NASA Technical Paper 2479

November 1985

NASA-TP-2479 19860004477

A Method To Stabilize Linear Systems Using Eigenvalue Gradient Information

Carol D. Wieseman

NASA

NASA Technical Paper 2479

1985

A Method To Stabilize Linear Systems Using Eigenvalue Gradient Information

Carol D. Wieseman

Langley Research Center
Hampton, Virginia

National Aeronautics and Space Administration

Scientific and Technical Information Branch

Summary

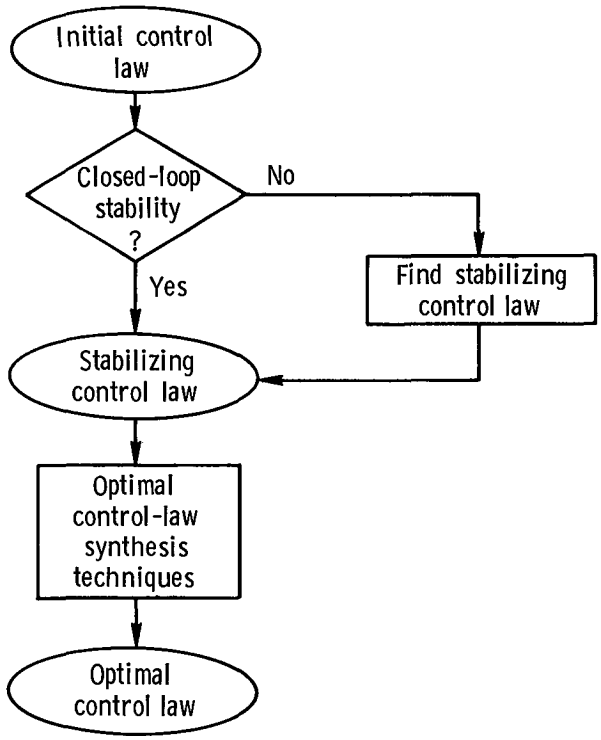
Formal optimization methods and eigenvalue gradient information are used to develop a stabilizing control law for a closed-loop linear system that is initially unstable. The method was originally formulated by using direct, constrained optimization methods with the constraints being the real parts of the eigenvalues. However, because of problems in trying to achieve stabilizing control laws, the problem was reformulated to be solved differently. The method described in this paper uses the Davidon-Fletcher-Powell minimization technique to solve an indirect, constrained minimization problem in which the performance index is the Kreisselmeier-Steinhauser function of the real parts of all the eigenvalues. The method is applied successfully to solve two different problems: the determination of a fourth-order control law that stabilizes a single-input single-output active flutter suppression system and the determination of a second-order control law for a multi-input multi-output lateral-directional flight control system. Various sets of design variables and initial starting points were chosen to show the robustness of the method.

Introduction

Many optimal control-law synthesis techniques require, as a starting point, a control law that results in a stable closed-loop system. Throughout this paper, such a control law will be referred to as a "stabilizing control law." Sketch A illustrates the relationship and sequence of the major steps to obtain optimal control laws.

The methods that are presently used to find stabilizing control laws (located in the loop in the sketch) are largely trial and error and therefore can be time consuming, especially for unstable open-loop systems (refs. 1 and 2). An automated method using Bass' algorithm (ref. 3) can determine a stabilizing control law when the structure of the control law is full-state feedback. However, this method is not applicable for reduced-order control laws of arbitrary structure.

The purpose of this paper is to present an automated method for obtaining stabilizing reduced-order control laws of arbitrary structure. In developing this method, an important goal was that these control laws be obtained with a minimum amount of trial and error by the control-law designer. Therefore, the method uses formal optimization methods and eigenvalue gradient information to obtain stabilizing control laws. The method was originally formulated as a constrained optimization problem with the constraints being the real parts of the eigenvalues. However, because of difficulties in achieving convergence, the method was reformulated as an indirect, constrained optimization problem with the performance index being the Kreisselmeier-Steinhauser (KS) function of the real parts of the eigenvalues. Analytical expressions for the gradients of the eigenvalues were derived and implemented.



Sketch A

The method is applied to solve two example problems. The first obtains a fourth-order stabilizing control law for a single-input single-output flutter suppression system. The second determines a second-order control law for a multi-input multi-output lateral-directional control system. Different choices of both the design variables and their initial values were used to illustrate the robustness of the method.

Symbols

$\mathbf{A}$	controller dynamics matrix $(M \times M)$
$\mathbf{B}$	controller input matrix $(M \times N_o)$
b_i, c_i	type 2 design variables
$\mathbf{C}$	controller output matrix $(N_c \times M)$
c	local chord of wing
d_i	type 1 design variables (coefficients of denominator polynomial)
$\mathbf{F}$	plant dynamics matrix $(N_s \times N_s)$

$\mathbf{F}_a$	augmented dynamics matrix $(N_s + M) \times (N_s + M)$
$\mathbf{G}_u$	plant input matrix $(N_s \times N_c)$
g_i	inequality constraints
$\mathbf{H}$	sensor output matrix $(N_o \times N_s)$
$\mathbf{I}$	identity matrix
J	performance index
j	$= \sqrt{-1}$
$\mathbf{K}$	matrix equal to real part of $[\mathbf{v}_i \mathbf{u}_i^T]$, $(N_s + M) \times (N_s + M)$
$\mathbf{K}_{11}$	$(N_s \times N_s)$ matrix formed from partitioning matrix $\mathbf{K}$
$\mathbf{K}_{12}$	$(N_s \times M)$ matrix formed from partitioning matrix $\mathbf{K}$
$\mathbf{K}_{21}$	$(M \times N_s)$ matrix formed from partitioning matrix $\mathbf{K}$
$\mathbf{K}_{22}$	$(M \times M)$ matrix formed from partitioning matrix $\mathbf{K}$
k	feedback gain
k_{nom}	nominal gain
M	order of controller
N_c	number of control inputs
N_o	number of outputs or sensors
N_s	number of plant states
n_i	type 1 design variables (coefficients of numerator polynomial)
$\mathbf{P}$	matrix defined in equation (15a), $(N_c + M) \times (N_o + M)$
P_{jk}	element of matrix $\mathbf{P}$
$\mathbf{R}$	key state selection matrix $(M \times N_s)$
r	drawdown factor
s	Laplace operator
$\mathbf{T}(s)$	transfer function matrix $(N_c \times N_o)$
$\mathbf{U}$	matrix consisting of all right eigenvectors, $(N_s + M) \times (N_s + M)$
$\mathbf{u}$	control input vector $(N_c \times 1)$
$\mathbf{u}_i$	normalized right eigenvector corresponding to λ_i, $(N_s + M) \times 1$
$\mathbf{V}$	matrix consisting of all left eigenvectors, $(N_s + M) \times (N_s + M)$
$\mathbf{v}_i$	normalized left eigenvector corresponding to λ_i, $(N_s + M) \times 1$
$\mathbf{X}$	vector of design variables
X_j	jth element of vector $\mathbf{X}$
$\mathbf{x}_a$	augmented state vector of order $(N_s + M) \times 1$
$\mathbf{x}_c$	controller state vector of order $(M \times 1)$
$\mathbf{x}_s$	plant state vector of order $(N_s \times 1)$
$\mathbf{y}$	measurement output vector of order $(N_o \times 1)$
$\alpha_i, \beta_i, \gamma_i$	type 1 design variables
δ_{ij}	Kronecker delta
λ_i	ith eigenvalue
λ_{i_R}	real part of ith eigenvalue

Subscripts:

I	imaginary
R	real

Superscript:

T	transpose

Matrix notation:

tr[ ]	trace of matrix
[ ]	matrix
{ }	vector
⌊ ⌋	row vector
$[\,]^T$	vector or matrix transpose
$[\,]^{-1}$	inverse of square matrix

Dots over symbols denote differentiation with respect to time.

Equations of Motion in State-Space Form

The specific applications of the method described in this paper are airplane control systems. This method is, however, applicable to any linear system as long as the equations of motion can be written in the following state space form (ref. 2):

$$\dot{\mathbf{x}}_s = \mathbf{F}\mathbf{x}_s + \mathbf{G}_u\mathbf{u} \tag{1}$$

$$\mathbf{y} = \mathbf{H}\mathbf{x}_s \tag{2}$$

These equations are combined with the following equations describing a control system with output feedback (ref. 2):

$$\dot{\mathbf{x}}_c = \mathbf{A}\mathbf{x}_c + \mathbf{B}\mathbf{y} \tag{3}$$

$$\mathbf{u} = \mathbf{C}\mathbf{x}_c \tag{4}$$

The block diagram in figure 1 illustrates the control scheme that is modeled in equations (1) to (4). The dashed lines in figure 1 indicate the three components of the control scheme: the plant, the measurement, and the control law. Using equations (3) and (4), the control law may be expressed in the form

$$\mathbf{u} = \mathbf{C}[s\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B}\mathbf{y} \tag{5}$$

where the matrix product $\mathbf{C}[s\mathbf{I}-\mathbf{A}]^{-1}\mathbf{B}$ is the transfer function matrix $\mathbf{T}(s)$.

Equations (1) to (4) can be combined into a set of augmented state equations by first defining an augmented state vector as

$$\mathbf{x}_a = \begin{Bmatrix} \mathbf{x}_s \\ \mathbf{x}_c \end{Bmatrix} \tag{6}$$

Next, substituting equation (4) into (1) and equation (2) into (3) yields, after rearrangement,

$$\begin{Bmatrix} \dot{\mathbf{x}}_s \\ \mathbf{x}_c \end{Bmatrix} = \begin{bmatrix} \mathbf{F} & \mathbf{G}_u\mathbf{C} \\ \mathbf{B}\mathbf{H} & \mathbf{A} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_s \\ \mathbf{x}_c \end{Bmatrix} \tag{7}$$

Defining the augmented system matrix as

$$\mathbf{F}_a = \begin{bmatrix} \mathbf{F} & \mathbf{G}_u\mathbf{C} \\ \mathbf{B}\mathbf{H} & \mathbf{A} \end{bmatrix} \tag{8}$$

allows equation (7) to be rewritten as

$$\dot{\mathbf{x}}_a = \mathbf{F}_a\mathbf{x}_a \tag{9}$$

Referring back to sketch A in the "Introduction," the initial control law is completely defined by matrices **A**, **B**, and **C** in equations (5) and (7). Closed-loop stability is determined by evaluating the eigenvalues of the augmented system matrix of equation (8). The stabilizing control law is completely defined by matrices **A**, **B**, and **C** when all the eigenvalues of equation (8) have negative real parts. The method presented in this paper is an automated method to select the elements of the **A**, **B**, and **C** matrices to provide a stable closed-loop system.

General Description of Design Algorithm

The method has been automated and implemented on the CDC CYBER computer system at the Langley Research Center. From now on, this method will be referred to as the "design algorithm." This section of the paper presents a general description of the design algorithm, shown in flow chart form in figure 2. (This flow chart will be referred to frequently in this section of the paper.) The purpose of the design algorithm is to produce a stabilizing control law, indicated by the ellipse on the right side of figure 2.

The design algorithm employs optimization to achieve its purpose. Necessary elements of the optimization routine are the design variables, a performance index J, and, in the present problem, the gradients of the closed-loop eigenvalues with respect to the design variables. Beginning at the top of the flow chart, the first step in the design algorithm is to choose the design variables and their initial values, thereby initializing the **A**, **B**, and **C** matrices and establishing the initial control law. Next, the eigenvalues of the augmented system matrix (closed-loop eigenvalues) are computed, after which a decision point is reached. If all the eigenvalues have negative real parts (that is, are stable), then this initial control law is a stabilizing control law and the design algorithm is terminated. If one or more eigenvalues have nonnegative real parts (that is, are neutrally stable or unstable), then the remainder of the design algorithm is exercised. The stability of the system is checked each time that the eigenvalues are calculated. If the system is still unstable, the number of iterations is checked to make sure that an arbitrary upper limit is not exceeded. This is done to prevent the design algorithm from continuing indefinitely if the solution is having difficulty converging.

In the event that the rest of the design algorithm will be exercised, a performance index is formulated. Gradients of the eigenvalues and the performance index are calculated. New values of the design variables are chosen by the optimization algorithm and the control law is updated. Iteration then continues as already described.

Choice of Design Variables

This section of the paper deals with choosing design variables as indicated by the top box of the flow chart in figure 2. The design variables are associated with the **A**, **B**, and **C** matrices within the control law. The total number of the elements within the **A**, **B**, and **C** matrices is $M(M + N_o + N_c)$. Of this total, $M(N_o + N_c)$ are independent (ref. 4) and can therefore be chosen as design variables for the design algorithm. The design algorithm accepts design variables in either of two different forms. The first (referred to a "type 1") is conveniently understood and interpreted by use of a transfer function matrix

and concepts from classical control theory; the second (type 2) is similar to a linear quadratic Gaussian (LQG) formulation. Both types of design variables result in reduced-order control laws.

Type 1 design variables. Type 1 design variables are chosen to be specific elements of the **A**, **B**, and **C** matrices. Any specific transfer function can be realized into state space **A**, **B**, **C** form in a variety of ways. For the sake of illustration, let us assume a fourth-order control law ($M = 4$) for a single-input ($N_c = 1$) single-output ($N_o = 1$) system. The number of available design variables for the design algorithm is $M(N_o + N_c) = 8$, and the transfer function matrix reduces to a single transfer function that may be written as

$$\mathbf{T}(s) = \mathbf{C}[s\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B} = \frac{k(n_3 s^3 + n_2 s^2 + n_1 s + n_0)}{s^4 + d_3 s^3 + d_2 s^2 + d_1 s + d_0} \quad (10)$$

A state space representation of the control law shown in equation (10) is given as follows:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & -d_0 \\ 1 & 0 & 0 & -d_1 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 1 & -d_3 \end{bmatrix} \quad (11a)$$

$$\mathbf{B} = \begin{Bmatrix} n_0 \\ n_1 \\ n_2 \\ n_3 \end{Bmatrix} \quad (11b)$$

$$\mathbf{C} = \lfloor 0 \quad 0 \quad 0 \quad k \rfloor \quad (11c)$$

Within the right side of equation (10), the nine quantities (k(Gain); n_0, n_1, n_2, and n_3 (coefficients of the numerator polynomial); and d_0, d_1, d_2, and d_3 (coefficients of the denominator polynomial)) are readily identified as elements of the **A**, **B**, and **C** matrices and are therefore candidates for design variables. From these nine quantities, the control-law designer chooses up to eight.

Type 2 design variables. Assuming again a fourth-order control law for a single-input single-output system, there are again eight available design variables. The type 2 design variables are the four elements of the **B** matrix and the four elements of the **C** matrix where

$$\mathbf{B} = \begin{Bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{Bmatrix} \quad (12a)$$

$$\mathbf{C} = \lfloor c_0 \quad c_1 \quad c_2 \quad c_3 \rfloor \quad (12b)$$

These design variables are analogous to elements of the gain matrices obtained from the LQG solution.

Recalling that the control laws in this paper are of reduced order, a key state selection matrix **R** is employed for retaining, deleting, and combining plant states. With the **B** and **C** matrices given in equations (12a) and (12b), respectively, the **A** matrix is a function of the **B** and **C** matrices and is given by (ref. 2)

$$\mathbf{A} = \mathbf{RFR}^T - \mathbf{BHR}^T + \mathbf{RG}_u\mathbf{C} \quad (13)$$

Eigenvalues of Augmented System Matrix

Once the control law has been initialized, the augmented system matrix is calculated. The eigenvalues of this matrix are calculated by subroutine EIGEN in ORACLS (ref. 5). The stability of the system is checked; if all the real parts of the eigenvalues are negative, then the program is terminated. If not, another decision must be made. If a maximum limit on the number of iterations is reached the algorithm is terminated. If not, the rest of the design algorithm is executed.

Eigenvalue Gradients

The eigenvalue gradients are needed for the design algorithm. Analytical expressions for the eigenvalue gradients are derived and depend on the choice of the design variables. Although the complete derivation of the expressions is given in appendix A, a general definition of the gradients of the eigenvalues is given by

$$\frac{\partial \lambda_i}{\partial \mathbf{P}} = [\mathbf{G}'^T \mathbf{v}_i \mathbf{u}_i^T \mathbf{H}'^T] \quad (14)$$

where

$$\mathbf{P} = \begin{bmatrix} 0 & \mathbf{C} \\ \mathbf{B} & \mathbf{A} \end{bmatrix}_{(N_c+M)\times(N_o+M)} \quad (15a)$$

Matrices $\mathbf{G}'$ and $\mathbf{H}'$ are defined, respectively, by

$$\mathbf{G}' = \begin{bmatrix} \mathbf{G}_u & 0 \\ 0 & \mathbf{I} \end{bmatrix}_{(N_s+M)\times(N_c+M)} \quad (15b)$$

$$\mathbf{H}' = \begin{bmatrix} \mathbf{H} & 0 \\ 0 & \mathbf{I} \end{bmatrix}_{(N_o+M)\times(N_s+M)} \quad (15c)$$

where **I** is an $(M \times M)$ identity matrix.

As seen in equation (14), both the left and right eigenvectors ($\mathbf{v}_i$ and $\mathbf{u}_i$, respectively) are needed to calculate the eigenvalue gradients. The right

eigenvectors were calculated by solving the following eigenvalue problem:

$$\mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{16}$$

There are two ways of calculating the left eigenvector. The first method is to solve the eigenvalue problem shown in equation (16) again but substituting $\mathbf{A}^T$ in place of $\mathbf{A}$. The second method is to calculate the left eigenvector directly from the right eigenvector. This method is derived from the relation

$$\mathbf{v}_i^T \mathbf{u}_j = \delta_{ij} \tag{17}$$

where δ_{ij} is the Kronecker delta that equals 1 when $i = j$ and equals 0 when $i \neq j$. For a given eigenvalue with its corresponding normalized eigenvectors, the inner product of the left and right eigenvectors is unity. For normalized eigenvectors corresponding to different eigenvalues, this inner product is 0.

The second method mentioned previously can be implemented in different ways. One way is to invert the complex matrix formed by all the right eigenvectors. To reduce the computation time, another method was developed. In this method, the left eigenvectors were calculated directly from the right eigenvectors without using complex matrix operations. A derivation of the method is given in appendix B.

