DESIGN OF ROBUST FEEDBACK CONTROL LAWS FOR HIGH-DIMENSIONED SYSTEMS

by

James P. Dunyak

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Engineering Mechanics

APPROVED:

S. L. Hendricks, Chairman

T. L. Herdman

D. T. Mook

A. Pap

August, 1987

Blacksburg, Virginia

Acknowledgements

The author wishes to thank his original advisor, Dr. John Junkins, for his guidance during the technical aspects of this work. He also wishes to thank Dr. Scott Hendricks for his assistance at the completion of this project. A special thanks also goes to Dr. Layne Watson and Mr. David Chichka for the extensive discussions which contributed to this thesis. Finally, the author expresses gratitude for the great patience of the Engineering Mechanics faculty and the staff and management at the Naval Research Laboratory.

Most especially, the author acknowledges the support and affection of his wife Alison during the overly long thesis effort. Her encouragement made this project possible.

Table of Contents

		Page
Acknowledgements		ii
Table of Contents		iii
List of Figures		v
List of Tables		viii
Chapter		
1	Introduction	1
2	Problem Formulation	5
	2.1 State Equations	5
	2.2 Eigenvalue Sensitivities	8
	2.3 Finite Element Model of a Bending Beam	16
3	The Optimization Method	27
	3.1 The Optimization Algorithm	27
	3.2 Using a Small Number of Sensors and Actuators	28
	3.3 Performance Measures in Noise	32
	3.4 A Measure of Robustness: Using Sensitivities	36
	3.5 Placing Both Real and Imaginary Components	40
4	The Continuation Method	45
	4.1 The Minimum Norm Algorithm	45
	4.2 A Robust Continuation Map	48
	4.3 Placing Eigenvalues With a Large Number of Gains	49
	4.4 Robust Design Through Sensitivities	52
5	Concluding Remarks	56
References		58

Table of Contents (continued)

		Page
Appendix		
A	Numerical Results	60
B	Optimization Routines Listings	123
C	Continuation Routines Listings	136
D	General Routines Listings	150
Vita		172

List of Figures

Figure		Page
2.1	A Flexible Beam	17
2.1	A Beam Element	19
2.3	Element Numbering Scheme	23
3.1	The Sensor and Actuator Configuration	29
3.2	The Sensor and Actuator Configuration	38
3.3	The Sensor and Actuator Configuration	42
A.1	Modes 1 and 2	62
A.2	Modes 3 and 4	63
A.3	Modes 5 and 6	64
A.4	Modes 7 and 8	65
A.5	Modes 9 and 10	66
A.6	Modes 11 and 12	67
A.7	Modes 13 and 14	68
A.8	Modes 15 and 16	69
A.9	Modes 17 and 18	70
A.10	Modes 19 and 20	71
A.11	Modes 21 and 22	72
A.12	Modes 23 and 24	73
A.13	Gain Plots	76
A.14	Cost Trajectory	77
A.15	Real Parts of Eigenvalues	78
A.16	Imaginary Parts of Eigenvalues	79
A.17	Gain Plots	82
A.18	Cost Trajectory	83

List of Figures (continued)

Figure		Page
A.19	Real Parts of Eigenvalues	84
A.20	Imaginary Parts of Eigenvalues	85
A.21	Cost Trajectory	88
A.22	Gain Plots	89
A.23	Gain Plots	90
A.24	Gain Plots	91
A.25	Gain Plots	92
A.26	Real Parts of Eigenvalues	93
A.27	Imaginary Parts of Eigenvalues	94
A.28	Cost Trajectories	97
A.29	Gain Plots	98
A.30	Gain Plots	99
A.31	Gain Plots	100
A.32	Gain Plots	101
A.33	Real Parts of Eigenvalues	102
A.34	Imaginary Parts of Eigenvalues	103
A.35	Cost Trajectories	106
A.36	Gain Plots	107
A.37	Gain Plots	108
A.38	Gain Plots	109
A.39	Gain Plots	110
A.40	Real Parts of Eigenvalues	111
A.41	Imaginary Parts of Eigenvalues	112
A.42	The Determinant	113

List of Figures (continued)

Figure		Page
A.43	Cost Trajectories	116
A.44	Gain Plots	117
A.45	Gain Plots	118
A.46	Gain Plots	119
A.47	Gain Plots	120
A.48	Real Parts of Eigenvalues	121
A.49	Imaginary Parts of Eigenvalues	122

List of Tables

Table		Page
2.1	Beam Physical Constants	26
A.1	Approximate and Exact Open Loop Frequencies	61
A.2	Gains and Parameters	74
A.3	Eigenvalue Initial and Final States	75
A.4	Gains and Parameters	80
A.5	Eigenvalue Initial and Final States	81
A.6	Gains and Parameters	86
A.7	Eigenvalue Initial and Final States	87
A.8	Gains and Parameters	95
A.9	Eigenvalue Initial and Final States	96
A.10	Gains and Parameters	104
A.11	Eigenvalue Initial and Final States	105
A.12	Gains and Parameters	114
A.13	Eigenvalue Initial and Final States	115

