ≥ 8
Y
N
MOVE OFF EACH CONSTRAINT UNTIL COL < 8
GO TO LB4..
GO TO N


BETTER
TEST CRITERION VALUE
N. G.
SHORTEN L
L ← 0.81L
N
IF L < 10^-5 L_o
DECREASE L
L * 0.81
Y
TAKE STEP IN DIRECTION NORMAL TO ACTIVE BEHAVIOR CONSTRAINT SATISFYING ALL OTHER CONSTRAINTS
Y
IF |MD(8)| > XA
N
IF L<10^-5 *L_o MOVE IN C(8) DIRECTION AND SATISFYING ALL OTHER CONSTRAINTS
I ← 8
ANL$
C(I) C1(I)+L*U(I)
I = (0, 1, F)
LINLEN$
WRITE U
MAKE |U| = 1
U(I) ← ACG(I)
I = (0, 1, F)
Y
START
START
N
CALCULATE U(I)
I = (0, 1, F)


START
L←0.91L
C(I) C1(I) + L*U(I)
I = (0, 1, F)
I←8
ANL
IF |MD (8)| ≤ XA
N
Y
IF MAC (8) ≤ MAC (-1)
N
Y
C1(I)←C(I)
I = (0, 1, F)
MD(-1)←MD(8)
MAC (-1)←MAC (8)
Q1 = |U⃗|
Q←|ACG/2|
Q←MAX (Q1, Q)
Q1←NORM OF (NTC^T NTC)^-1
IF Q ≤ EPL/(Q2(2*Q+1.0))
N
Y
GO TO L6..
LE1..
STOP

```
MONITOR SYSTEM -- 65KB4 02/17/65
    RUN     MR E RYBICKI                                                                    TIME!     17!55!56   DATE!   19 FEB 65


        ALG     ABC
      ALGOL        DECEMBER 26, 1964        INTERFACE   DECEMBER 23, 1964     PASS2   DECEMBER 23, 1964
            1         INTEGER ARRAY  RS(1..29)       $
BLOCK 1      LEVEL 1                                                                               B0
            2         INTEGER I,N,P,J,WEDGE,VALY,K,COL,F,FS ,CS8,CS9,LC,ALC                   $
            3         BOOLEAN GRD ,ONE,CUSP,EXAM,FO,PK,CDA,PS,PPG,SB1                          $
            4         REAL T,H,XA    ,DS,DT,Q,CBT,E,CBM,EP,L,Q1,Q2,Q3,EPL,MS,XE,KLB,KUB,T1,
            5         FS11                                                                     $
            6         ARRAY C1(0..8),TM,D,V,ACCT(0..500),DC(0..8),  ACGM(-5..0,0..8),
            7         MC,MAC(-1..8) ,VQ(0..6,0..8),CAL,CAU,DC1(0..8),C(0..8),ACG,DG(0..8)   ,
            8         V2(0..8),V3(0..6),V4(0..8), V5(-2..8),DC2(0..8),SMX,TL(1..6),
            9         MTC(0..11,0..29),S1,S(0..15,0..15),R(0..8,1..29),U1,U(0..8)              $
           10         LIST
           11          AL(FOR I=(0,1,N-1) DO (TM(I),D(I),V(I),ACCT(I))),
           12         AL9(FOR I=(0,1,6) DO C1(I)),AL10(D(0),V(0),H,T    ,DT, FOR I=(0,1,8)
           13         DO DC(I),CBT,XA,E,CBM ,EP,EXAM,FO,PK,KLB,KUB,T1,CS8,CS9,CDA,PS,
           14         FOR K=(0,1,8) DO CAL(K),FOR K=(0,1,8) DO CAU(K) ,FS11,EPL             )$
           15         FORMAT GAM(    *(4R16.6,A1),A1,A1),GAM1('VALLEY',A1),GAM2('WEDGE',A1)
           16         ,GAM3(' NO FUTHER MOVE BY SEQUENTIAL STUDY',A1),GAM4('CUSP',A1),
           17         GAM5('NO CONSTRAINTS IN VIOLATION , AM GOING TO SEQUENTIAL STUDY',A1)
           18                                                                                  $