The expressions for the eigenvalue gradients depend on the choice of the design variables. For type 1 design variables, the expressions for the gradients are given by

$$\frac{\partial \lambda_{i_R}}{\partial \mathbf{A}} = [\mathbf{K}_{22}] \tag{18a}$$

$$\frac{\partial \lambda_{i_R}}{\partial \mathbf{B}} = [\mathbf{K}_{21}]\mathbf{H}^T \tag{18b}$$

$$\frac{\partial \lambda_{i_R}}{\partial \mathbf{C}} = \mathbf{G}_u^T [\mathbf{K}_{12}] \tag{18c}$$

where $[\mathbf{K}_{22}]$, $[\mathbf{K}_{21}]$, and $[\mathbf{K}_{12}]$ are partitioned matrices from the real part of $\mathbf{v}_i \mathbf{u}_i^T$ given as

$$\text{Real } (\mathbf{v}_i \mathbf{u}_j^T) = \begin{bmatrix} [\mathbf{K}_{11}] & [\mathbf{K}_{12}] \\ [\mathbf{K}_{21}] & [\mathbf{K}_{22}] \end{bmatrix}_{(N_s+M)\times(N_s+M)} \tag{19}$$

For type 2 design variables, the additional relationship given by equation (13) must be satisfied. Because of this interdependence, the gradients of the eigenvalues with respect to the type 2 design variables have an additional term. The expressions for the gradients of the eigenvalues with respect to the type 2 design variables are

$$\frac{d\lambda_{i_R}}{d\mathbf{B}} = [\mathbf{K}_{21}]\mathbf{H}^T - [\mathbf{K}_{22}]\mathbf{R}\mathbf{H}^T \tag{20a}$$

$$\frac{d\lambda_{i_R}}{d\mathbf{C}} = \mathbf{G}_u^T [\mathbf{K}_{12}] + \mathbf{G}_u^T \mathbf{R}^T [\mathbf{K}_{22}] \tag{20b}$$

Optimization Formulation

Direct constrained approach. A stabilizing control law requires that the real parts of all closed-loop eigenvalues be negative. During the early development of the design algorithm, it seemed appropriate to meet this requirement directly by the use of constrained optimization. Therefore, inequality constraints of the form

$$g_i < 0 \qquad (i = 1, \ldots, n) \tag{21}$$

were specified where g_i are the real parts of the eigenvalues.

For any optimization problem, an objective function to be minimized must be specified. Notice that the purpose of the design algorithm will have been met when all the constraints are satisfied. Therefore, the selection of the objective function for this formulation is only of secondary importance and it was arbitrarily chosen to be one-half the sum of the squares of the elements of the row vector $\mathbf{C}$:

$$J = \frac{1}{2}\mathbf{C}\mathbf{C}^T \tag{22}$$

The CONMIN program (ref. 6), employing the method of feasible directions (ref. 7), was chosen for incorporation in the design algorithm.

An insurmountable numerical problem occurred within the design algorithm that resulted in the abandonment of the direct optimization approach. The subroutine that calculates the eigenvalues sorts them in order of increasing magnitude. During the optimization process, the eigenvalues do not remain in the same order but they may switch places. Unstable eigenvalues may switch places with stable ones, thus making it very difficult to keep track of unstable eigenvalues. Because gradients of the violated constraints (unstable eigenvalues) are needed during the optimization process, and since it is difficult to track the eigenvalues, it appears to the optimization routine that the gradients of the unstable eigenvalues are discontinuous. Therefore, as stated previously, this formulation was abandoned.

Indirect constrained approach. The problem was reformulated by using indirect constrained optimization methods. The performance index was chosen to be the Kreisselmeier-Steinhauser (KS) function (ref. 8). The KS function, which is a penalty function that takes into account the real parts of all of the eigenvalues (stable and unstable), is given as

$$\mathrm{KS} = \frac{1}{r} \ln \left[\sum_{i=1}^{n} \exp(r\lambda_{i_R}) \right] \tag{23}$$

The function is dominated by the unstable eigenvalues (those with positive real parts). The contribution of the stable eigenvalues to the KS function is relatively small because of the behavior of the exponential function. The quantity r is a drawdown factor that (for values of r greater than unity) serves to increase the relative contribution of the unstable eigenvalues to the performance index. Figure 3 shows the exponential function and its relation to some representative eigenvalues.

The gradients of the performance index with respect to the design variables were derived analytically. The equations for the gradients are

$$\frac{\partial \mathrm{KS}}{\partial X_j} = \frac{\sum_{i=1}^{n} \exp(r\lambda_{i_R}) \partial \lambda_{i_R} / \partial X_j}{\sum_{i=1}^{n} \exp(r\lambda_{i_R})} \tag{24}$$

Expressions for the partial derivatives $\partial \lambda_{i_R} / \partial X_j$ within the numerator of equation (24) are given in equations (18) and (20).

To perform the indirect optimization, the Automated Design Synthesis (ADS) package of programs (ref. 9) was used. Within ADS, a variable metric method, the Davidon-Fletcher-Powell algorithm (ref. 10), was chosen because of its known rapid rate of convergence for this type of problem.

Termination

The design algorithm may terminate for the following three different reasons, as indicated in figure 2.

Successful termination. The first reason is the successful termination that occurs when a stabilizing control law is found.

Maximum iterations exceeded. The second reason occurs when the maximum number of iterations is exceeded. Depending on how the optimization process is proceeding, a feasible solution may be reached if one just increases the maximum number of iterations or restarts the design algorithm at the final control law of the previous optimization process. If, in the attempt to minimize the performance index, the magnitude of the positive real part of the unstable eigenvalues continues to decrease, but the number of unstable eigenvalues starts to increase, then a solution may not be achievable. In this case, the only option is to choose another initial starting point.

Unsuccessful termination. The third reason occurs when a local minimum of the performance index is found before the closed-loop system is stabilized. In this case it was postulated that convergence could be reached if the design algorithm started with this control law for the next iteration but with an increased value of the drawdown factor r. An option to restart the optimization process with an increased value of r was implemented in the design algorithm; but after exercising the algorithm with several different starting points, it was determined that if the algorithm did not converge with the initial value of r, it would not converge with an increased value of r. Therefore, the option was eliminated. The default value of r used in all cases was chosen as unity. However, this value is checked initially to make sure that the magnitude of r multiplied by the value of the real part of the most unstable eigenvalue did not exceed a maximum value. This maximum value is defined by the computer and represents an upper limit for which the exponential function can be evaluated. In a few test cases run from different starting points, a local minimum was found before a feasible solution was found.

Numerical Examples

The design algorithm described in this paper is applied to solve two different problems. The first application determines a fourth-order control law that stabilizes a single-input single-output unstable open-loop system for active flutter suppression. The second application determines a second-order control law for a multi-input multi-output, open-loop unstable lateral-directional control system. Within each application several examples will be given to illustrate the effectiveness of the method.

First Application

The configuration used for the flutter suppression problem is an aeroelastic wind-tunnel model of the DAST ARW-1 (Drones for Aerodynamic and Structural Testing—Aeroelastic Research Wing 1) shown in figure 4. The plant model is a single-input single-output system of 25th order. The locations of the

sensor and the control surface are shown in the figure. The sensor measures the vertical acceleration that is fed back through the control system, which in turn determines control deflections needed to suppress flutter.

Case 1. The first example involves only one design variable: the gain of a given fourth-order control law. An unstable starting point was derived from a known stabilizing control law, given by equation (30) in reference 2, and is repeated here as

$$\mathbf{T}(s) = \frac{k(s+24.74)(s^2+87.63s+13\,806)}{(s+3.864)(s+3270)(s^2+20.97s+1423)} \tag{25}$$

When the gain k is the nominal value of gain in reference 2 (k_{nom}), this control law results in a stable closed-loop system. A value of gain ($k = 10k_{\text{nom}}$) was chosen to provide an unstable closed-loop system as a starting point. Using the design algorithm, a stable closed-loop system was found.

The history of the normalized performance index is given in figure 5. The solution converged to a stabilizing control law during the fifth iteration. The final value of the normalized performance index was -0.044. The actual value is not important and can be positive, negative, or zero. For this particular example with only one design variable, the optimization algorithm corresponded to a simple one-dimensional search. Figure 6 shows a history of the gain during the iteration process. The final value of gain is $2.08k_{\text{nom}}$. Figure 7 shows a history of some of the eigenvalues. At the starting point, there was one complex conjugate pair of unstable eigenvalues indicated by the open-circle symbol in the right half-plane.

Case 2. The second example uses the same form of the control law as in equations (10) and (25): a third-order numerator polynomial over a fourth-order denominator polynomial. The design variables were chosen as the coefficients of the numerator and denominator polynomials (type 1 design variables). The initial values of the eight design variables as well as the value of gain were all chosen as unity.

Figure 8 shows a history of the normalized performance index. The design algorithm converged to a stable system in five iterations. Figure 9 shows histories of the design variables. Design variables d_0, d_1, and n_0 (corresponding, respectively, to the coefficients of the two lowest order terms of the denominator polynomial and the constant term of the numerator polynomial) remained at their initial values. A history of some of the eigenvalues is given in figure 10. At the starting point, there were a complex conjugate pair of unstable eigenvalues and a real unstable eigenvalue. During the first iteration the real eigenvalue became more unstable. During the second iteration it combined with a real stable eigenvalue and formed a complex conjugate pair of unstable eigenvalues. The design algorithm then stabilized this and the other unstable pair of eigenvalues.

Case 3. The method was also verified by using type 2 design variables. The order of the control law was chosen again to be four, resulting in eight design variables (four elements each of the **B** and **C** matrices). To generate the **A** matrix of the control law, four states needed to be chosen. The key states chosen were the displacements and velocities of the first wing bending and first wing torsion modes.

An arbitrary starting point was chosen in which all the design variables were set to unity. However, for this starting point, the design algorithm did not converge to a stabilizing control law because a local minimum was found without obtaining a stabilizing control law. A new starting point was arbitrarily chosen by setting the elements of the **C** matrix to 0.1 (an effect which is equivalent to decreasing the gain of the control law by a factor of 10). From this starting point, the design algorithm converged to a stabilizing control law in 14 iterations. A history of the normalized performance index is shown in figure 11. Histories of the design variables are shown in figure 12. All the values of the design variables changed during the optimization process. A history of some of the eigenvalues is shown in figure 13. At the starting point, there were a pair of complex conjugate unstable eigenvalues and two real unstable eigenvalues.

Three observations about the behavior of the eigenvalues can be made from figure 13. The first is that during the attempt to stabilize all the eigenvalues, the eigenvalues can actually move in the direction opposite to that desired: that is, unstable eigenvalues become more unstable and stable eigenvalues become less stable. The second is that during the optimization process, some eigenvalues can crisscross back and forth across the imaginary axes, thus becoming unstable or stable until all the eigenvalues are finally stable. The third is that real eigenvalues combined and formed complex conjugate pairs of eigenvalues.

Second Application

The second application successfully solved by using this method was the determination of a second-order control law for stabilizing an unstable multi-input multi-output lateral-directional automatic

flight control system (AFCS) for the DAST ARW-2 (ref 11). The augmented ARW-2 vehicle has an unstable Dutch roll mode. Figure 14 shows a planform of the ARW-2 with the locations of the control surfaces (rudder and differential stabilizer) and the sensors (which measure roll and yaw rates) to be used in the lateral-directional AFCS. Figure 15 is a block diagram of the lateral-directional AFCS, which is characterized by two feedback loops (one for yaw rate and one for roll rate) and cross-feed. The design algorithm was used to find the coefficients of the two first-order transfer functions identified as control laws in the block diagram. These transfer functions from figure 15 are given as

$$\mathbf{T}_1(s) = \frac{\gamma_1}{\beta_1 s + \alpha_1} \tag{26}$$

$$\mathbf{T}_2(s) = \frac{\gamma_2}{\beta_2 s + \alpha_2} \tag{27}$$

The transfer function is denoted by $\mathbf{T}_1(s)$ in the roll rate loop and by $\mathbf{T}_2(s)$ in the yaw rate loop. The transfer function matrix representation of the control law can be realized in the state space **A**, **B**, **C** form given by the 2×2 matrices shown in

$$\mathbf{A} = \begin{bmatrix} -\alpha_1 & 0 \\ 0 & -\alpha_2 \end{bmatrix} \tag{28}$$

$$\mathbf{B} = \begin{bmatrix} \beta_1 & 0 \\ 0 & \beta_2 \end{bmatrix} \tag{29}$$

$$\mathbf{C} = \begin{bmatrix} \gamma_1 & 0 \\ 0 & \gamma_2 \end{bmatrix} \tag{30}$$

Two different examples are presented, both with six type 1 design variables. The six design variables are α_1, α_2, β_1, β_2, γ_1, and γ_2.

Case 1. For the first example, the initial values of the design variables in the order listed previously are 0, 1, 1, 1, 1, and 1, which correspond to $\mathbf{T}_1(s) = 1/s$ and $\mathbf{T}_2(s) = 1/(s+1)$. A history of the normalized performance index is shown in figure 16. The program converged to a stable system in one iteration. The design variables are shown in figure 17. All the design variables changed during the optimization. A history of some of the eigenvalues is shown in figure 18. At the starting point there was a pair of unstable complex conjugate eigenvalues that were stabilized by using the design algorithm.

Case 2. For the second example, the initial values of the design variables are 0, 0, 1, 1, 0, and 0, which correspond to $\mathbf{T}_1(s) = 0/s$ and $\mathbf{T}_2(s) = 0/s$. The eigenvalues at this starting point are the open-loop eigenvalues plus two additional eigenvalues at the origin. A plot of the normalized performance index is shown in figure 19. A stable system was found in three iterations. The histories of the design variables are shown in figure 20. A history of some of the eigenvalues is shown in figure 21.

Concluding Remarks

A method has been developed that uses a formal optimization method and eigenvalue gradient information to determine a control law that stabilizes a linear system. The goal of the method is to determine a stabilizing control law of a given general structure with a minimum amount of trial and error by the control-law designer. The method is easy to implement for a general problem in which the state space model of the system and the structure of the control law are given. The method was originally formulated as a constrained optimization problem, but because of convergence problems, the problem was reformulated as an indirect, constrained optimization problem in which the performance index is a function of the real parts of the eigenvalues. Analytical expressions for the gradients of the eigenvalues were derived and implemented.

The effectiveness of this method was successfully demonstrated by using several numerical examples. Using a single-input single-output flutter suppression problem as the first application of the method, several fourth-order control laws were determined that stabilized the unstable system. Two types of the design variables and several initial values of these design variables were used. The second application was the determination of a second-order control law for a multi-input multi-output lateral-directional system. The results of the numerical examples showed that the method was very successful in reaching its objective of finding a stabilizing control law.

NASA Langley Research Center
Hampton, VA 23665-5225
August 13, 1985

Appendix A

Eigenvalue Gradients

This appendix describes the derivation of the equations for the eigenvalue gradients (shown in eqs. (18) and (20)) using the method of reference 12. It is known that for any real square matrix $\mathbf{A}$, that is $(N \times N)$, the following equations are satisfied:

$$\mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{A1}$$

$$\mathbf{A}^T \mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{A2}$$

In the previous equations, λ_i are distinct eigenvalues, and $\mathbf{u}_i$ and $\mathbf{v}_i$ are complex right and left normalized eigenvectors (dimensioned $N \times 1$) associated with the specific eigenvalue. Differentiating equation (A1) with respect to a parameter p gives

$$\frac{\partial \mathbf{A}}{\partial p}\mathbf{u}_i + \mathbf{A}\frac{\partial \mathbf{u}_i}{\partial p} = \frac{\partial \lambda_i}{\partial p}\mathbf{u}_i + \lambda_i \frac{\partial \mathbf{u}_i}{\partial p} \tag{A3}$$

Equation (A3) is a vector equation. In order to solve for $\partial\lambda_i/\partial p$, which is a scalar, the equation is rearranged and premultiplied by $\mathbf{v}_i^T$ to obtain

$$\mathbf{v}_i^T \frac{\partial \lambda_i}{\partial p}\mathbf{u}_i = \mathbf{v}_i^T \frac{\partial \mathbf{A}}{\partial p}\mathbf{u}_i + \mathbf{v}_i^T \mathbf{A}\frac{\partial \mathbf{u}_i}{\partial p} - \mathbf{v}_i^T \lambda_i \frac{\partial \mathbf{u}_i}{\partial p} \tag{A4}$$

Since this is a scalar equation, the last two terms can be transposed since the transpose of a scalar is itself. Using the distributive property

$$\mathbf{v}_i^T \frac{\partial \lambda_i}{\partial p}\mathbf{u}_i = \mathbf{v}_i^T \frac{\partial \mathbf{A}}{\partial p}\mathbf{u}_i + \frac{\partial \mathbf{u}_i^T}{\partial p}(\mathbf{A}^T \mathbf{v}_i - \lambda_i \mathbf{v}_i) \tag{A5}$$

The last term is identically 0 by employing equation (A2). Equation (A5) becomes

$$\mathbf{v}_i^T \frac{\partial \lambda_i}{\partial p}\mathbf{u}_i = \mathbf{v}_i^T \frac{\partial \mathbf{A}}{\partial p}\mathbf{u}_i \tag{A6}$$

Since $\partial\lambda_i/\partial p$ is a scalar, equation (A6) is a scalar equation, and $\mathbf{v}_i^T\mathbf{u}_i = 1$ for normalized eigenvalues, the following equation is obtained:

$$\frac{\partial \lambda_i}{\partial p} = \mathbf{v}_i^T \frac{\partial \mathbf{A}}{\partial p}\mathbf{u}_i \tag{A7}$$

Steps (A1) to (A7) are also given in reference 12.