Chapter One

Introduction

The development of the space shuttle has opened the door to the use of space. Popular literature is full of discussions of space stations, space based telescopes, solar power satellites the size of Manhattan, and space manufacturing of new medicines and alloys. In fact, ready transportation into orbit has opened possibilities for many new military and civilian applications. But it has also created some challenging technical problems. Because of severe payload constraints, the large space structures must be manufactured with a minimum of material. These highly flexible structures may not be able to bear their own weight in an Earth environment, so ground testing will be limited. And the structures will face stringent specifications for attitude control and vibration suppression. The design of active controllers for flexible structures is a difficult problem. Modeling of the distributed parameter system is usually accomplished by finite element analysis, and further model order reduction is often required for controller design. This results in a high dimension, discrete linear model. This model is used to develop control strategies which hopefully control the full distributed system. A discussion of the entire development process of a control law is beyond the scope of any single document. Instead, this thesis concentrates on one part of the process, the design of controls for a high dimensioned linear system. This alone can be a substantial task.

As described in reference [1], a number of methods are available for designing control laws for high-dimensioned linear systems. However, many of these methods pose serious limitations, such as requiring observers and requiring an actuator for each mode controlled. The design of observers for flexible structures is a formidable task, and most practical applications provide strict limitations on the number of actuators available. These requirements are relaxed when applying the eigenspace optimization and constraint ideas presented here. A major simplification occurs through use of only linear output feedback,

since output feedback control does not use an observer. Output feedback control of flexible structures for pole placement is discussed extensively in reference [2], but the control laws still require an actuator per mode controlled. Pole placement using output feedback control is also discussed in references [3,4]. Reference [5] discusses feedback of only velocity measurements, and shows that velocity feedback can be used to guarantee a stability margin. Finally, references [6,7] discuss the interaction between flexible structures and controls using output feedback based on reduced order models.

Another important simplification results from the design methodology. There is no a priori assumption that an actuator is required for each controlled mode. One or more performance measures are chosen which allow rather versatile mission specific measures of robustness to be included in the design process. Functions of eigenvalues, eigenvectors, sensitivities and other robustness measures are allowed. A large fraction of practical applications have disturbance models and robustness requirements which are easily described in the frequency domain, so the eigenspace approach easily accommodates these constraints and performance measures. The design methodology is similar in philosophy, though different in implementation, to reference [8]. In this reference, complex systems are designed through the use of optimization procedures. Starting from a nominal design, parameters are adjusted to improve performance. This method has been particularly successful in improving designs of complicated circuits [Ref. 9]. The methods proposed in references [8,9] and here have great flexibility in performance measure, but this flexibility has a cost. No guarantee is made that the final design is satisfactory. Instead, the designer is "in the loop" of the design process. He proposes a performance measure and nominal design, applies the algorithms and checks the final design. If the design is not adequate, he changes the configuration or performance measure and repeats the procedure. This interaction allows the final design to meet a variety of design goals.

Two ideas are discussed in detail in this thesis: design by optimization of a

performance measure, and design by constraint of a set of performance measures. The author's paper, "Eigenspace Optimization for High-Dimensioned Feedback Control Systems," in reference [10], contains a description of the earlier portions of this work. Control law design for low-dimension linear systems by eigenspace or parameter optimization has also been considered in references [11,12,13]. Reference [14] illustrates a method of choosing output feedback gains, sensor positions, and actuator positions to maximize dissipated energy of a flexible structure.

The second design method is inspired by the success of relaxation methods in solving difficult numerical problems in satellite attitude control [Ref. 15]. In relaxation methods, the numerical problem is imbedded in a set of closely related problems. The first of the set is easily solved, and the solution is used as an initial guess for the second problem. Hopefully, the solution to the first problem is "close" to the solution to the second problem, and the numerical algorithm will converge. This is continued until the final problem is solved. In our case, a set of nonlinear constraints is to be solved. As shown in the author's paper in reference [16], the Chow-Yorke homotopy algorithm offers promise of large regions of convergence for nonlinear problems. The Chow-Yorke algorithm is discussed in references [17,18]. Failure to generalize the Chow-Yorke algorithm to the underconstrained problem is one of the major disappointments of this work, but investigation of the ideas behind Chow-Yorke leads to Smale's continuation algorithm [Ref. 19]. In Smale's algorithm, the sequence of problems from the relaxation method is replaced by a differential equation. Smale's algorithm is extended by the author to a more numerically robust continuation map.

This thesis is organized with the analysis in the chapters and the detailed numerical results in the appendices. .Chapter 2 presents the problem formulation and develops the formulas common to the remaining chapters. Chapter 3 discusses the optimization design method, while Chapter 4 discusses the continuation method. Chapter 5 ends the analysis

with some concluding remarks. To improve the readability of the thesis, comprehensive numerical tables and plots are in Appendix A. Appendices B, C, and D contain a sample listing of the code developed for this research effort.