           19         PROCEDURE ANL(T,I,A,B,AC,E)$  VALUE T,I $
BLOCK 2      LEVEL 2
           20         REAL T                                 $ INTEGER I  $ ARRAY A,B,AC,E   $
           21         BEGIN INTEGER J,W,PS ARRAY          K,L(1..4)$ REAL X,Y,G,S,CB,KX  $          B1
           22         ARRAY MC,MM(1..6,-1..8)                                                  $
           23
           24         IF GRD THEN I=0$ LG1..
           25         FOR W=(1,1,LC) DO BEGIN
           26         X=Y=0.0$                                                                         B2
           27         G=0.0$ J=0$
           28         IF NOT GPL THEN GOTO L1$
           29         FOR J =(0,1,8) DO C(J)=C1(J)$ G=0.0$ J=0$
           30         IF I EQL -1 THEN GOTO L1 ELSE C(I)=C(I)+DC(I) $
           31         L1..
           32         IF G GTR 0.06 THEN H=0.01  ELSE H=0.001  $
           33         K(1)=Y*H$
           34         T=G$
           35         IF T LEQ DT THEN CB=C(0)+(C(1)-C(0))*(T/DT) ELSE
           36         IF T LEQ 2*DT THEN CB=C(1)+(C(2)-C(1))*(T-DT)/DT ELSE
           37         IF T LEQ 3*DT THEN CB = C(2)+(C(3)-C(2))*(T-2*DT)/DT ELSE
           38         IF T LEQ 4*DT THEN CB = C(3)+(C(4)-C(3))*(T-3*DT)/DT  ELSE
           39         IF T LEQ 5*DT THEN CB = C(4)+(C(5)-C(4))*(T-4*DT)/DT  ELSE
           40         CB=C(5)     $
           41         IF T LEQ TL(W) THEN S=SMX(W) ELSE S=0.0$
           42         IF X LEQ C(8)    THEN KX=C(6)*X ELSE KX=C(6)*C(8) +C(7)*(X-C(8))$
           43         L(1)=(-CB*Y-KX+S)*H$
           44         K(2)=(Y+0.5*L(1))*H   $
      T=G+0.5*H$
      IF T LEQ DT THEN CB=C(0)+(C(1)-C(0))*(T/DT) ELSE
      IF T LEQ 2*DT THEN CB=C(1)+(C(2)-C(1))*(T-DT)/DT ELSE
      IF T LEQ 3*DT THEN CB = C(2)+(C(3)-C(2))*(T-2*DT)/DT ELSE
      IF T LEQ 4*DT THEN CB = C(3)+(C(4)-C(3))*(T-3*DT)/DT  ELSE
      IF T LEQ 5*DT THEN CB = C(4)+(C(5)-C(4))*(T-4*DT)/DT  ELSE
      CB=C(5)     $
      IF T LEQ TL(W) THEN S=SMX(W) ELSE S=0.0$
      IF X+0.5*K(1) LEQ C(8)
                     THEN KX=C(6)*(X+0.5*K(1))  ELSE KX=C(6)*C(8) +
      C(7)*(X+0.5*K(1)-C(8) )$
      L(2)=(-CB*(Y+0.5*L(1))-KX+S)*H$
      K(3)=(Y+0.5*L(2))*H $
      IF X+0.5*K(2) LEQ C(8)
                     THEN KX=C(6)*(X+0.5*K(2))  ELSE KX=C(6)*C(8) +
      C(7)*(X+0.5*K(2)-C(8) )$
      L(3)=(-CB*(Y+0.5*L(2))-KX+S)*H$
      K(4)=(Y+L(3))*H $
      T=G+H$
      IF T LEQ DT THEN CB=C(0)+(C(1)-C(0))*(T/DT) ELSE
      IF T LEQ 2*DT THEN CB=C(1)+(C(2)-C(1))*(T-DT)/DT ELSE
      IF T LEQ 3*DT THEN CB = C(2)+(C(3)-C(2))*(T-2*DT)/DT ELSE
      IF T LEQ 4*DT THEN CB = C(3)+(C(4)-C(3))*(T-3*DT)/DT  ELSE
      IF T LEQ 5*DT THEN CB = C(4)+(C(5)-C(4))*(T-4*DT)/DT  ELSE
      CB=C(5)     $
      IF T LEQ TL(W) THEN S=SMX(W) ELSE S=0.0$
      IF X+0.5*K(3) LEQ C(8)
                     THEN KX=C(6)*(X+0.5*K(3))  ELSE KX=C(6)*C(8) +
      C(7)*(X+0.5*K(3)-C(8) )$
      L(4)=(-CB*(Y+L(3))-KX+S)*H     $
      N=J=J+1$
     IF N GEQ 499 THEN BEGIN WRITE(GAM,N,AL)$WRITE(AL9)$ WRITE(AL10)$             B3
      END$                                                                        E3
      A(J)=X=X+(1.0/6.0)*(K(1)+2*K(2)+2*K(3)+K(4)) $