The next effort is to evaluate $\partial\mathbf{A}/\partial p$. Matrix $\mathbf{A}$ is the closed-loop augmented system matrix. For our specific example, the augmented system matrix $\mathbf{A}$ is the matrix $\mathbf{F}_a$ defined by

$$\mathbf{F}_a = \begin{bmatrix} \mathbf{F} & \mathbf{G}_u\mathbf{C} \\ \mathbf{B}\mathbf{H} & \mathbf{A} \end{bmatrix}_{(N_s+M)\times(N_s+M)} \tag{A8}$$

The parameters p are defined to be the elements P_{jk} of the matrix $\mathbf{P}$ where $\mathbf{P}$ is defined as

$$\mathbf{P} = \begin{bmatrix} 0 & \mathbf{C} \\ \mathbf{B} & \mathbf{A} \end{bmatrix}_{(N_c+M)\times(N_o+M)} \tag{A9}$$

N_c is the number of control inputs, N_o is the number of outputs, and M is the order of the controller.

The matrices $\mathbf{F}'$, $\mathbf{G}'$, and $\mathbf{H}'$ are defined, respectively, as

$$\mathbf{F}' = \begin{bmatrix} \mathbf{F} & 0 \\ 0 & 0 \end{bmatrix}_{(N_s+M)\times(N_s+M)} \tag{A10}$$

$$\mathbf{G}' = \begin{bmatrix} \mathbf{G}_u & 0 \\ 0 & \mathbf{I} \end{bmatrix}_{(N_s+M)\times(N_c+M)} \tag{A11}$$

$$\mathbf{H}' = \begin{bmatrix} \mathbf{H} & 0 \\ 0 & \mathbf{I} \end{bmatrix}_{(N_o+M)\times(N_s+M)} \tag{A12}$$

The symbol $\mathbf{I}$ is an $(M \times M)$ identity matrix. Using the definitions given in equations (A10), (A11), and (A12), the augmented state system matrix is then defined by

$$\mathbf{F}_a = \mathbf{F}' + \mathbf{G}'\mathbf{P}\mathbf{H}' \tag{A13}$$

Differentiating equation (A13) and noting that $\mathbf{F}'$ is not a function of the parameters P_{jk}, the following equation is obtained:

$$\frac{\partial \mathbf{F}_a}{\partial P_{jk}} = \frac{\partial(\mathbf{G}'\mathbf{P}\mathbf{H}')}{\partial P_{jk}} \tag{A14}$$

Substituting equation (A14) into (A7) gives the following scalar equation:

$$\frac{\partial \lambda_i}{\partial P_{jk}} = \mathbf{v}_i^T \frac{\partial(\mathbf{G}'\mathbf{P}\mathbf{H}')}{\partial P_{jk}}\mathbf{u}_i \tag{A15}$$

The trace properties of compatible matrix products can be used to write equation (A15) as

$$\frac{\partial \lambda_i}{\partial P_{jk}} = \mathrm{tr}\left[\frac{\partial(\mathbf{G}'\mathbf{P}\mathbf{H}')}{\partial P_{jk}}\mathbf{u}_i\mathbf{v}_i^T\right] \tag{A16}$$

Since $\mathbf{H}'$ is not a function of P_{jk}, equation (A16) can be written as

$$\frac{\partial \lambda_i}{\partial P_{jk}} = \mathrm{tr}\left[\frac{\partial(\mathbf{G}'\mathbf{P})}{\partial P_{jk}}\mathbf{H}'\mathbf{u}_i\mathbf{v}_i^T\right] \tag{A17}$$

Since $\mathrm{tr}[\mathbf{AB}] = \mathrm{tr}[\mathbf{BA}]$, equation (A17) can be written as

$$\frac{\partial \lambda_i}{\partial P_{jk}} = \mathrm{tr}\left[\mathbf{H}'\mathbf{u}_i\mathbf{v}_i^T\frac{\partial(\mathbf{G}'\mathbf{P})}{\partial P_{jk}}\right] \tag{A18}$$

Since $\mathrm{tr}[\mathbf{A}] = \mathrm{tr}[\mathbf{A}^T]$, equation (A18) can be written as

$$\frac{\partial \lambda_i}{\partial P_{jk}} = \mathrm{tr}\left[\frac{\partial(\mathbf{P}^T\mathbf{G}'^T)}{\partial P_{jk}}\mathbf{v}_i\mathbf{u}_i^T\mathbf{H}'^T\right] \tag{A19}$$

Since $\mathbf{G}'$ is not a function of P_{jk}, equation (A19) can be written as

$$\frac{\partial \lambda_i}{\partial P_{jk}} = \mathrm{tr}\left[\frac{\partial(\mathbf{P}^T)}{\partial P_{jk}}\mathbf{G}'^T\mathbf{v}_i\mathbf{u}_i^T\mathbf{H}'^T\right] = \left[\mathbf{G}'^T\mathbf{v}_i\mathbf{u}_i^T\mathbf{H}'^T\right]_{jk} \tag{A20}$$

Combining all derivatives with respect to P_{jk} gives

$$\frac{\partial \lambda_i}{\partial \mathbf{P}} = \left[\mathbf{G}'^T\mathbf{v}_i\mathbf{u}_i^T\mathbf{H}'^T\right] \tag{A21}$$

Having already defined $\mathbf{G}'$ and $\mathbf{H}'$ in equations (A11) and (A12), respectively, the next step is to get an expression for $\mathbf{v}_i\mathbf{u}_i^T$. The right eigenvector $\mathbf{u}_i$ is calculated by solving the eigenvalue problem given in equation (A1). The left eigenvector $\mathbf{v}_i$ is calculated from the right eigenvector without using the complex matrix operations. (This method is derived in appendix B.) The product of the left and right eigenvectors is expanded by separating both the left eigenvector and the right eigenvector into their respective real and imaginary parts as seen in

$$\mathbf{v}_i\mathbf{u}_i^T = \left(\mathbf{v}_{i_R} + j\mathbf{v}_{i_I}\right)\left(\mathbf{u}_{i_R} + j\mathbf{u}_{i_I}\right)^T \tag{A22}$$

Expanding the product of the eigenvectors gives

$$\mathbf{v}_i\mathbf{u}_i^T = \left(\mathbf{v}_{i_R}\mathbf{u}_{i_R}^T - \mathbf{v}_{i_I}\mathbf{u}_{i_I}^T\right) + j\left(\mathbf{v}_{i_R}\mathbf{u}_{i_I}^T + \mathbf{v}_{i_I}\mathbf{u}_{i_R}^T\right) \tag{A23}$$

For the gradients of the real parts of the eigenvalues, only the real part of equation (A23) is needed. This term is shown as

$$\mathbf{K} = \mathrm{Real}\left[\mathbf{v}_i\mathbf{u}_i^T\right] \tag{A24}$$

where $\mathbf{K}$ can be partitioned as shown in

$$[\mathbf{K}] = \begin{bmatrix}\mathbf{K}_{11} & \mathbf{K}_{12}\\ \mathbf{K}_{21} & \mathbf{K}_{22}\end{bmatrix} = \begin{bmatrix}(N_s \times N_s) & (N_s \times M)\\ (M \times N_s) & (M \times M)\end{bmatrix} \tag{A25}$$

Substituting equation (A25) into (A21) gives

$$\frac{\partial \lambda_i}{\partial \mathbf{P}} = \begin{bmatrix}\mathbf{G}_u & 0\\ 0 & \mathbf{I}\end{bmatrix}^T\begin{bmatrix}\mathbf{K}_{11} & \mathbf{K}_{12}\\ \mathbf{K}_{21} & \mathbf{K}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{H} & 0\\ 0 & \mathbf{I}\end{bmatrix}^T \tag{A26}$$

Knowing the definition of $\mathbf{P}$ given in equation (A9), the gradients of the real parts of the eigenvalues with respect to the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are given, respectively, by

$$\frac{\partial \lambda_{i_R}}{\partial \mathbf{A}} = [\mathbf{K}_{22}] \tag{A27}$$

$$\frac{\partial \lambda_{i_R}}{\partial \mathbf{B}} = [\mathbf{K}_{21}]\mathbf{H}^T \tag{A28}$$

$$\frac{\partial \lambda_{i_R}}{\partial \mathbf{C}} = \mathbf{G}_u^T[\mathbf{K}_{12}] \tag{A29}$$

These equations are valid for type 1 design variables.

For the case 2 design variables, the following equation is also satisfied:

$$\mathbf{A} = \mathbf{R}\mathbf{F}\mathbf{R}^T - \mathbf{B}\mathbf{H}\mathbf{R}^T + \mathbf{R}\mathbf{G}_u\mathbf{C} \tag{A30}$$

The total derivatives of the eigenvalues with respect to the design variables $\mathbf{B}$ and $\mathbf{C}$ have an additional term because of this expression. The gradients of the real parts of the eigenvalues for type 2 design variables are shown as

$$\frac{d\lambda_{i_R}}{d\mathbf{B}} = \frac{\partial \lambda_{i_R}}{\partial \mathbf{B}} - \frac{\partial \lambda_{i_R}}{\partial \mathbf{A}}\mathbf{R}\mathbf{H}^T \tag{A31}$$

$$\frac{d\lambda_{i_R}}{d\mathbf{C}} = \frac{\partial \lambda_{i_R}}{\partial \mathbf{C}} + \mathbf{G}_u^T\mathbf{R}^T\frac{\partial \lambda_{i_R}}{\partial \mathbf{A}} \tag{A32}$$

After substituting the appropriate partial derivatives (eqs. (A28) and (A29)) into equations (A31) and (A32), the gradients of the real parts of the eigenvalues with respect to the type 2 design variables are given as

$$\frac{d\lambda_{i_R}}{d\mathbf{B}} = [\mathbf{K}_{21}]\mathbf{H}^T - [\mathbf{K}_{22}]\mathbf{R}\mathbf{H}^T \tag{A33}$$

$$\frac{d\lambda_{i_R}}{d\mathbf{C}} = \mathbf{G}_u^T[\mathbf{K}_{12}] + \mathbf{G}_u^T\mathbf{R}^T[\mathbf{K}_{22}] \tag{A34}$$

Appendix B

Calculation of Left Eigenvector

This appendix describes how the complex left eigenvectors are calculated directly from the complex right eigenvectors without complex matrix operations. The definitions of the right eigenvectors $\mathbf{u}_i$ and left eigenvectors $\mathbf{v}_i$ are given, respectively, by

$$\mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{B1}$$

$$\mathbf{A}^T \mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{B2}$$

Note that both $\mathbf{A}$ and $\mathbf{A}^T$ have the same eigenvalues and if $\mathbf{A} = \mathbf{A}^T$, the right and left eigenvectors are equal. The following equation is known to be satisfied for normalized eigenvectors:

$$\mathbf{v}_i^T \mathbf{u}_j = \delta_{ij} \tag{B3}$$

Since the eigenvectors $\mathbf{u}_j$ and $\mathbf{v}_i$ are both complex, equation (B3) can be expanded to

$$\left(\mathbf{v}_{i_R} + j\mathbf{v}_{i_I}\right)^T \left(\mathbf{u}_{j_R} + j\mathbf{u}_{j_I}\right) = \delta_{ij} \tag{B4}$$

Expanding the relation, the equation becomes

$$\left(\mathbf{v}_{i_R}^T \mathbf{u}_{j_R} - \mathbf{v}_{i_I}^T \mathbf{u}_{j_I}\right) + j\left(\mathbf{v}_{i_R}^T \mathbf{u}_{j_I} + \mathbf{v}_{i_I}^T \mathbf{u}_{j_R}\right) = \delta_{ij} \tag{B5}$$

The real terms can be separated from the imaginary terms to form

$$\mathbf{v}_{i_R}^T \mathbf{u}_{j_R} - \mathbf{v}_{i_I}^T \mathbf{u}_{j_I} = \delta_{ij} \tag{B6}$$

$$\mathbf{v}_{i_R}^T \mathbf{u}_{j_I} + \mathbf{v}_{i_I}^T \mathbf{u}_{j_R} = 0 \tag{B7}$$

If the right eigenvector is real where $\mathbf{u}_j$ is given and i refers to any eigenvector, equations (B6) and (B7) become, respectively,

$$\mathbf{v}_{i_R}^T \mathbf{u}_{j_R} = \delta_{ij} \tag{B8}$$

$$\mathbf{v}_{i_I}^T \mathbf{u}_{j_R} = 0 \tag{B9}$$

Since the imaginary part of the right eigenvector is 0, the following equations are satisfied:

$$\mathbf{v}_{i_I}^T \mathbf{u}_{j_I} = 0 \tag{B10}$$

$$\mathbf{v}_{i_R}^T \mathbf{u}_{j_I} = 0 \tag{B11}$$

For a right eigenvector $\mathbf{u}_j$ that is complex, the defining relations are equations (B6) and (B7). Its complex conjugate is also an eigenvector. Equation (B7) becomes

$$-\mathbf{v}_{i_R}^T \mathbf{u}_{j_I} + \mathbf{v}_{i_I}^T \mathbf{u}_{j_R} = 0 \tag{B12}$$

Adding equations (B12) and (B7) gives

$$2\left(\mathbf{v}_{i_I}^T \mathbf{u}_{j_R}\right) = 0 \tag{B13}$$

which can also be written as

$$\mathbf{v}_{i_I}^T \mathbf{u}_{j_R} = -\mathbf{v}_{i_I}^T \mathbf{u}_{j_R} = 0 \tag{B14}$$

Equation (B14) can be substituted into equation (B7). Then, equations (B6) and (B7) can be added to obtain

$$\mathbf{v}_{i_R}^T \mathbf{u}_{j_R} - \mathbf{v}_{i_I}^T \mathbf{u}_{j_I} + \mathbf{v}_{i_R}^T \mathbf{u}_{j_I} - \mathbf{v}_{i_I}^T \mathbf{u}_{j_R} = \delta_{ij} \tag{B15}$$

Applying the distributive property gives

$$\left(\mathbf{v}_{i_R} - \mathbf{v}_{i_I}\right)^T \left(\mathbf{u}_{j_R} + \mathbf{u}_{j_I}\right) = \delta_{ij} \tag{B16}$$

Combining all the equations for all the eigenvalues gives

$$\left(\mathbf{V}_R - \mathbf{V}_I\right)^T \left(\mathbf{U}_R + \mathbf{U}_I\right) = \mathbf{I} \tag{B17}$$

The matrix of the left eigenvectors can be determined from the right eigenvectors by using

$$\left(\mathbf{V}_R - \mathbf{V}_I\right) = \left[\left(\mathbf{U}_R + \mathbf{U}_I\right)^{-1}\right]^T \tag{B18}$$

The real and imaginary parts of the eigenvectors corresponding to each specific eigenvalue can then be extracted.

References

1. Abel, Irving; Perry, Boyd, III; and Newsom, Jerry R.: *Comparison of Analytical and Wind-Tunnel Results for Flutter and Gust Response of a Transport Wing With Active Controls.* NASA TP-2010, 1982.
2. Mukhopadhyay, Vivek; Newsom, Jerry R.; and Abel, Irving: *A Method for Obtaining Reduced-Order Control Laws for High-Order Systems Using Optimization Techniques.* NASA TP-1876, 1981.
3. Armstrong, Ernest S.: An Extension of Bass' Algorithm for Stabilizing Linear Continuous Constant Systems. *IEEE Trans. Autom. Control,* vol. AC-20, no. 1, Feb. 1975, pp. 153–154.
4. Denery, Dallas G.: *Identification of System Parameters From Input-Output Data With Application to Air Vehicles.* NASA TN D-6468, 1971.
5. Armstrong, Ernest S.: *ORACLS—A System for Linear-Quadratic-Gaussian Control Law Design.* NASA TP-1106, 1978.
6. Vanderplaats, Garret N.: *CONMIN—A FORTRAN Program for Constrained Function Minimization—User's Manual.* NASA TM X- 62282, 1973.
7. Zoutendijk, G.: *Methods of Feasible Directions.* Elsevier Pub. Co., 1960.
8. Kreisselmeier, G.; and Steinhauser, R.: Systematic Control Design by Optimizing a Vector Performance Index. *Computer Aided Design of Control Systems,* M. A. Cuenod, ed., Pergamon Press, Inc., c.1980, pp. 113–117.
9. Vanderplaats, G. N.: *ADS—A FORTRAN Program for Automated Design Synthesis—Version 1.00.* NASA CR-172460, 1984.
10. Fletcher, R.; and Powell, M. J. D.: A Rapidly Convergent Descent Method for Minimization. *Computer J.,* vol. 6, no. 2, July 1963, pp. 163–168.
11. Boeing Wichita Co.: *Integrated Design of a High Aspect Ratio Research Wing With an Active Control System for Flight Tests on a BQM-34E/F Drone Vehicle, Volume I.* Doc. No. D3-11535-1 Volume I (Contract NAS1-14665), Boeing Co., 1979. (Available as NASA CR-166108.)
12. Porter, Brian; and Crossley, Roger: *Modal Control: Theory and Applications.* Taylor & Francis (London), 1972.

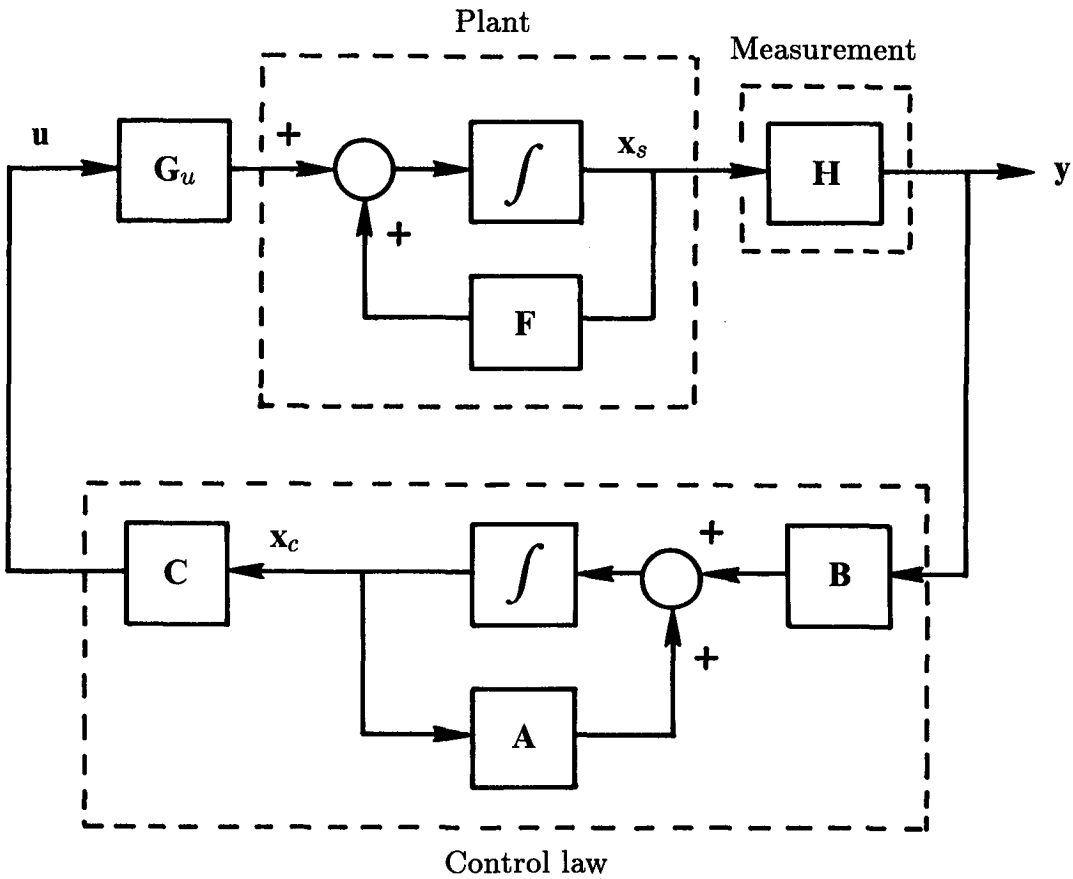


Figure 1. Block diagram of augmented system.