      B(J)=Y=Y+(1.0/6.0)*(L(1)+2*L(2)+2*L(3)+L(4)) $
      G=G+H$ E(J)=G$
      IF X LEQ C(8) THEN KX=C(6)*X ELSE KX=C(6)*C(8)+C(7)*(X-C(8))$
      AC(J)=CB*Y+KX$
                             IF ABS(A(J)) LSS ABS(A(J-1)) THEN
      GOTO L4$ GOTO L1$
      L4.. MMM(W,I)=0.0$ FOR J=(0,1,N) DO IF ABS(AC(J)) GTR MMM(W,I) THEN
      BEGIN
      MMM(W,I)=ABS(AC(J))$  XE=A(J) END$                                          B4   E4
      MC(W,I)=A(N-1)$
      IF I NEQ -1 THEN
      IF GRD THEN C(I)=C(I)-DC(I) $
      END$                                                                        E2
      MAC(I)=MD(I)=0.0$
      FOR W=(1,1,LC) DO BEGIN
      IF MMM(W,I) GTR MAC(I) THEN   BEGIN MAC(I)=MMM(W,I)$ ALC=W$ END$            B5   B6   E6
      IF MC(W,I)  GTR MD(I)  THEN          MD(I)=MC(W,I)$
      END$                                                                        E5
      IF GRD THEN BEGIN  I=I+1$ IF I LEQ 6 THEN GOTO LG1$ END$                    B7   E7
      END ANL $                                                                   E1   C
```

```
  END BLOCK 2
         99
        100        PROCEDURE GRA$
BLOCK 3    LEVEL 2
        101        BEGIN INTEGER J$ REAL L$                                                        B8
        102        GRD=TRUE$
        103                    ANL(T,I,D ,V ,ACCT ,TM )=
        104        FOR J=(0,1,F) DO BEGIN
        105        ACG(J)=(-1)*(MAC(J)-MAC(-1))/DC(J) $                                           B9
        106        DG(J)=(-1)*(MD(J)-MD(-1))/DC(J) $ END$                                         E9
        107        IF F EQL 7 THEN ACG(8)=DG(8)=0.0$
        108        Q=0.0$ Q1=0.0$ FOR J=(0,1,F) DO BEGIN
        109        Q=Q+ACG(J)*ACG(J)$ Q1=Q1+DG(J)*DG(J) $END$                                     B10    E10
        110        Q=SQRT(Q)$ Q1=SQRT(Q1)$
        111        Q2=Q$
        112        FOR J=(0,1,F) DO BEGIN ACG(J)=ACG(J)/Q$ DG(J)=DG(J)/Q1$ END$                   B11    E11
        113        IF CDA THEN FOR K=(0,1,F) DO ACG(K)=DG(K)$
        114            WRITE('DG',DG)$ WRITE('ACG',ACG)$
        115        GRD=FALSE$
        116        END GRA $                                                                       E6     C
  END BLOCK 3
        117
        118        PROCEDURE INV(S,COL)$ ARRAY S$ INTEGER COL$
BLOCK 4    LEVEL 2
        119        BEGIN ARRAY B(1..30,1..30), U(1..30)        $ COL=COL+1$                        B12
        120        FOR I=(1,1,COL) DO FOR J=(1,1,COL) DO B(I,J)=S(I-1,J-1)$
        121        FOR I=(1,1,COL) DO BEGIN
        122        FOR J=(1,1,COL) DO U(J)=B(1,J)/B(1,1)$                                        B13
        123        U(COL+1)=(1.0)/B(1,1)$ FOR K=(1,1,COL-1) DO
        124       BEGIN FOR J=(1,1,COL-1)DO B(K,J)=B(K+1,J+1)-B(K+1,1)*U(1+J)$                   B14
        125       B(K,COL)=-B(K+1,1)*U(COL+1) END$                                                E14
        126       FOR J=(1,1,COL)DO B(COL,J)=U(1+J) END$                                          E13
        127        COL=COL-1$ FOR I=(0,1,COL) DO FOR J=(0,1,COL) DO VQ(I,J)=B(I+1,J+1)$
        128        END INV     $                                                                   E12    C
  END BLOCK 4
        129
        130        PROCEDURE LINLEN $
BLOCK 5    LEVEL 2
        131        BEGIN INTEGER I,J,J1,K,L1 $ REAL A,B,D,D1,M,L2,M1$                              B15
        132        L=0.0$ M=(10.0**6.0) $
        133        FOR J=(3,2,11) DO BEGIN A=0.0$                                                  B16
        134        FOR K=(0,1,8) DO A=A+R(K,J)*U(K) $
        135        IF ABS(A) NEQ 0.0 THEN BEGIN
        136        IF A LSS 0.0 THEN D=-1.0 ELSE D= 1.0 $                                          B17
        137        A=ABS(A)$
        138        B= (-CBT*UT       -(C((J-3)/2)-C((J-1)/2)))*(-1.0) $
        139        IF B GTR 0.0 THEN
        140        IF B/A LSS M THEN
        141        BEGIN M1=B/A$ L2=-D*M1$IF L2 GTR 0.0 THEN BEGIN L=L2$ M=M1$ END$ END$         B18    B19    E19    E18
        142        END$ END$                                                                       E17    E16
        143        FOR J=(4,2,12) DO BEGIN A=0.0$                                                  B20
        144        FOR K=(0,1,8) DO A=A+R(K,J)*U(K) $
        145        IF ABS(A) NEQ 0.0 THEN BEGIN
        146        IF A LSS 0.0 THEN D=-1.0 ELSE D= 1.0 $                                          B21
```

```
A=ABS(A)$
B= ( CBT*LT      -(C((J-4)/2)-C((J-2)/2)))$
IF B GTR 0.0 THEN
IF B/A LSS M THEN
BEGIN M1=B/A$ L2=-D*M1$IF L2 GTR 0.0 THEN BEGIN L=L2$ M=M1$ END$ END$      B22   B23   E23   E22
END$ END$                                                                  E21   E20
FOR  J=(13,2,27) DO  BEGIN A=U((J-13)/2) $                                 B24
IF A NEQ 0.0 THEN BEGIN IF A LSS 0.0                                       B25
THEN D=-1.0 ELSE D=1.0 $
A=ABS(A)$
B=((C((J-13)/2) ))$
IF B GTR 0.0 THEN
IF B/A LSS M THEN
BEGIN M1=B/A$ L2=-D*M1$IF L2 GTR 0.0 THEN BEGIN L=L2$ M=M1$ END$ END$      B26   B27   E27   E26
END END$                                                                   E25   E24
FOR J=(14,2,24) DO BEGIN A=U((J-14)/2) $                                   B28
IF A NEQ 0.0 THEN BEGIN
IF A LSS 0.0 THEN D=-1.0 ELSE D=1.0$                                       B29
A=ABS(A)$
B=(CBM   -C((J-14)/2))       $
IF B GTR 0.0 THEN
IF B/A LSS M THEN
BEGIN M1=B/A$ L2= D*M1$IF L2 GTR 0.0 THEN BEGIN L=L2$ M=M1$ END$ END$      B30   B31   E31   E30
END$ END$                                                                  E29   E28
J=26$     IF F EQL 7 THEN GOTO LL1$
A=U(6)$ IF A NEQ 0.0 THEN BEGIN
IF A LSS 0.0 THEN D=-1.0 ELSE D=1.0$  A=ABS(A)$                            B32
B=XA   -C(8)$ IF B/A LSS M THEN
IF B GTR 0.0 THEN
BEGIN M1=B/A$ L2= D*M1$IF L2 GTR 0.0 THEN BEGIN L=L2$ M=M1$ END$ END$      B33   B34   E34   E33
END$                                                                       E32
A=U(6)$ IF A NEQ 0.0 THEN BEGIN
IF A LSS 0.0 THEN D=-1.0 ELSE D=1.0$  A=ABS(A)$                            B35
B=C(6)$ IF B GTR 0.0 THEN IF B/A LSS M THEN
BEGIN M1=B/A$ L2=-D*M1$IF L2 GTR 0.0 THEN BEGIN L=L2$ M=M1$ END$ END$      B36   B37   E37   E36
END$                                                                       E35
LL1..