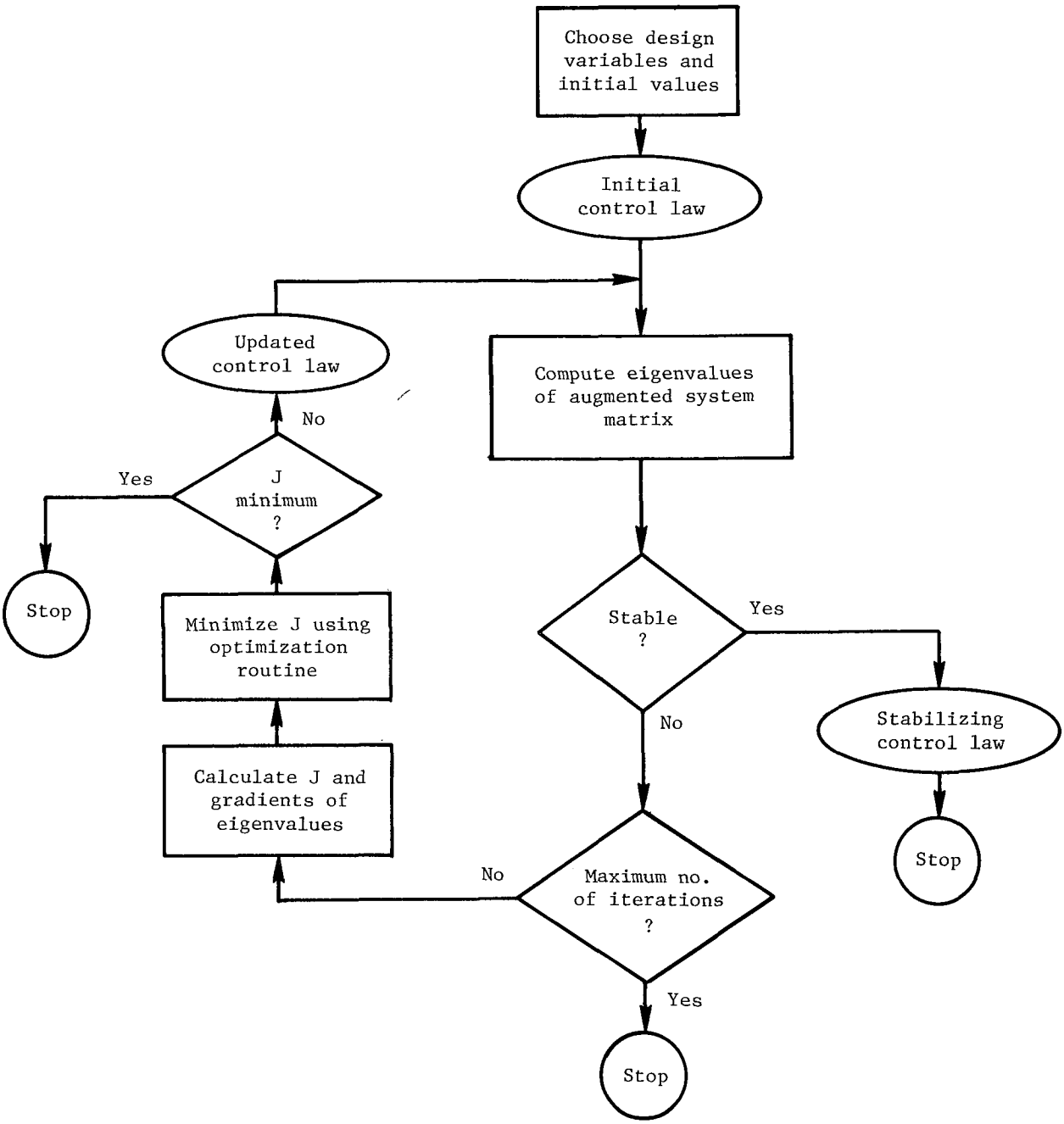
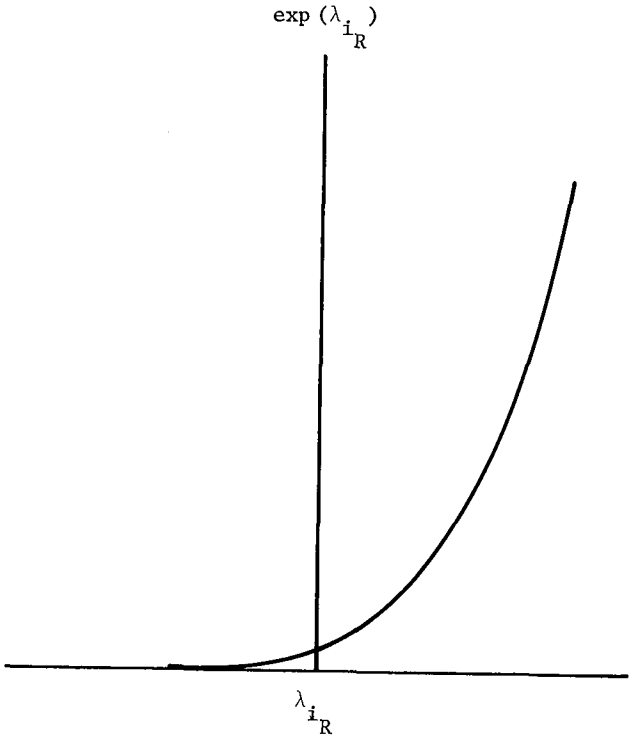
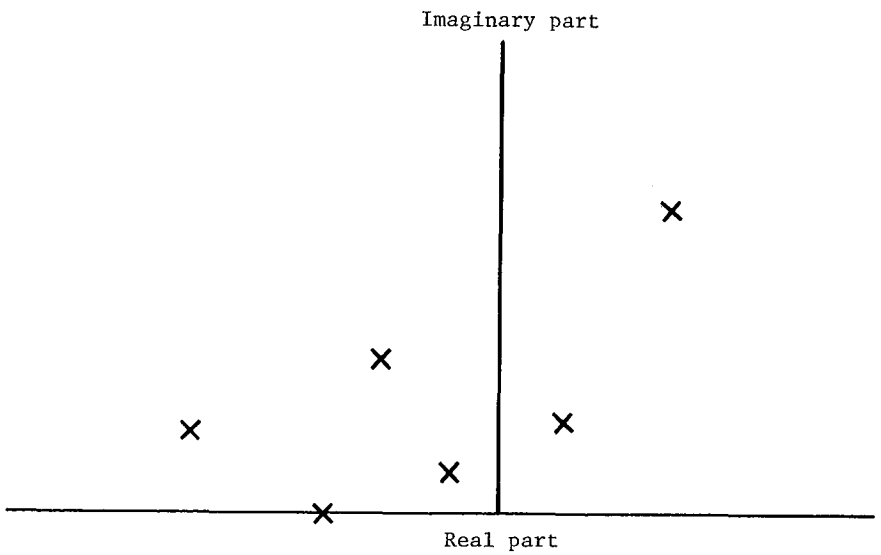


Figure 2. Flow chart of design algorithm.



(a) Sketch of exponential function.



(b) Sketch of eigenvalues.

Figure 3. Relationship between exponential function and eigenvalues.

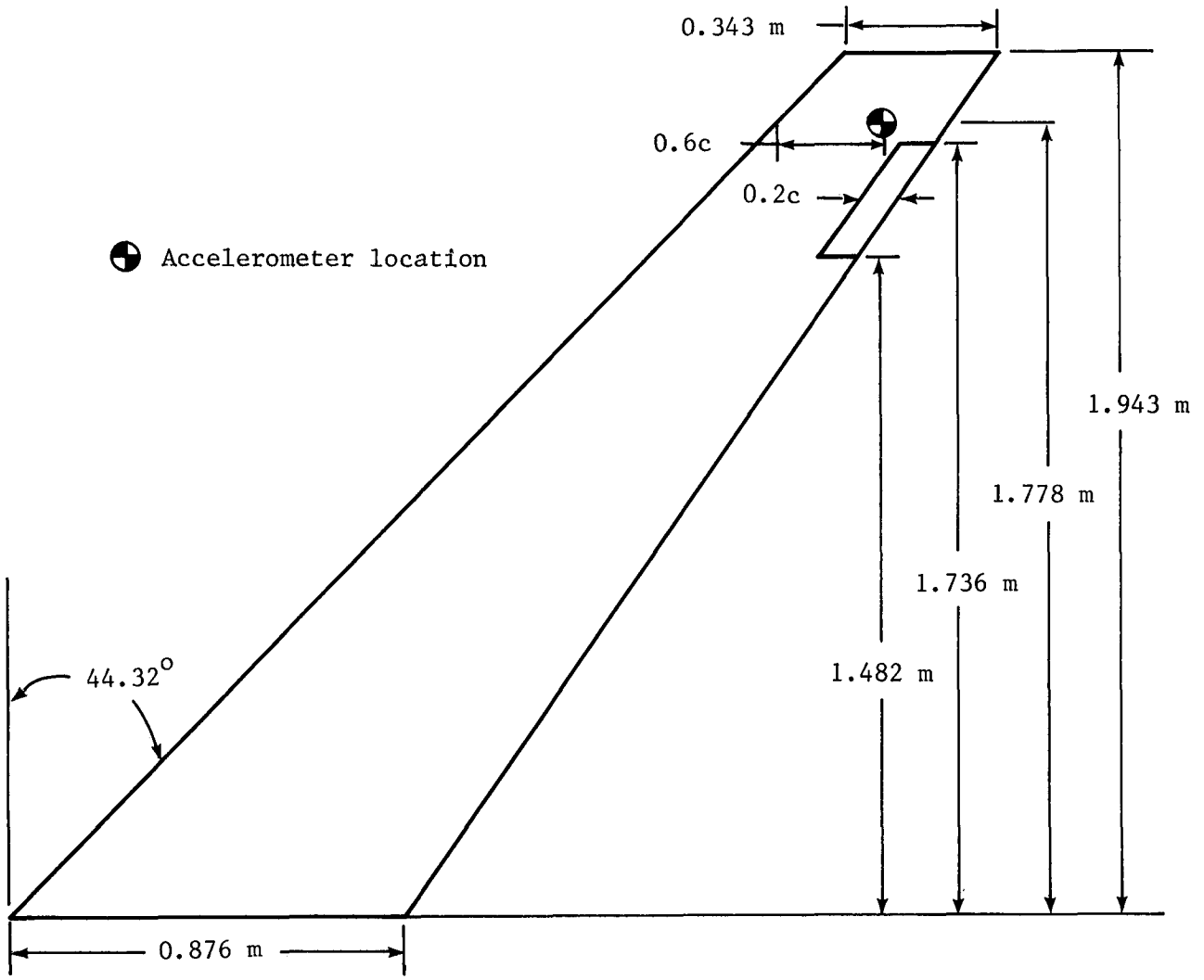


Figure 4. Model geometry showing sensor and control-surface locations of DAST ARW-1.

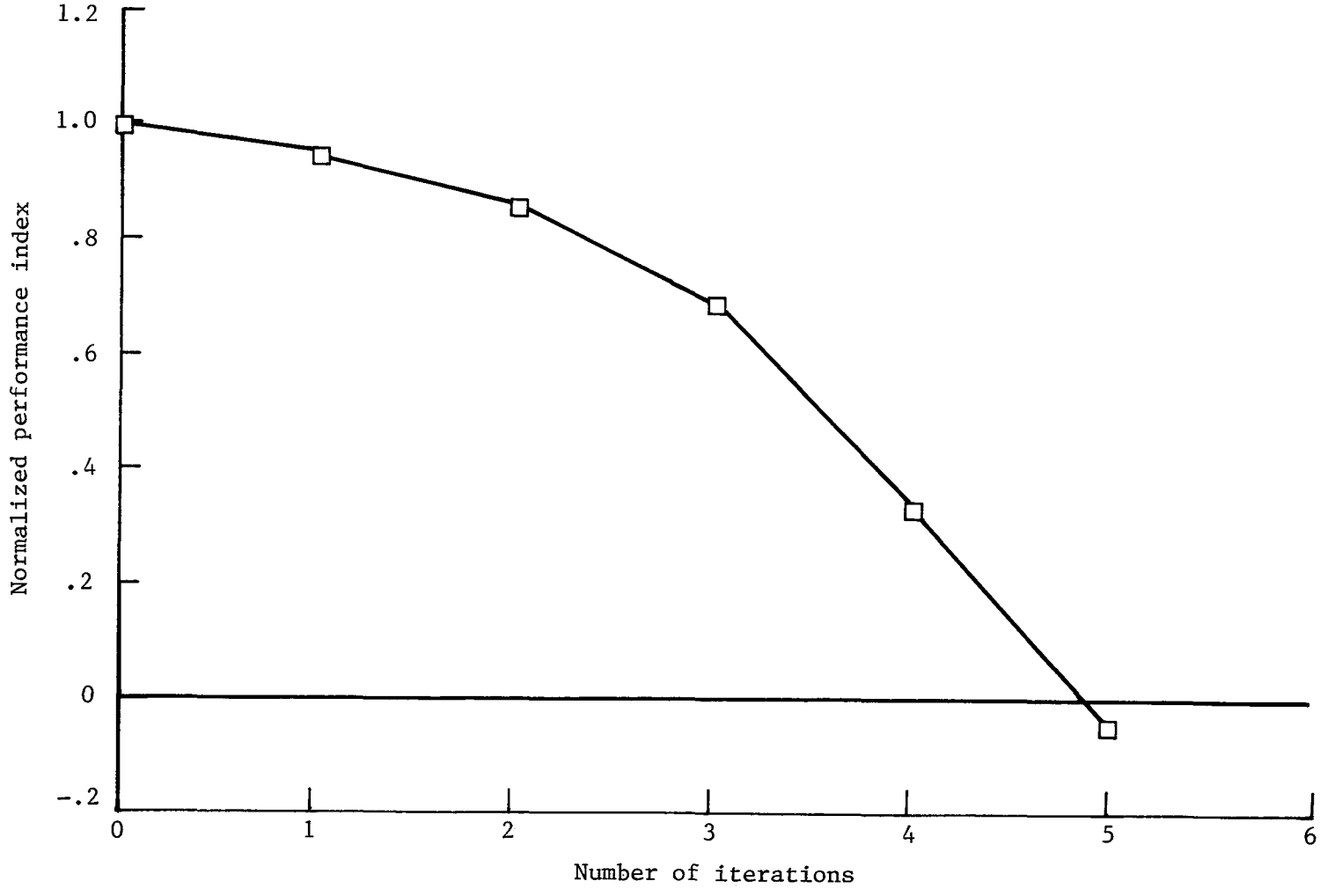


Figure 5. History of normalized performance index for first application—case 1. Normalization factor, 68.9798.

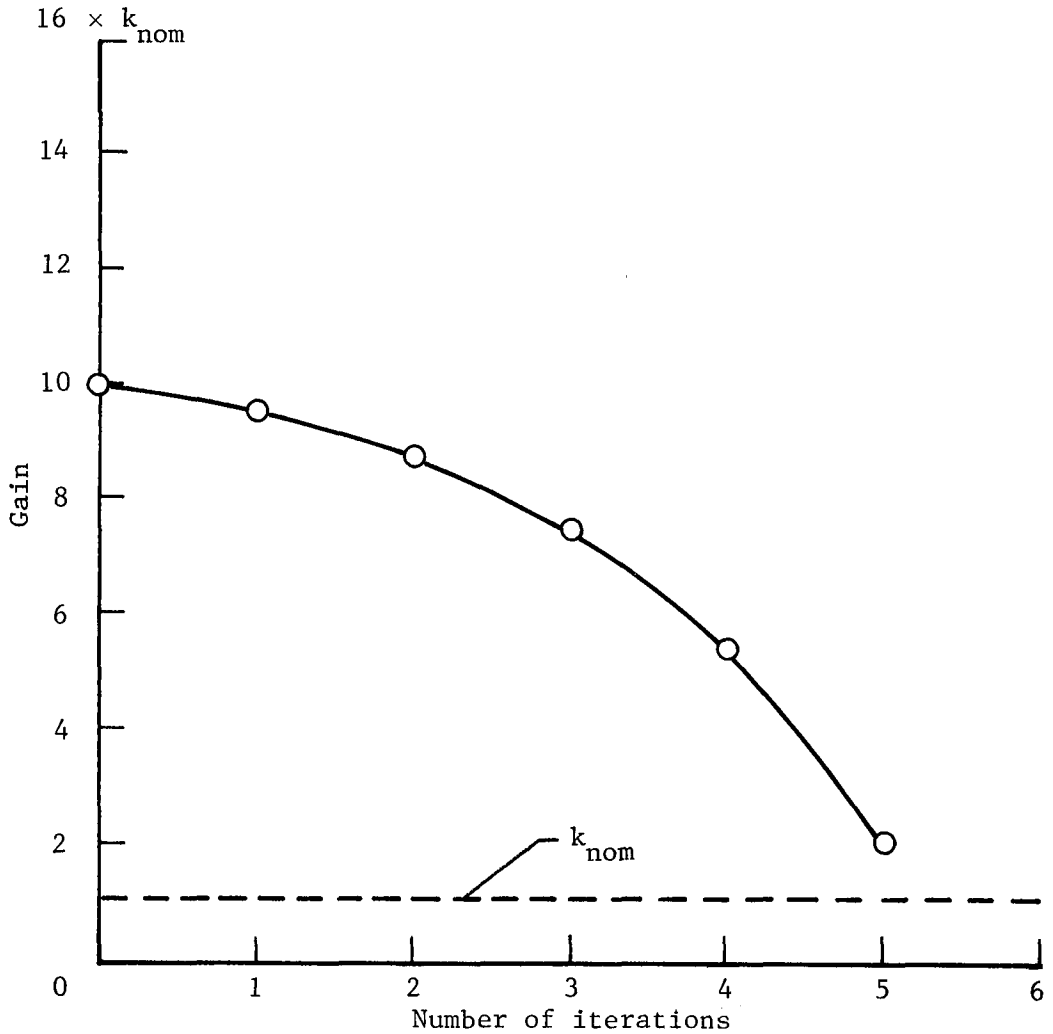


Figure 6. History of gain for first application—case 1.

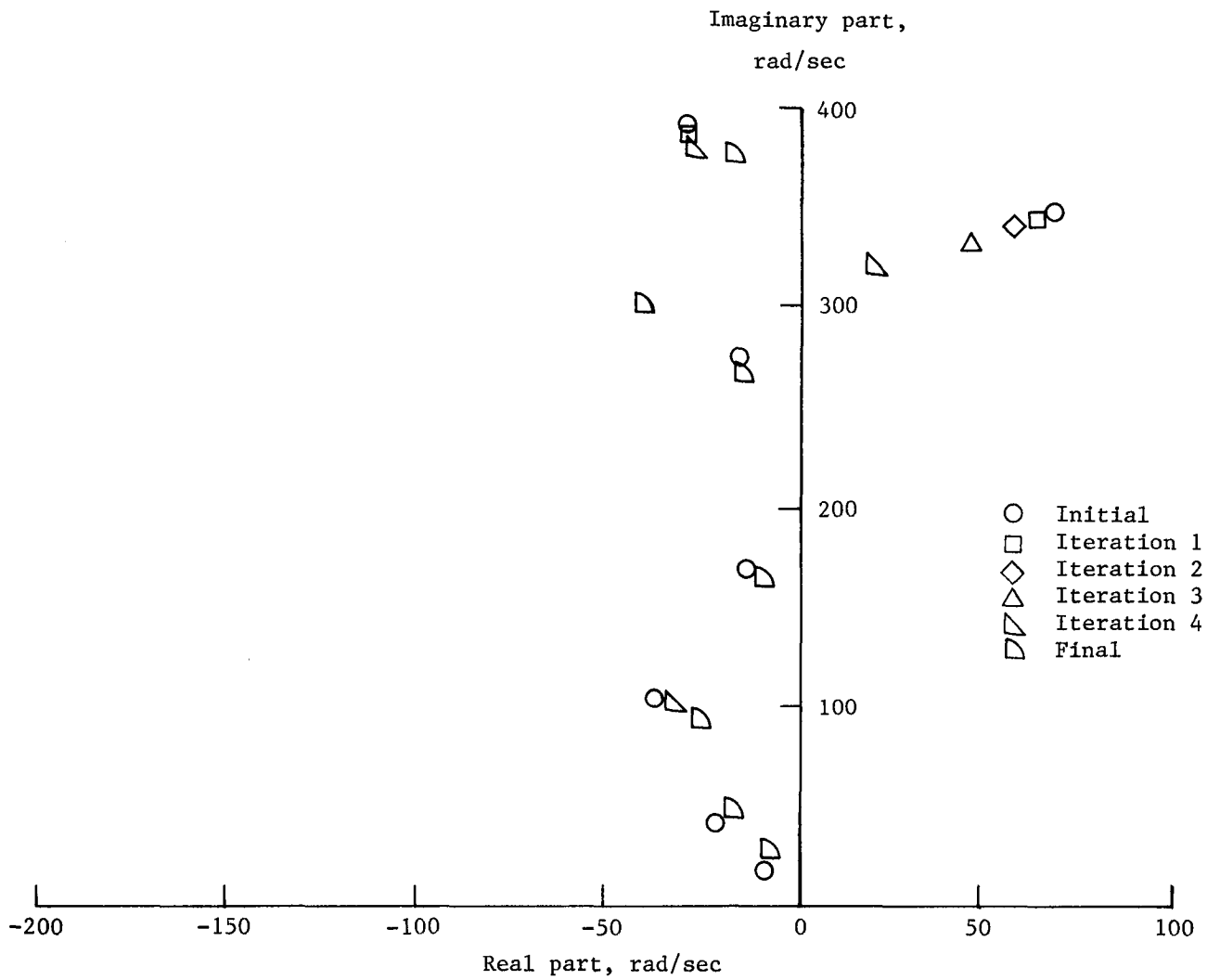


Figure 7. History of eigenvalues for first application—case 1.

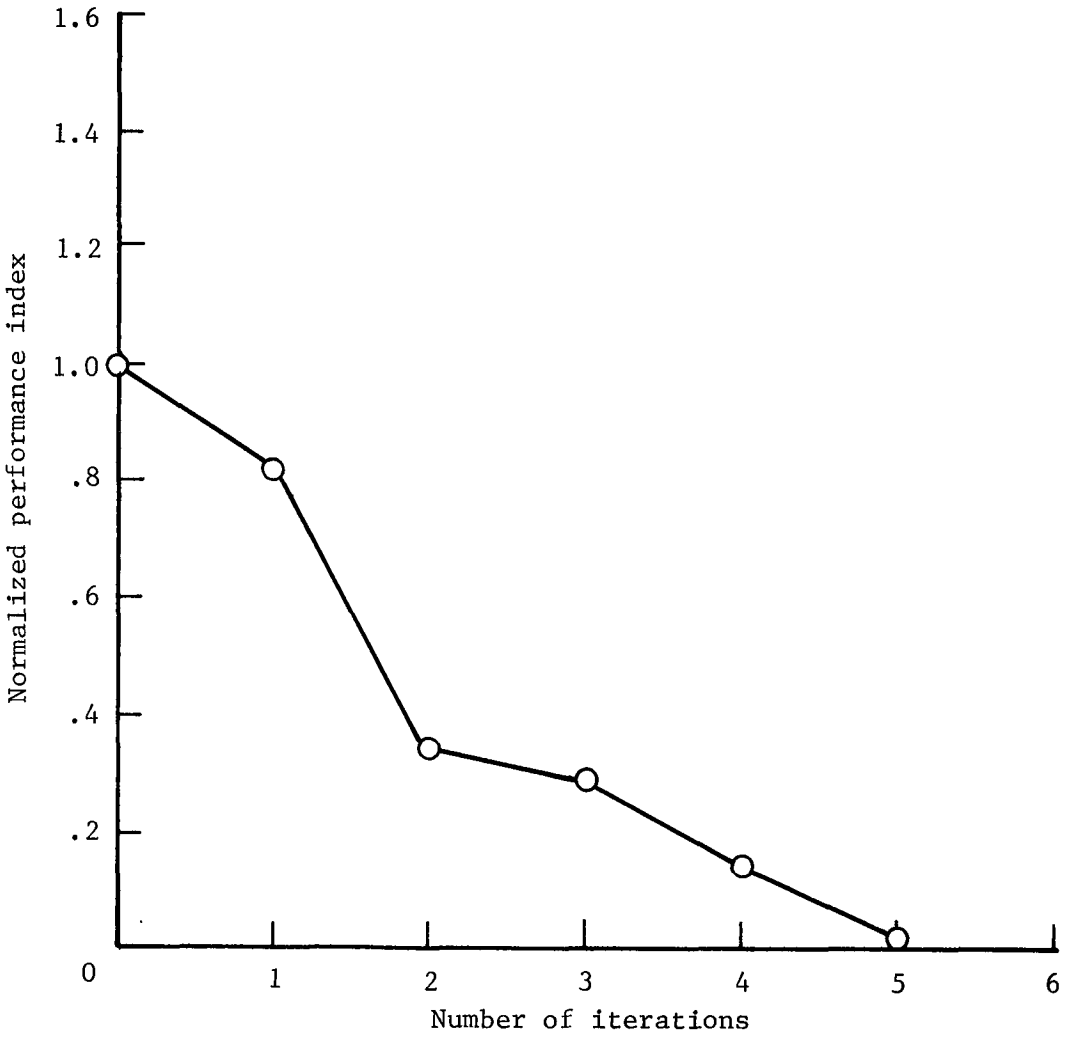


Figure 8. History of normalized performance index for first application—case 2. Normalization factor, 339.5172.

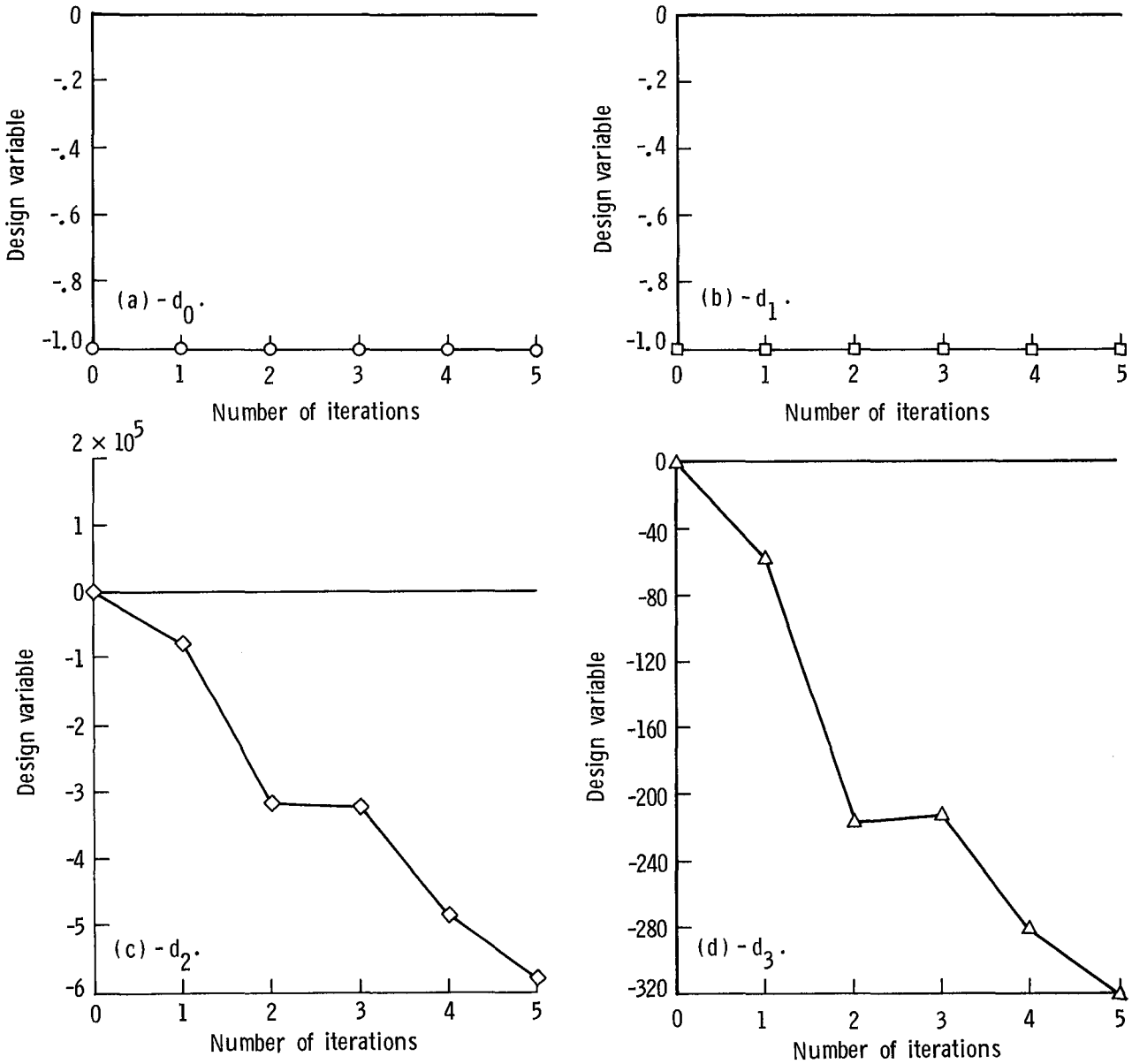


Figure 9. History of design variables for first application—case 2.

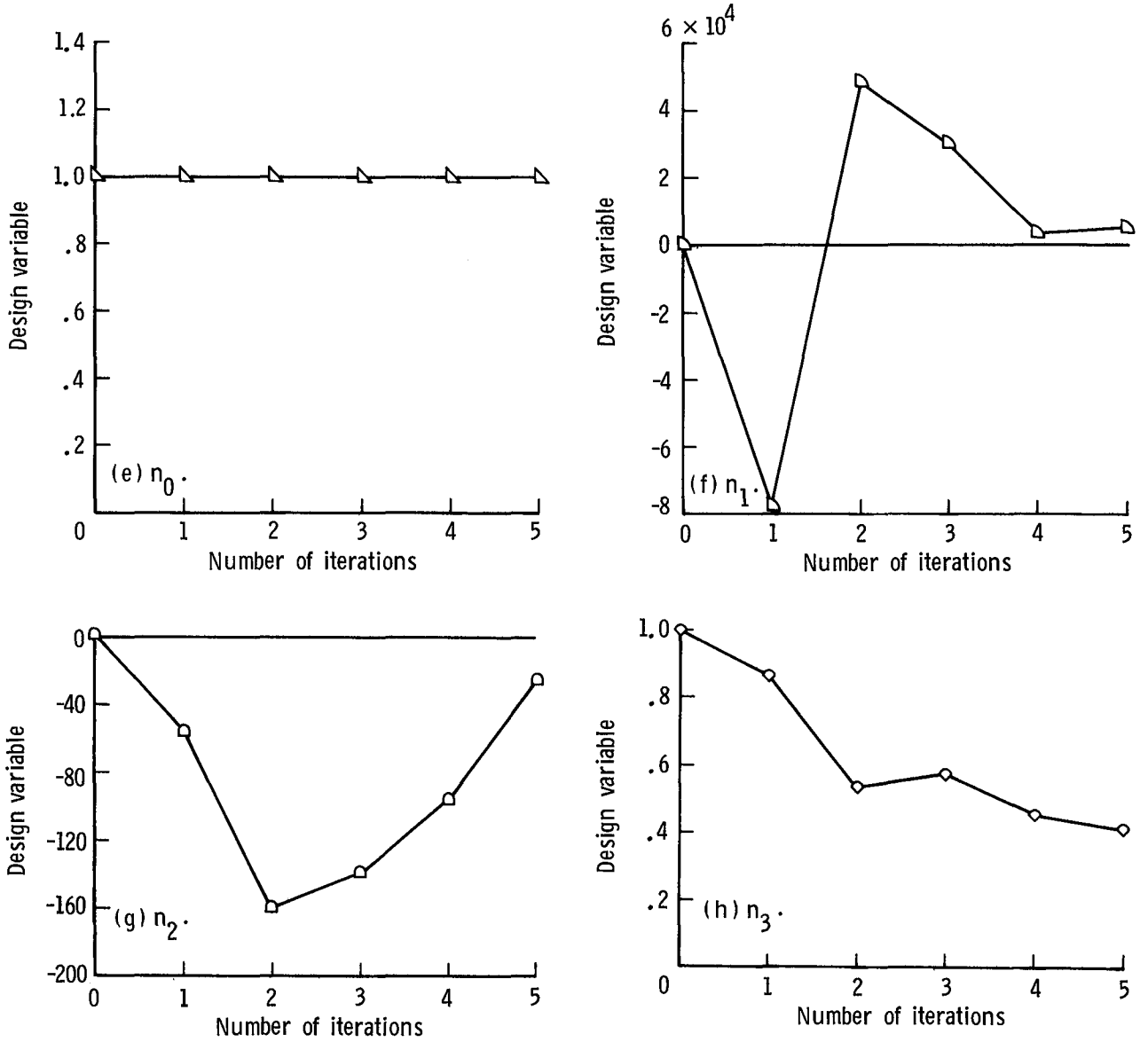


Figure 9. Concluded.