IF ML(-1) GTR 0.99*XA THEN BEGIN                                           B38
IF NOT LDA THEN BEGIN                                                      B39
A=0.0$ FOR  K=(0,1,8) DO A=A+UG(K)*U(K)$
IF A NEQ 0.0 THEN BEGIN                                                    B40
A=-A$
IF A LSS 0.0 THEN D=-1.0 ELSE D=1.0$ A=ABS(A)$
B=XA -ML(-1) $
IF B GTR 0.0 THEN
IF B/A LSS M THEN
BEGIN M1=B/A$ L2= D*M1$IF L2 GTR 0.0 THEN BEGIN L=L2$ M=M1$ END$ END$      B41   B42   E42   E41
D=-D$
B= XA +MC(-1) $
IF B GTR 0.0 THEN
IF B/A LSS M THEN
BEGIN M1=B/A$ L2=-D*M1$IF L2 GTR 0.0 THEN BEGIN L=L2$ M=M1$ END$ END$      B43   B44   E44   E43
END$ END$                                                                  E40   E39
END$                                                                       E38
```

```
        WRITE(L,'EXIT LINLEN',L)$
        L=0.95*L$
        IF L GTR 50.0 THEN L=50.0$
        END LINLEN $                                                          E15   C
END BLOCK 5

        READ(AL9)$ READ(AL10)$
        WRITE(AL9)$ WRITE(AL10)$
        READ(LC)$ WRITE(LC)$
        READ( FOR J=(1,1,LC) DO (SMX(J),TL(J)))$
        WRITE(FOR J=(1,1,LC) DO (SMX(J),TL(J)))$
                        VS(-1)=10.0**10$
        FOR J=(0,1,6) DO C(J)=C1(J)$
        Q=1.0$
        F=8$
        PFG=FALSE$
        CHD= FALSE$
        SE1=FALSE$
        I=-1$ANL(T,I,U,V,ACCT,TM) $
        FOR K=(0,1,7) DO DC2(K)=0.25$
        DC1(6)=DC1(7)=10.0$FOR J=(0,1,5) DO DC1(J)=2.0$   DC1(8)=0.05    $
        WRITE(GAM,K,AL)$
        WRITE(MD(-1),MAC(-1))$
        IF MD(-1) GEQ XA   THEN GOTO LE1$
        L10..
        ONE=FALSE$
        CUSP=FALSE$
        WEDGE=VALY=0$
        MS=((10.0)**6)$
        L6..
        WRITE('L6')$
        FOR K=(0,1,8) DO DC1(K)=0.1 $
        FOR I=(0,1,6) DO FOR J=(1,1,29) DO R(I,J)=0.0$
        FOR I=(0,1,4) DO FOR J=(2*I+3,1,2*I+4) DO BEGIN
        R(I,J)=1.0$ R(I+1,J)=-1.0$ END$                                       B45   E45
        FOR I=(0,1,5) DO FOR J=(2*I+13,1,2*I+14) DO R(I,J)=1.0$
        R(6,25)=R(7,27)=R(8,29)=1.0$
        FOR I=(0,1,5) DO FOR J=( 4,2,12) DO R(I,J)=-R(I,J)$
        L7..
        WRITE('L7')$
        IF C1(8) LEQ 0.00001 THEN C1(8)=0.0$
        IF ABS(MD(-1)) LEQ XA THEN GOTO LB3$
        L70..   FOR J=(0,1,6) DO C(J)=C1(J)$
        L71..
        WRITE('L71')$
        ONE=CUSP=FALSE$ WEDGE=VALY=0$
        GRA$
        Q=1.0$
        LE1..
        WRITE('LE1')$
        FOR K=(0,1,F) DO  C(K)=C1(K)+1.23*Q*DC(K)$
        LB11..
        WRITE('LB11')$
        FOR I=(0,1,8) DO
```

```
IF C(I) LSS 0.0 THEN GOTO LB2$
 FOR I=(0,1,4) DO BEGIN IF ABS(C(I)-C(I+1)) GTR CBT*DT                  B46
 THEN GOTO LB2$ IF C(I) GTR CBM THEN GOTO LB2$ END$                     E46
 IF C(5) GTR CBM THEN GOTO LB2$
 I=1$ T=0.0$ ANL(T,I,C ,V ,ACCT ,TM )$
 IF ABS(MD(1)) LEQ XA THEN BEGIN IF MAC(1) LSS MAC(-1)                  B47
 THEN BEGIN FOR J=(0,1,8) DO C1(J)= C(J)$MD(-1)=MD(1)$ MAC(-1)=MAC(1)$  B48
 GOTO L7 END ELSE GOTO LB4 END$                                         E48   E47
 IF WEDGE EQL 2 THEN BEGIN CUSP=TRUE$ WRITE('CUSP BY MOD GRAD.')$       B49
 GOTO L71 END ELSE BEGIN Q=1.11*Q$ GOTO LB1 END$                        E49   B50   E50
 LB2.. IF CUSP THEN BEGIN  CUSP = FALSE$ GOTO L71$ END$                 B51   E51
 WEDGE=2$ Q=0.02*Q$
 WRITE('LB2')$
 IF Q LSS 0.001 THEN BEGIN CUSP=TRUE$                                   B52
 WRITE(O,GAM4)$ GOTO LB3$ END ELSE                                      E52
 GOTO LB1$
 LB3..