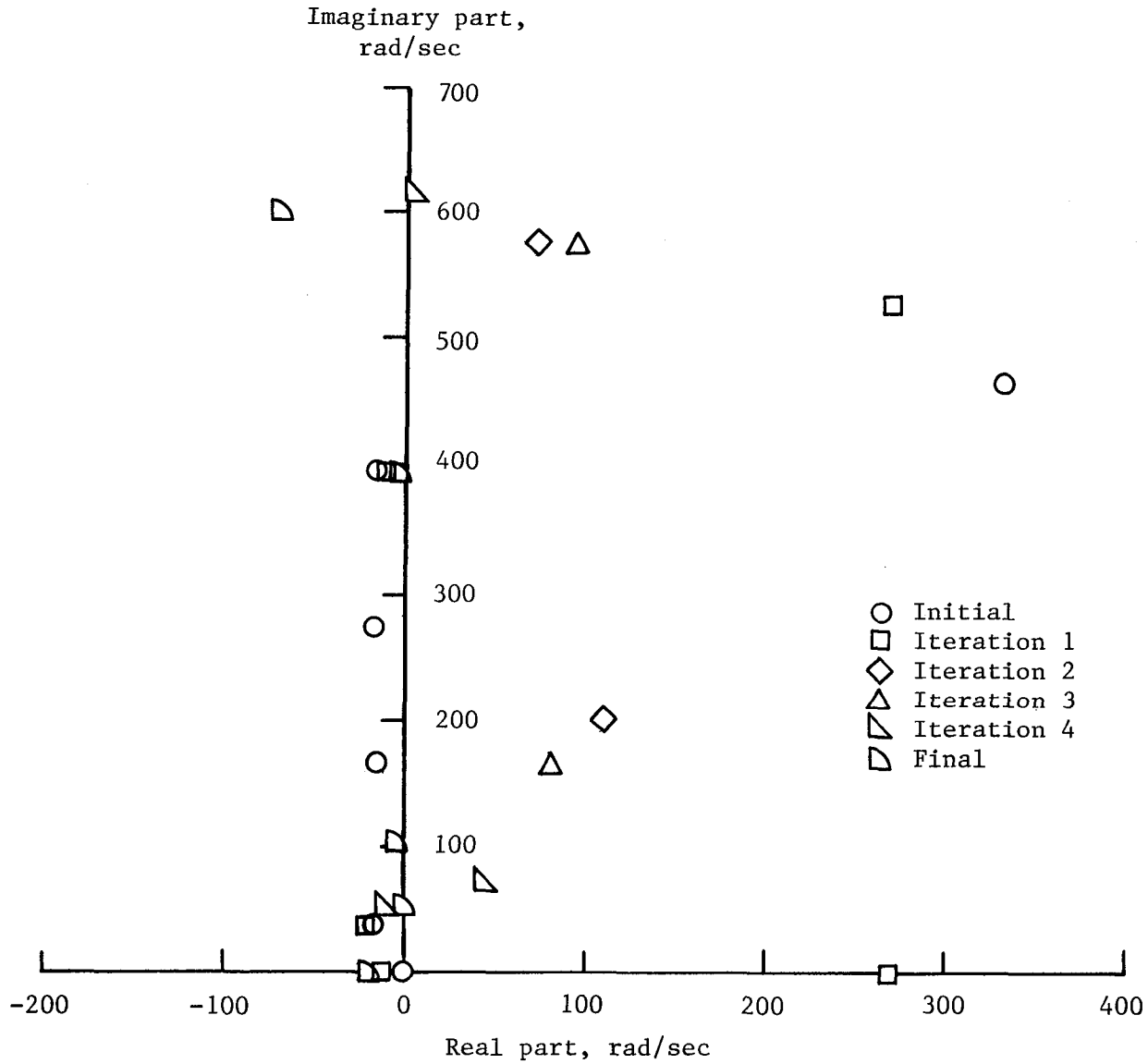


Figure 10. History of eigenvalues for first application—case 2.

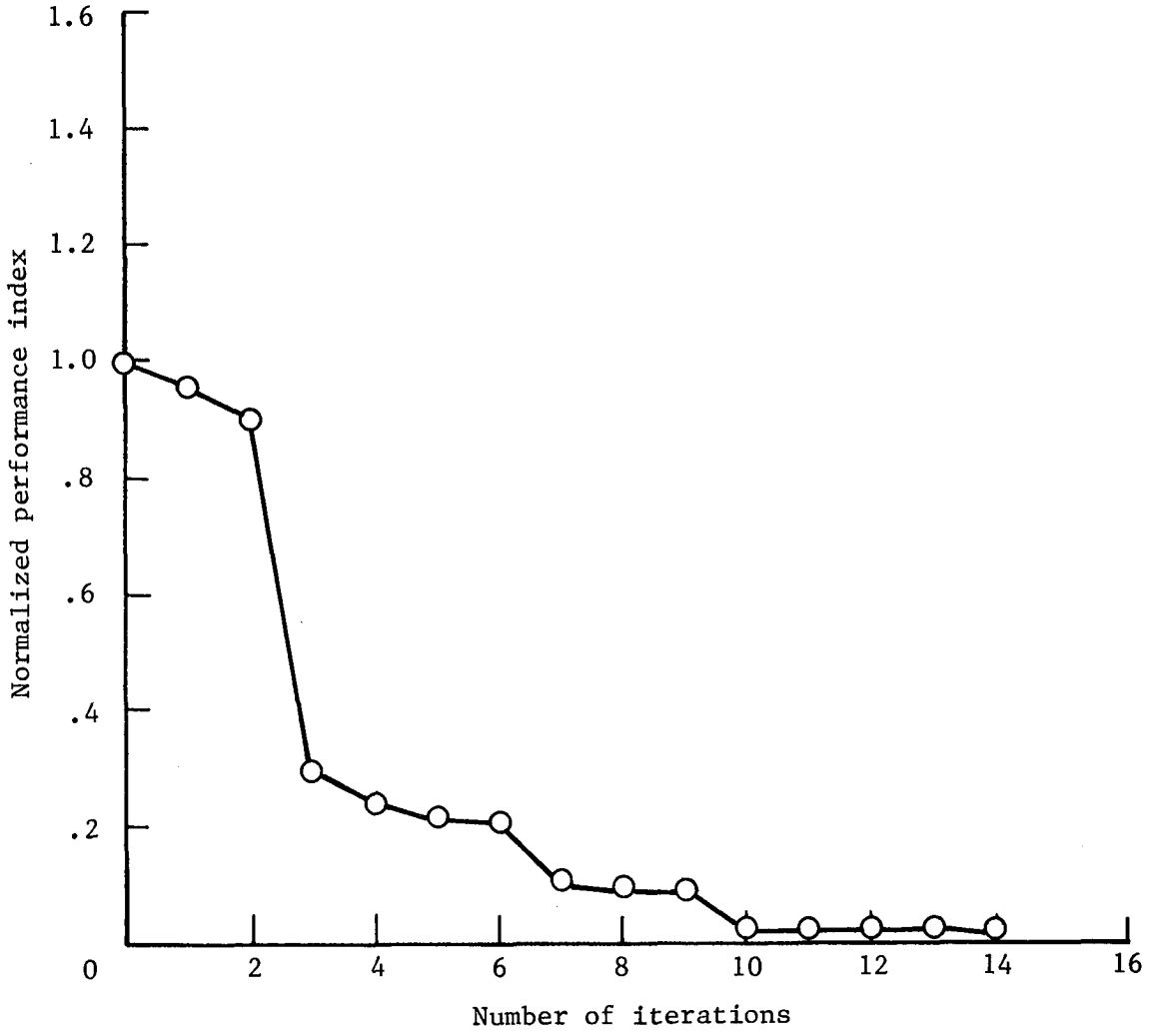


Figure 11. History of normalized performance index for first application—case 3. Normalization factor, 66.954.

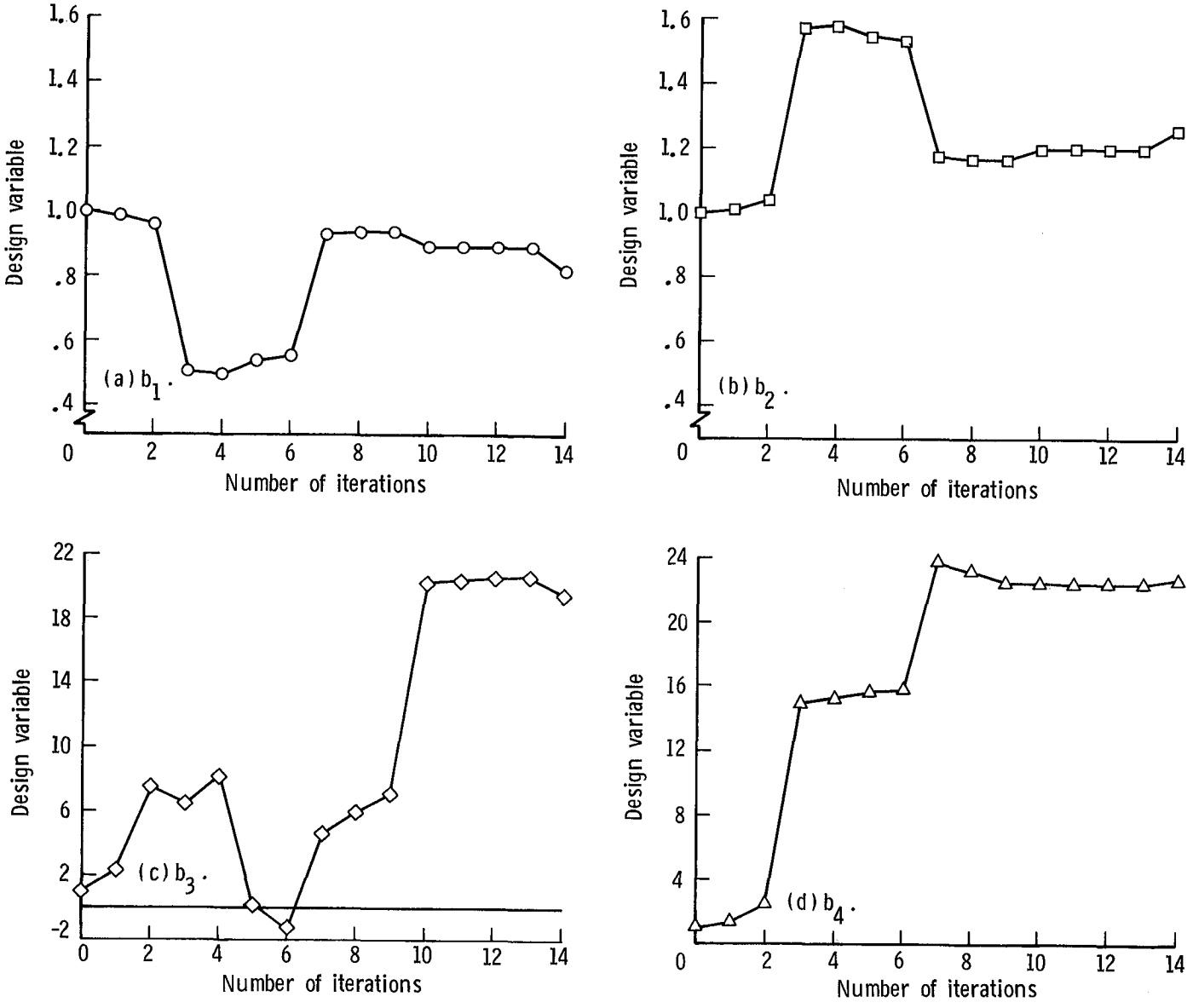


Figure 12. History of design variables for first application—case 3.

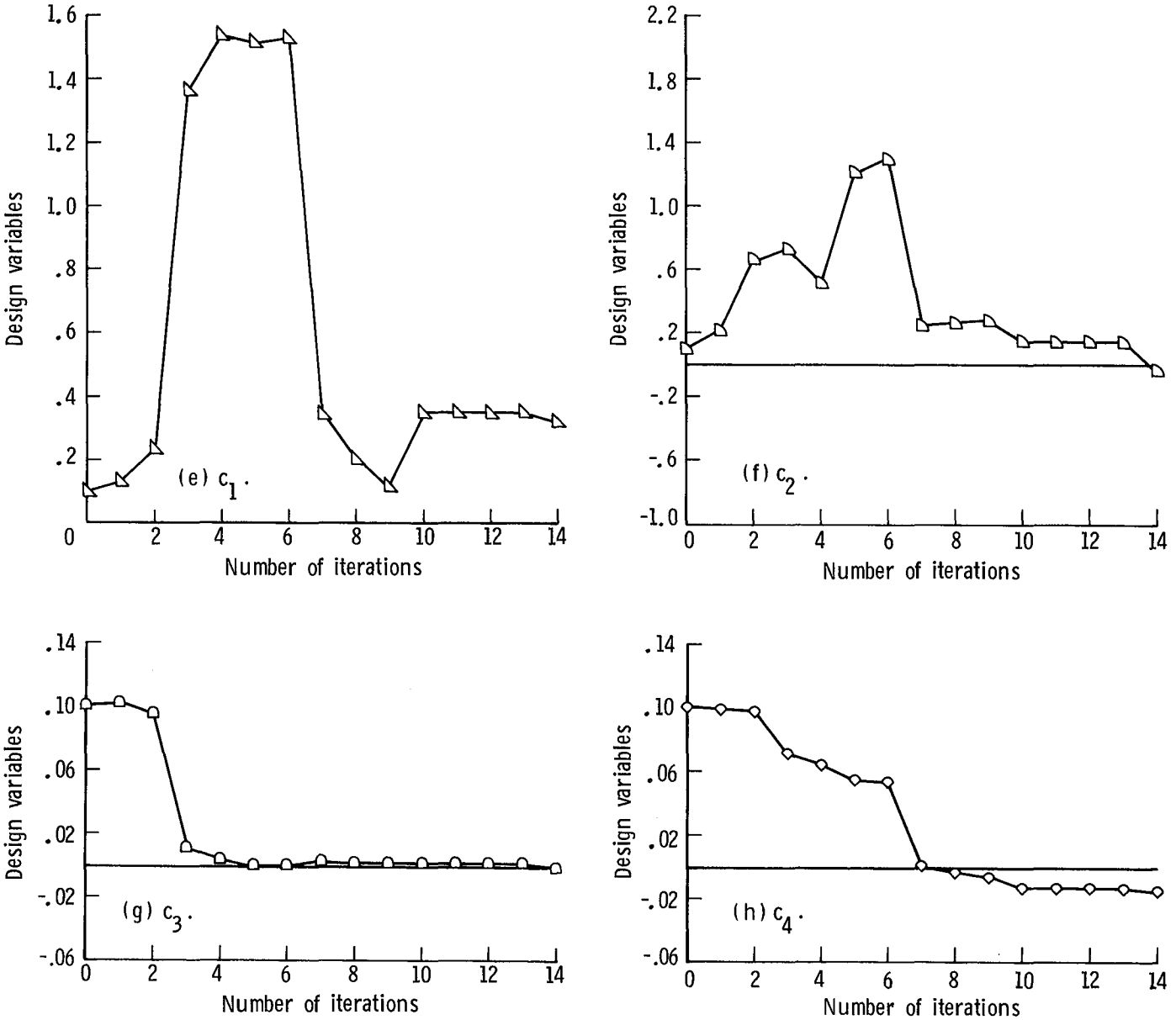
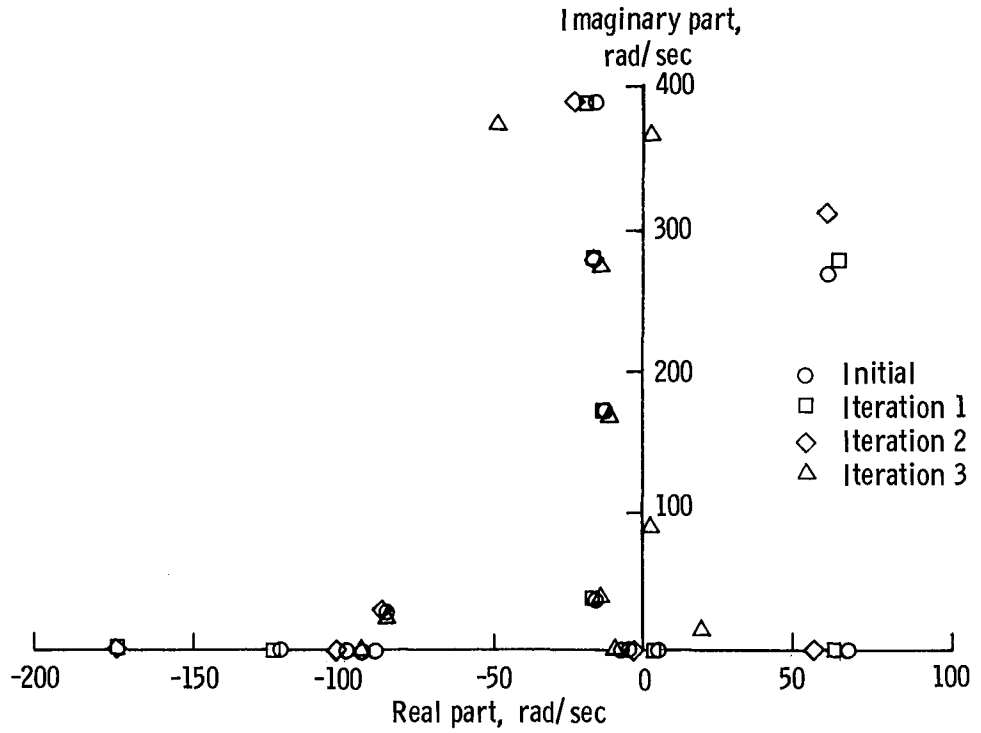
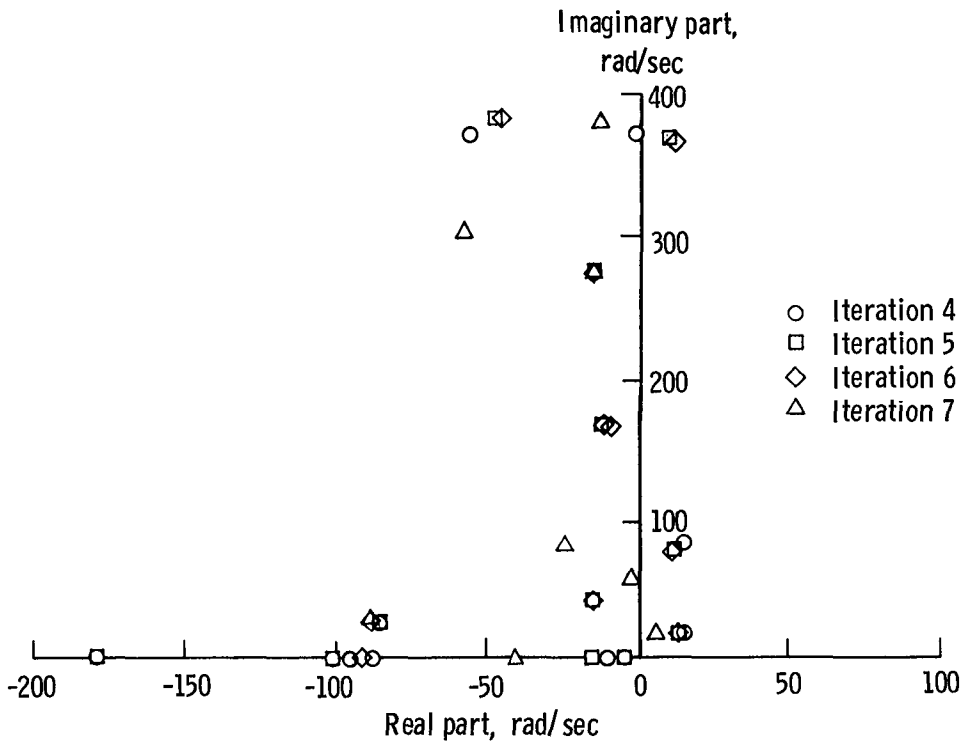


Figure 12. Concluded.

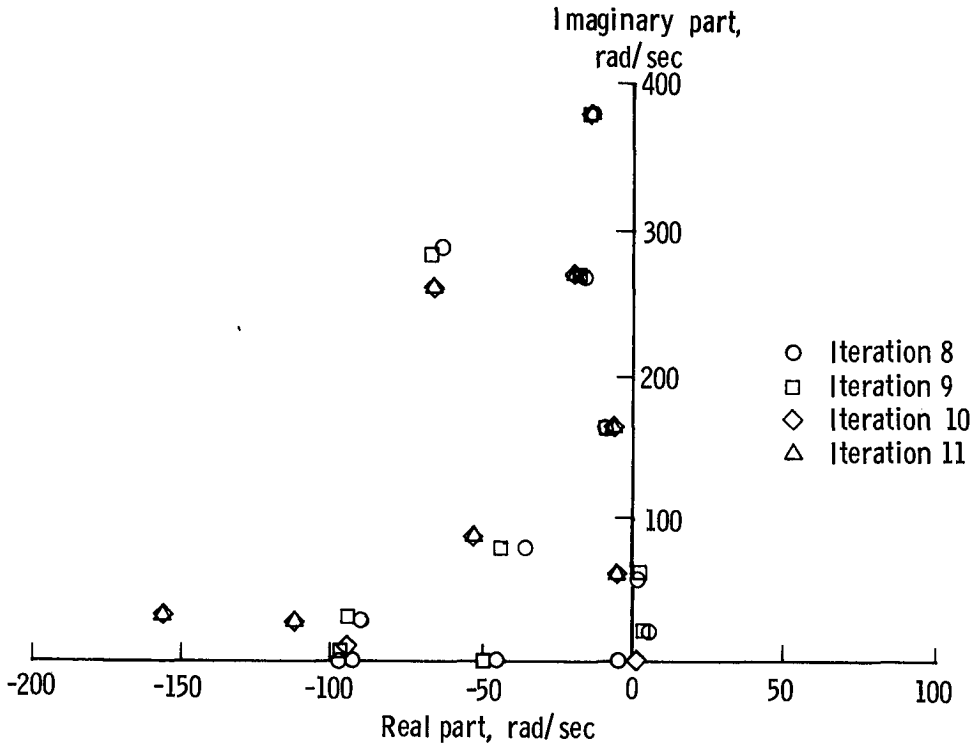


(a) Initial point and iterations 1 to 3.

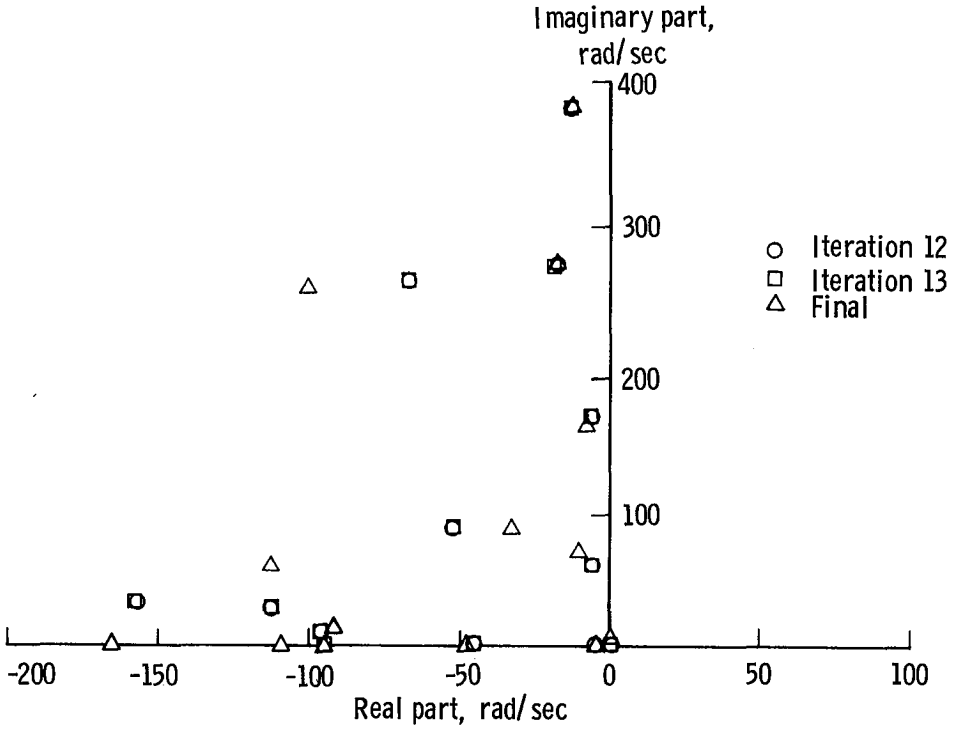


(b) Iterations 4 to 7.

Figure 13. History of eigenvalues for first application—case 3.



(c) Iterations 8 to 11.



(d) Final point and iterations 12 and 13.

Figure 13. Concluded.

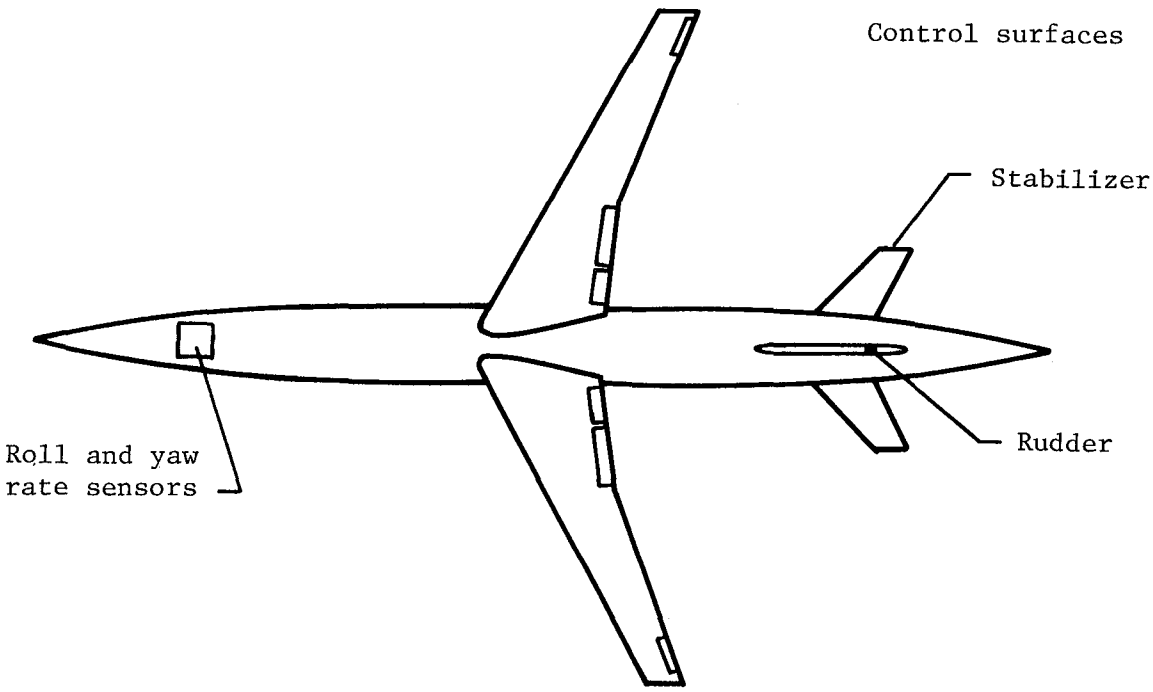


Figure 14. DAST ARW-2 sensor and control-surface locations for lateral-directional automatic flight control system (AFCS).

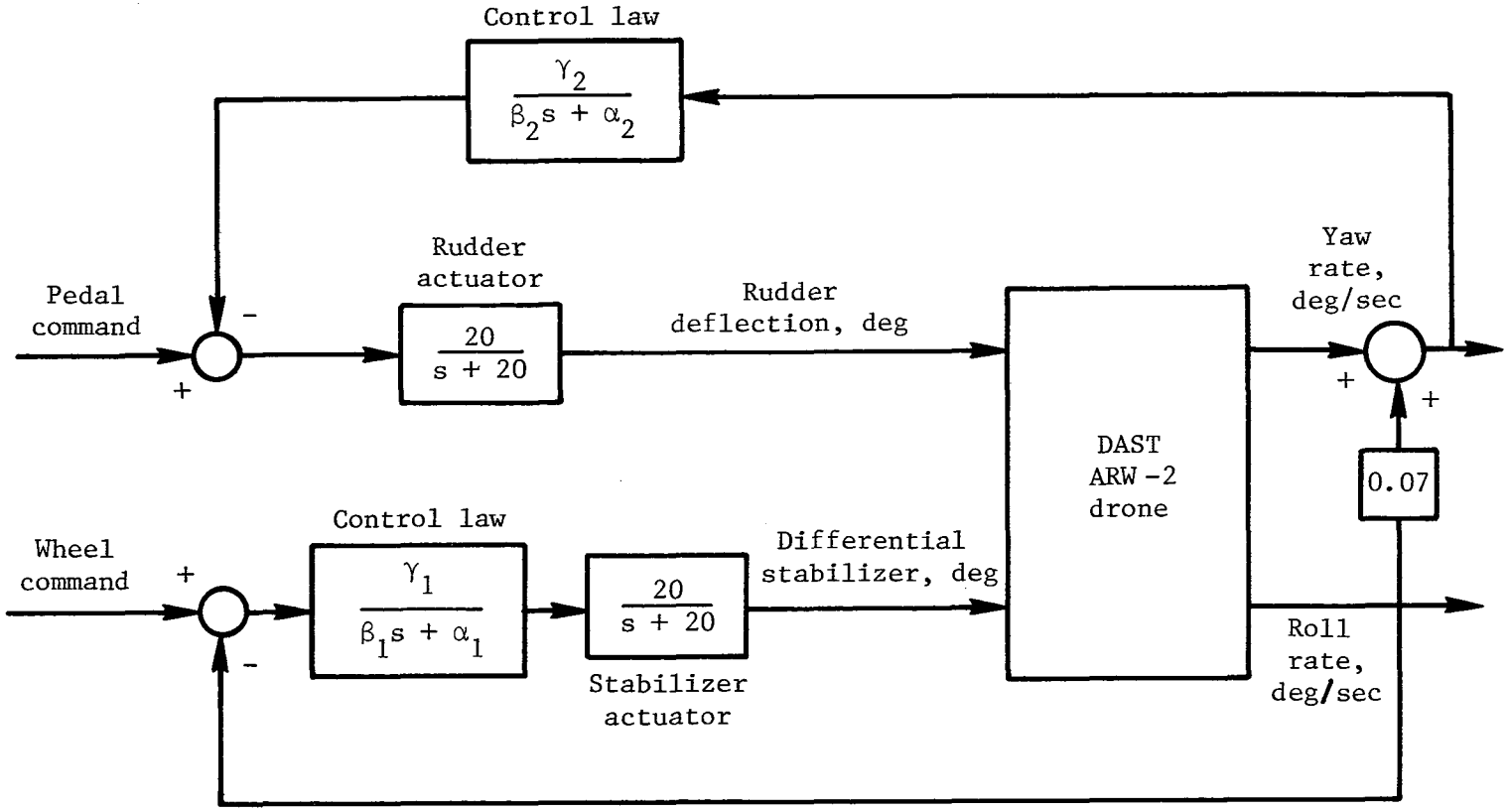


Figure 15. Lateral-directional AFCS block diagram for DAST ARW-2.

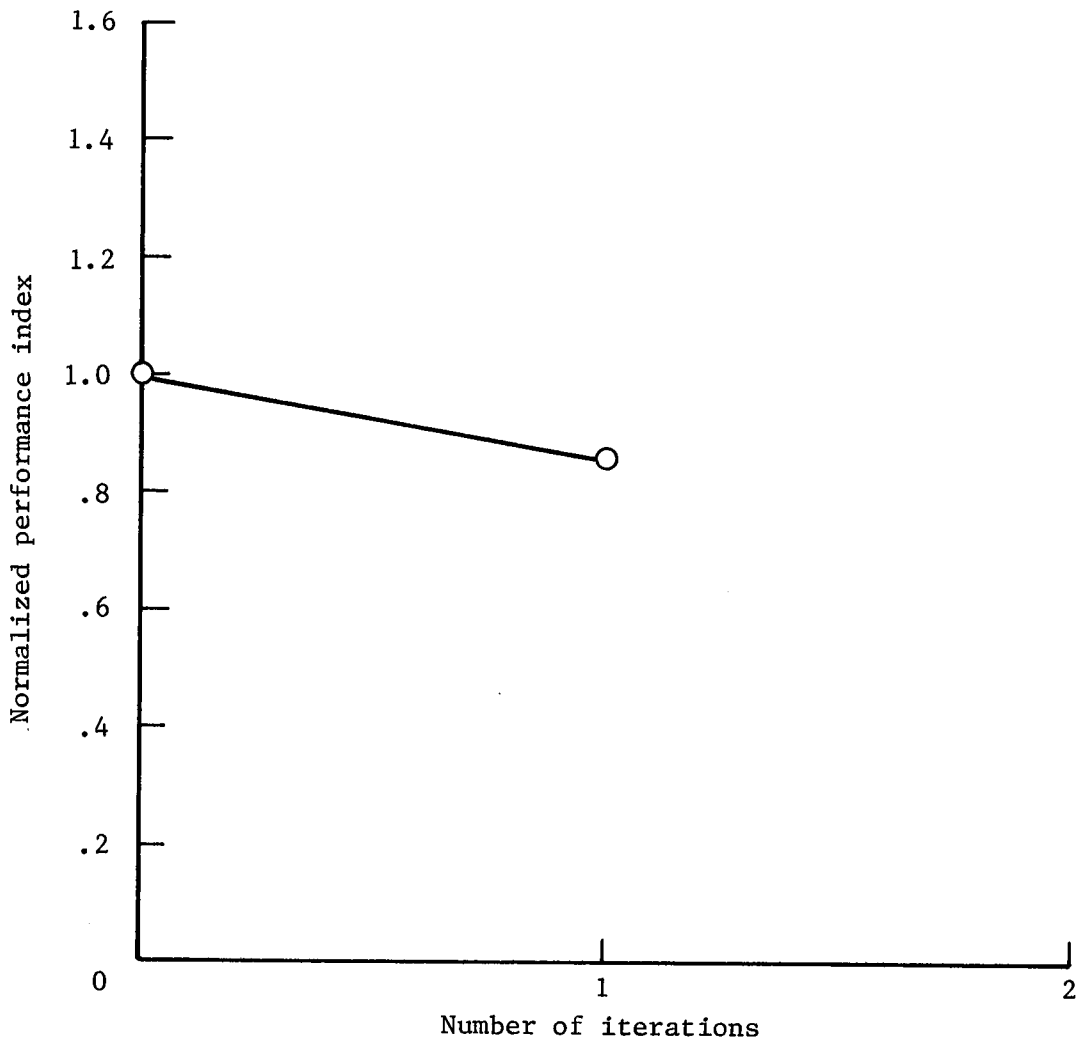


Figure 16. History of normalized performance index for application—case 1. Normalization factor, 1.6517.

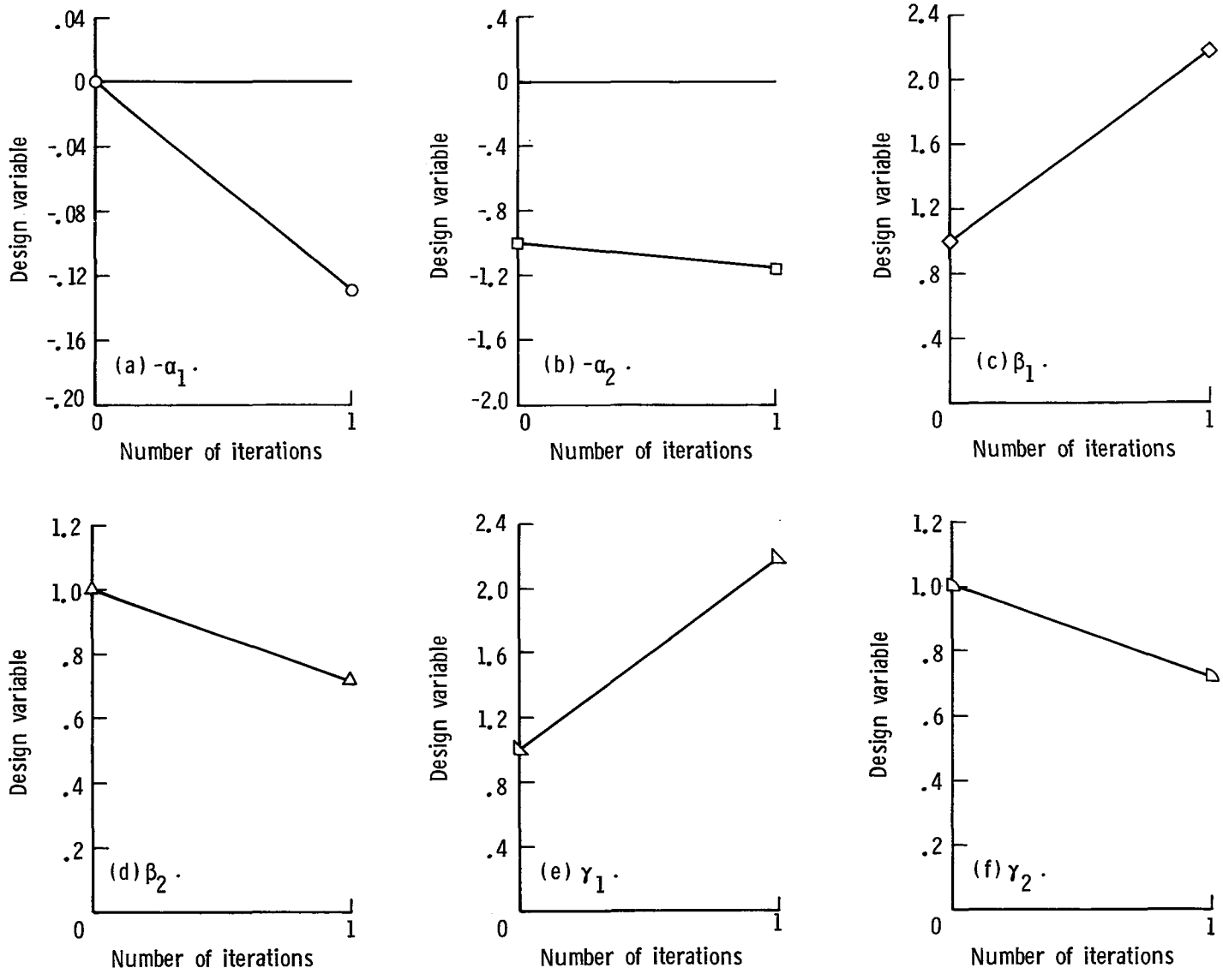


Figure 17. History of design variables for second application—case 1.