 WRITE('LB3')$
 FOR I=(0,1,F) DO C(I)=C1(I)$
 Q3=1.0$
 LB4..
 VALY=2$
 FOR J=(1,1,29) DO RS(J)=0$
 FS=FS+1$ IF FS GTR CS9 THEN FS=1$ IF FS LSS 1 THEN FS=1$
 IF FS LEQ CS8 THEN F=8 ELSE F=7$
 FOR I=(0,1,4) DO
 IF ABS(C(I+1)-C(I)) GEQ CBT*DT -DC1(I)-EP
 THEN BEGIN IF C(I)-C(I+1)      GEQ CBT*DT -DC1(I)-EP                   B53
 THEN RS(4+2*I)=1 ELSE RS(3+2*I)=1 END$                                 E53
 FOR I=(0,1,5) DO BEGIN
 IF C(I) GTR CBM - DC1(I)-EP THEN                                       B54
 RS(14+2*I)=1$        IF C(I) LSS DC1(I)+EP THEN
 RS(13+2*I)=1$ END$                                                     E54
 IF ABS(MD(-1)) GTR XA-EP THEN BEGIN
 IF MD(-1) GTR XA-EP THEN RS(2)=1 ELSE RS(1)=1 END$                     B55   E55
 CCL = 0$
 Q=0.0$ FOR K=(0,1,F) DO Q=Q+ACG(K)*CG(K)$ IF Q LSS
 -0.999 THEN BEGIN WRITE('GRD. NOR. TO CNST.')$  END$                   B56   E56
 IF CCA THEN RS(1)=RS(2)=0$
 GRA$
 FOR J=(-2,1,-1) DO
                    FOR K=(0,1,8) DO ACGR(J,K)=ACGR(J+1,K)$
 FOR K=(0,1,8) DO ACGR(0,K)=ACG(K)$
 PPG=FALSE$
 Q=0.0$ FOR K=(0,1,8) DO Q=Q+ACGR(-1,K)*ACGR(0,K)$
 IF Q LSS 0.707 THEN BEGIN
 LPR1..                                                                 B57
 PPG=TRUE$ FOR K=(0,1,8) DO ACG(K)=ACGR(0,K)-Q*ACGR(-1,K) END$          E57
 LEPR..
 FOR J=(3,1,29) DO IF RS(J) EQL 1 THEN BEGIN Q3=0.0$ FOR K=(0,1,8) DO    B58
 Q3=Q3+ ACG(K)*R(K,J)$ IF Q3 GTR 0.0 THEN RS(J)=2$ END$                 E58
 FOR J=(1,1,2) DO IF RS(J) EQL 1 THEN
 BEGIN      FOR I=(0,1,8) DO NTC(I,0)=UG(I)$                            B59
 CCL=1$ END$                                                            E59
```

```
FOR J=(3,1,29) DO IF RS(J) EQL 1 THEN
BEGIN FOR I=(0,1,F) DO NTC(I,COL)=R(I,J)$                                 B60
                    COL=COL+1$ ENDS                                        E60
LB5..
WRITE('RS',RS)$
IF COL EQL 0 THEN BEGIN  FOR K=(0,1,F) DO U(K)=ACG(K)$                    B61
GOTO LB52 ENDS                                                             E61
COL =COL-1$
IF COL GEQ F THEN BEGIN IF NOT ONE THEN BEGIN                              B62
FOR J=(0,1,F) DO BEGIN V4(J)=C(J)$ DC1(J)=DC1(J)/2.0$ ENDS                 B63   B64   E64
ONE=TRUE$ ENDS                                                             E63
Q3=1.2*Q3$
FOR J=(0,1,F) DO C(J)=C(J)+Q3*ACG(J)$ GOTO LB4 ENDS                        E62
IF ONE THEN FOR J=(0,1,F) DO C(J)=C1(J)$
LB51..