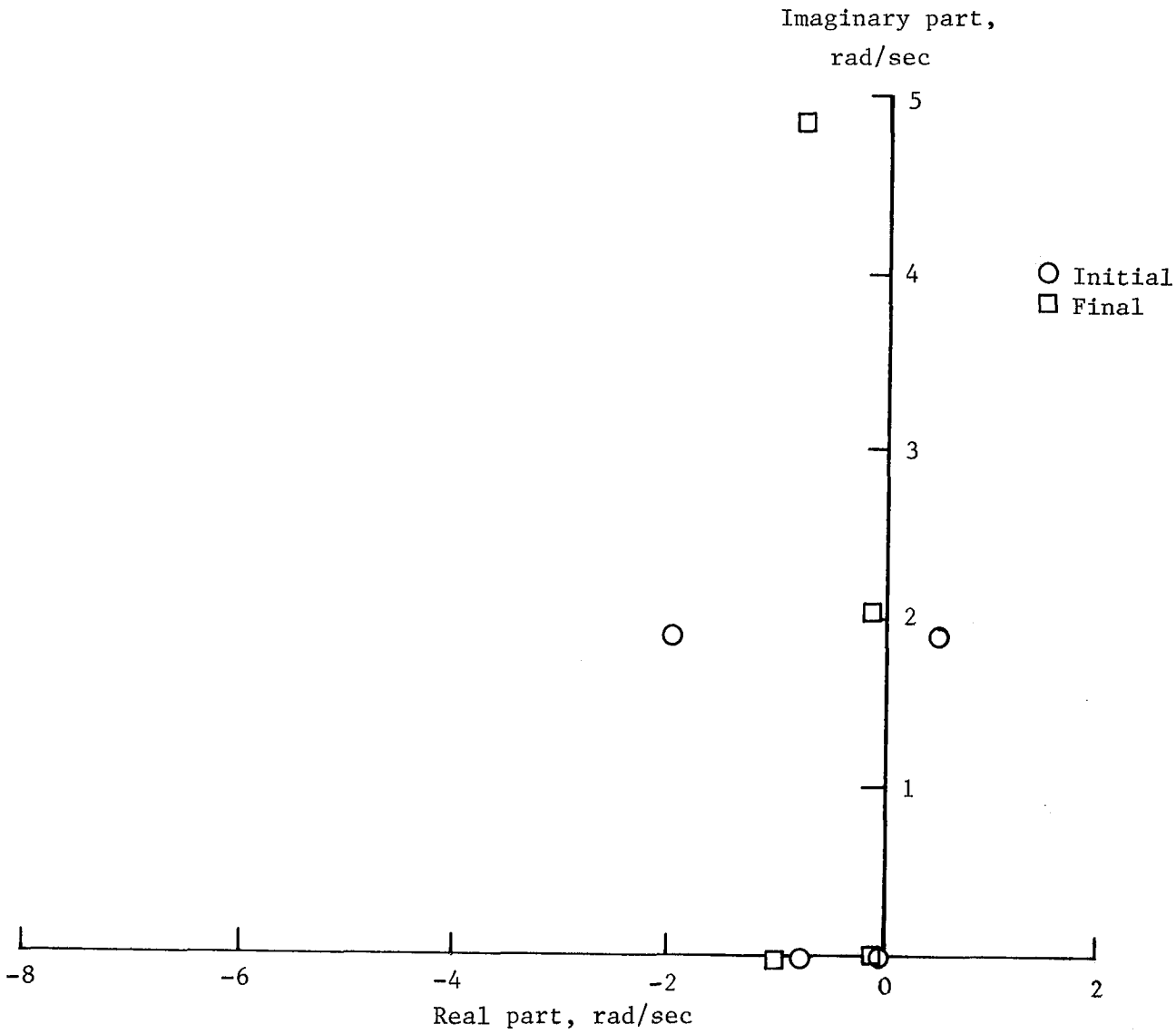


Figure 18. History of eigenvalues for second application—case 1.

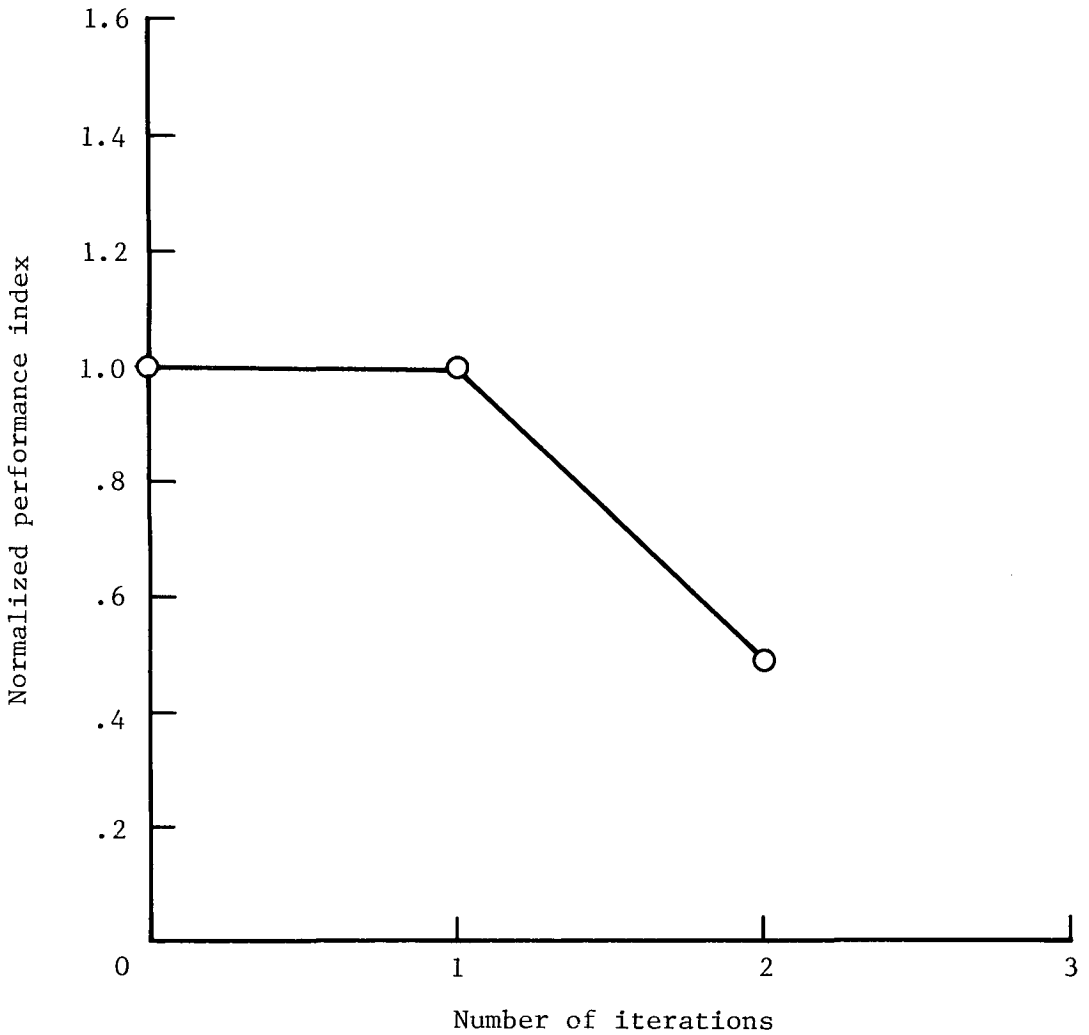


Figure 19. History of normalized performance index for second application—case 2. Normalization factor, 1.6898.

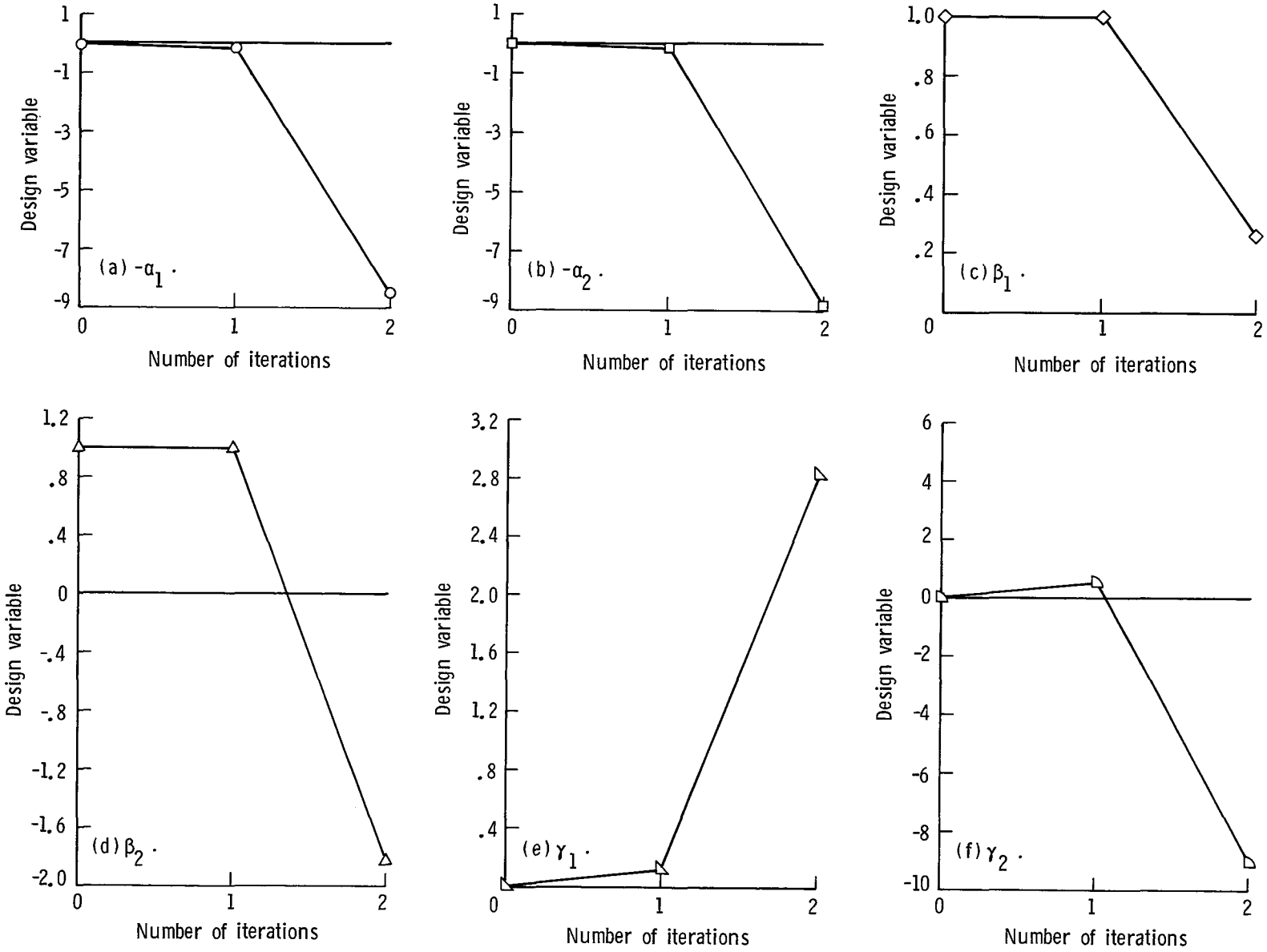


Figure 20. History of design variables for second application—case 2.

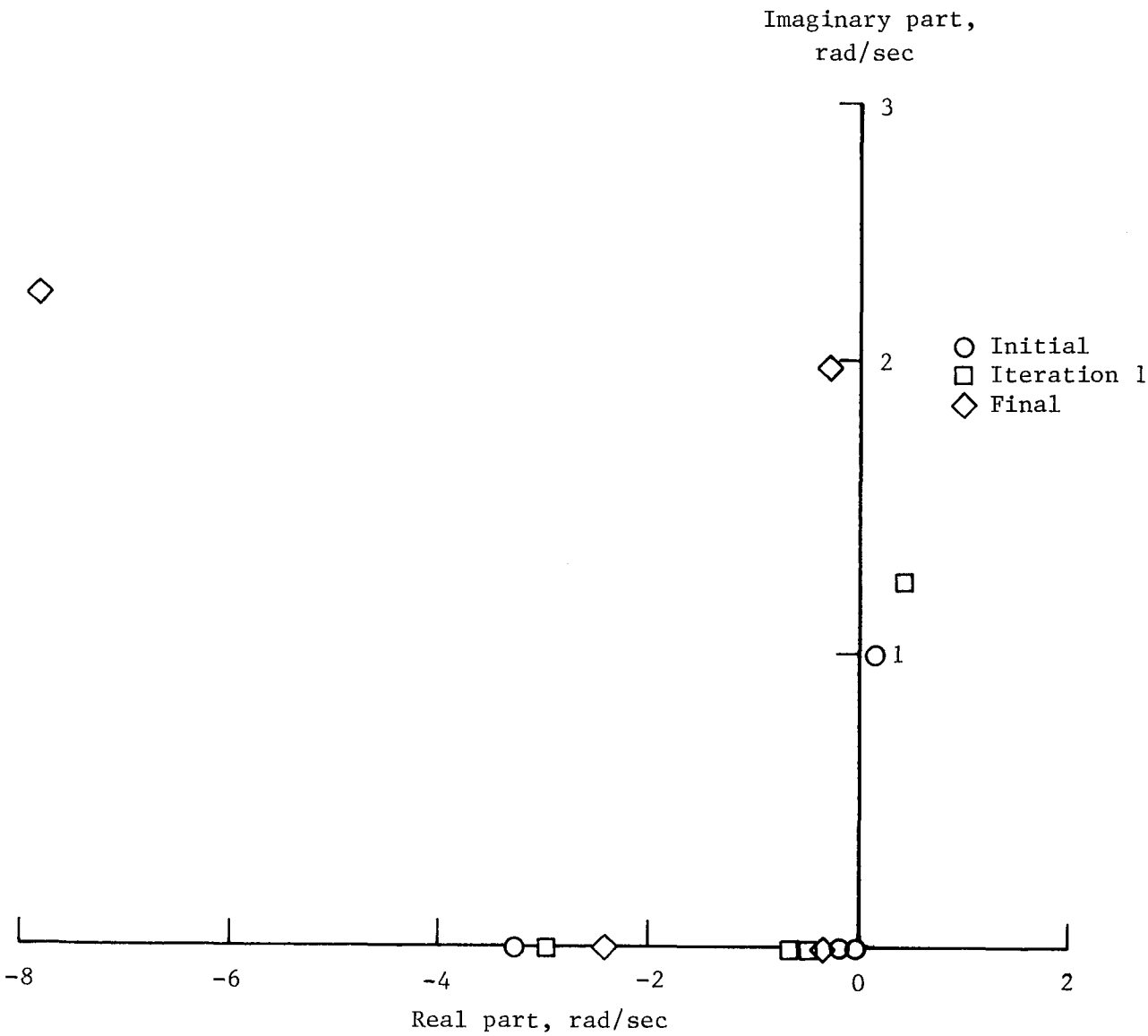


Figure 21. History of eigenvalues for second application—case 2.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TP-2479		

4. Title and Subtitle	5. Report Date
A Method To Stabilize Linear Systems Using Eigenvalue Gradient Information	November 1985
	6. Performing Organization Code: 505-33-43-13
7. Author(s): Carol D. Wieseman	8. Performing Organization Report No.: L-15964
9. Performing Organization Name and Address: NASA Langley Research Center, Hampton, VA 23665-5225	10. Work Unit No.
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, DC 20546-0001	13. Type of Report and Period Covered: Technical Paper
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Formal optimization methods and eigenvalue gradient information are used to develop a stabilizing control law for a closed-loop linear system that is initially unstable. The method was originally formulated by using direct, constrained optimization methods with the constraints being the real parts of the eigenvalues. However, because of problems in trying to achieve stabilizing control laws, the problem was reformulated to be solved differently. The method described in this paper uses the Davidon-Fletcher-Powell minimization technique to solve an indirect, constrained minimization problem in which the performance index is the Kreisselmeier-Steinhauser function of the real parts of all the eigenvalues. The method is applied successfully to solve two different problems: the determination of a fourth-order control law that stabilizes a single-input single-output active flutter suppression system and the determination of a second-order control law for a multi-input multi-output lateral-directional flight control system. Various sets of design variables and inital starting points were chosen to show the robustness of the method.

17. Key Words (Suggested by Authors(s))	18. Distribution Statement
Optimization Stability Sensitivities Control Eigenvalues	Unclassified—Unlimited Subject Category 63

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	38	A03

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1985

National Aeronautics and
Space Administration
Code NIT-3

Washington, D.C.
20546-0001

NASA

LIBRARY MATERIAL SLIP

DO NOT REMOVE SLIP FROM MATERIAL

Delete your name from this slip when returning material to the library.

NAME	DATE	MS
Kvaternik	4/02	340

NASA Langley Form 474 (Rev. Oct. 1999)