WRITE('LB51')$
FOR J=(0,1,COL) DO FOR I=(0,1,COL) DO  BEGIN S(J,I)=0.0$                   B65
FOR K=(0,1,F) DO
S(J,I)=S(J,I)+NTC(K,J)*NTC(K,I)$ ENDS                                      E65
WRITE(FOR I=(0,1,COL) DO FOR J=(0,1,COL) DO S(I,J) )$
INV(S,COL)    $
FOR I=(0,1,F) DO FOR J=(0,1,COL) DO BEGIN S1(I,J)=0.0$                     B66
FOR K=(0,1,COL) DO
S1(I,J)=S1(I,J)+NTC(I,K)*VQ(K,J)$ ENDS                                     E66
FOR I=(0,1,F)DO FOR J=(0,1,F) DO  BEGIN S(I,J)=0.0$                        B67
FOR K=(0,1,COL) DO S(I,J)=S(I,J)+ S1(I,K)*NTC(J,K)$ ENDS                   E67
FOR I=(0,1,F) DO BEGIN U1(I)=0.0$                                          B68
FOR J=(0,1,F) DO
U1(I)=U1(I)+S(I,J)*ACG(J)$ ENDS                                            E68
FOR I=(0,1,F) DO U(I)=ACG(I)-U1(I)$
LB52..
FOR K=(0,1,5) DO BEGIN IF C(K) LSS 0.01 THEN IF U(K) LSS 0.0               B69
THEN U(K)=C(K)=0.0$ ENDS                                                   E69
          FOR I=(6,1,7) DO BEGIN IF C(I) LSS KLB+DC1(I) THEN IF U(I)      B70
LSS 0.0 THEN BEGIN C(I)=KLB$ U(I)=0.0 ENDS IF C(I) GTR KUB-DC1(I) THEN     B71   E71
IF U(I)  GTR 0.0 THEN BEGIN C(I)=KUB$ U(I)=0.0$ ENDSENDS                   B72   E72   E70
FOR I=(6,1,7) DO IF C(I) GTR KUB THEN C(I)=KUB$
FOR I=(6,1,7) DO IF C(I) LSS KLB THEN C(I)=KLB$
U(8)=U(8)/20.0$
IF ACG(8) EQL 0.0 THEN                    U(8)=0.01*DG(8)$
IF F EQL 7 THEN U(8)=0.0$
FOR J=(3,1,12) DO BEGIN IF RS(J) EQL 2 THEN BEGIN Q=0.0$                   B73   B74
FOR K=(0,1,8) DO Q=Q+U(K)*R(K,J)$ IF Q LSS  0.0 THEN BEGIN
I=(J-3)/2$ IF ABS(U(I)) GTR ABS(U(I+1)) THEN U(I+1)=U(I)                   B75
ELSE U(I)=U(I+1)$ ENDS ENDS ENDS                                           E75   E74   E73
 Q=0.0$ FOR I=(0,1,F) DO Q=Q+U(I)*U(I)$
Q=SQRT(Q)$ FOR I=(0,1,F) DO U(I)=U(I)/Q$
WRITE( FOR I=(0,1,8) DO U(I)) $
LINLEN $
V4(1)=L$ FOR K=(0,1,F) DO V3(K)=C(K)$
FOR J=(0,1,F) DO             C(J)= C(J)+L*U(J)$
LE6..
T=0.0$ I=8$ ANL(T,I,D ,V ,ACCT ,TM )$
IF ABS(MD(8)) GTR XA THEN BEGIN
```

```
L=0.9*L$IF L LSS 0.000001*V4(1) THEN BEGIN                                     B76
WRITE('DEF. CONSTR. CURVATURE PROHIBITS FURTHER MOVE')$ IF CDA THEN            B77
GOTO LCAL$ GOTO LE1 END$                                                       E77
FOR K=(0,1,F) DO C(K)=V3(K)+L*U(K)$ GOTO LB6 END$                              E76
IF CDA THEN BEGIN CDA=FALSE$ GOTO L7 END$                                      B78   E78
IF MAC(8) GEQ MAC(-1) THEN BEGIN
                                   L=0.7*L$ IF L LEQ 0.000001*V4(1) THEN       B79
BEGIN
Q=0.0$ FOR K=(0,1,F) DO Q=Q+ACG(K)*U(K)$                                       B80
IF L*Q*G2 LSS 0.000001*MAC(-1) THEN
BEGIN WRITE('NO POSSIBLE PROGRESSIVE MOVE','MAC ',MAC(-1),'MD',MD(-1))$        B81
IF SB1 THEN GOTO LB61$
IF PPG THEN BEGIN WRITE('PPG=TRUE')$                                           B82
Q=0.0$ FOR K=(0,1,8) DO Q=Q+ACG(K)*ACGR(-2,K)$
FOR K=(0,1,8) DO ACG(K)=ACG(K)-Q*ACGR(-2,K)$ SB1=TRUE$ GOTO LBPR
END ELSE                                                                       E82
BEGIN  Q=0.0$ FOR K=(0,1,8) DO Q=Q+ACG(K)*ACGR(-1, K)$                         B83
          GOTO BPR1 END$                                                       E83
LB61.. SB1 = FALSE$
IF ABS(FS11) LEQ 0.0001 THEN BEGIN WRITE('CONVER','V5',V5)$                    B84
GOTO LE1 END$ IF CS8 NEQ 0  THEN BEGIN FS=1$                                   E84   B85
CS8=0$ GOTO L7 END$                                                            E85
IF MAC(-1) LSS V5(-1) THEN BEGIN FOR K=(0,1,8) DO V5(K)=C1(K)$                 B86
V5(-1)=MAC(-1)$ V5(-2)=MD(-1)$
MAC(-1)=10.0**10$ PS=FALSE$
                  IF FS11 LSS 0.0 THEN CDA=TRUE ELSE CDA=FALSE$
C1(8)=C1(8)-FS11$ GOTO L7 END$  LCAL..                                         E86
                               C1(8)=C1(8)+FS11$ FS11=-0.5*FS11$
IF FS11 LSS 0.0 THEN CDA= TRUE ELSE CDA=FALSE$
C1(8)=C1(8)-FS11$ FOR K=(0,1,7) DO C(K)=V5(K)$
MAC(-1)=10.0**10$ PS=FALSE$
                  GOTO L7$ END$                                                E81
          END$ FOR J=(0,1,F) DO C(J)=V3(J)+L*U(J)$ GOTO LB6 END$               E80   E79
LB7..
MD(-1)=MD(8)$ MAC(-1)=MAC(8)$
FOR I=(0,1,F) DO C1(I)=C(I)$
IF PS THEN BEGIN
WRITE('C1',C1)$                                                                B87
WRITE('MAC',MAC(-1),'MD',MD(-1) )$
END$                                                                           E87
PR=TRUE$
EXAM=TRUE$
MS=((10.0)**6)$
L=0.8*L$ FOR K=(0,1,F) DO C(K)=V3(K)+L*U(K)$
I=8$            ANL(T,I,D,V,ACCT,TM)$
IF MAC(8) LSS MAC(-1) THEN IF MD(8) LEQ XA THEN GOTO LB7 ELSE FOR
K=(0,1,F) DO C(K)=C1(K)$
IF NOT PS THEN WRITE('MAC',MAC(-1),'MD',MD(-1),'C1',C1)$
PS=TRUE$
Q1=0.0$ FOR K=(0,1,F) DO Q1=Q1+ACG(K)*U(K)$
             WRITE('Q12  FOR CONVG',Q1,Q2)$
IF L*Q2*Q1 LEQ    0.1*MAC(-1) THEN BEGIN EPL=0.001*MAC(-1)$                    B88
FOR I=(0,1,COL) DO BEGIN U(I)=0.0$                                             B89
FOR J=(0,1,COL) DO
```

```
        FOR K=(0,1,F)  DO
        U(I)=U(I)+VQ(I,J)*NTC(K,J)*ACG(K)$ ENDS                              E89
        Q=-(10.0**6)$
        IF COL GTR 0 THEN
              FOR J=(0,1,COL) DO IF U(J)/SQRT(VQ(J,J)) GTR Q
        THEN                          Q= U(J)/SQRT(VQ(J,J)) $
        WRITE('BETA',Q)$
        FOR J=(0,1,F) DO U(J)=ACG(J)-U1(J)$
        Q1=0.0$ FOR J=(0,1,F) DO Q1=Q1+U(J)*U(J)$
        Q1=SQRT(Q1)$
        IF Q1 GTR Q/2.0 THEN Q=Q1 ELSE Q=Q/2.0$
        Q1=0.0$ FOR J=(0,1,COL) DO Q1=Q1+SQRT(VQ(J,J))$
        WRITE('Q123',Q,Q1,Q2,Q3)$
        IF Q LEQ 0.1*EPL/(Q2*(2*Q1+1.0)) THEN BEGIN
                                    WRITE('CALCULATED DIFFERENCE BETWEEN     B90   Q
        MIN AND POINT IS',EPL)$ WRITE('VQ',VQ)$
        WRITE('Q',Q,'Q1',Q1)$
        WRITE('Q3',Q3)$
        GOTO LE1 ENDS                                                        E90
        ENDS                                                                 E88
        GOTO L7$
        LE1..
        FINISH$                                                              F0
  END BLOCK 1
COMPILATION COMPLETED
```