Chapter Two

Problem Formulation

Our goal is to measure the performance of the control law. This is accomplished by choosing functions of the gains, **f(g)**, which measure the performance of a discretized model in the space of closed loop eigenvalues and eigenvectors. Minimization or constraint of **f(g)** will establish the solution gain vector. The great flexibility of this choice is the primary source of user flexibility (and responsibility!) with this approach. **f** can be a function of the location of the eigenvalues, sensitivities of the eigenvalues to the gains, sensitivities of the eigenvalues to poorly modeled system parameters, sensitivities of the eigenvalues to actuator and sensor placement, singular values, etc. Any robustness criterion which can be written as a differentiable function of system parameters can be included, in principle. The most fundamental constraint, of course, is stability; thus the problem must be formulated so that the solution results in the appropriate closed loop eigenvalues residing in the appropriate sub-domains of the left half of the complex plane.

The performance measures require analysis of the eigenvalue structure of the closed loop system. This chapter provides this analysis. Section 2.1 sets up the problem in output feedback form. Section 2.2 derives the sensitivities which are needed in the continuous design approaches, and also discusses the structure of the set of eigenvalues from a more theoretical standpoint. Finally, Section 2.3 provides a sample problem by deriving a finite element model of a bending bar.

2.1 *State Equations*

Consider the discrete model of a vibrating structure

$$\mathbf{M_S}\,\mathbf{x}'' + \mathbf{C_S}\,\mathbf{x}' + \mathbf{K_S}\,\mathbf{x} = \mathbf{B_S}\,\mathbf{u} \qquad (2\text{-}1)$$

where

$' \equiv d(\cdot)/dt$

and

$\mathbf{x}$	n x 1 configuration vector
$\mathbf{M_s}$	n x n mass matrix
$\mathbf{C_s}$	n x n damping matrix
$\mathbf{K_s}$	n x n stiffness matrix
$\mathbf{B_s}$	n x m control influence matrix
$\mathbf{u}$	m x 1 control vector.

$\mathbf{M_s}$ is symmetric and positive definite, and $\mathbf{C_s}$ and $\mathbf{K_s}$ are symmetric and positive semidefinite. The control $\mathbf{u}$ is desired in the output feedback form

$$\mathbf{u} = \mathbf{G_1}\,\mathbf{y} + \mathbf{G_2}\,\mathbf{z}' \qquad (2\text{-}2)$$

where

$\mathbf{y}$	m_1 x 1 displacement measurement vector
$\mathbf{z}'$	m_2 x 1 velocity measurement vector
$\mathbf{G_1}$	m x m_1 gain matrix
$\mathbf{G_2}$	m x m_2 gain matrix.

$\mathbf{G_1}$ is the position feedback gain matrix, and $\mathbf{G_2}$ is the velocity feedback gain matrix. Let g denote a k x 1 gain vector. It will prove convenient to write

$$\mathbf{G_1} = \mathbf{G_1}(g) \qquad (2\text{-}3)$$

$$G_2 = G_2(g) \ .$$

Assuming the measurements can be expressed as a linear combination of the state variables and velocities,

$$y = N_S \, x \tag{2-4}$$

$$z' = O_S \, x'$$

and

N_S m_1 x n measurement matrix

O_S m_2 x n measurement matrix.

N_S and O_S are determined by sensor type and placement.

Combining equations (2-1), (2-2), (2-3) and (2-4) results in the closed loop system

$$M_S \, x'' + [\, C_S - B_S \, G_2(g) \, O_S \,] \, x' + [\, K_S - B_S \, G_1(g) \, N_S \,] \, x = 0. \tag{2-5}$$

This system can also be expressed in the state space form

$$M \, \eta' = K \, \eta \tag{2-6}$$

where

$$\eta = \{ \, x' \mid x \, \}^t$$

and

$$M(g)= \begin{bmatrix} & : & \\ M_S & : & [0] \\ \cdots & \cdots & \cdots \\ [0] & : & I \\ & : & \end{bmatrix}$$

$$K(g)= \begin{bmatrix} & : & \\ B_S G_S O_S - C_S & : & B_S G_1 N_S - K_S \\ \cdots & \cdots & \cdots \\ I & : & [0] \\ & : & \end{bmatrix}$$

$\mathbf{M}$ 2n x 2n positive definite matrix

$\mathbf{K}$ 2n x 2n matrix.

The system as written in the form (2-6) will be used in the following developments. The eigenvectors and eigenvalues of equations (2-5) and (2-6) can be computed routinely even for moderately high order systems. Equations are derived below to analytically calculate the first and second partial derivatives of the eigenvalues with respect to parameters in the system matrices **M** and **K**. Finding the sensitivities of any function of the eigenvalues is then a straightforward process.

2.2 *Eigenvalue Sensitivities*

The behavior of the deformable structure is modeled by the eigenvalues and eigenvectors of equation (2-5). These closed loop eigenvalues and vectors are in turn determined by the physical structure and by the choice of gains. This section discusses some of the mathematical properties of the eigenvalue problem. Sufficient conditions are found for the existence of the derivatives of the eigenvalues and vectors with respect to the gains, and formulas are developed to calculate these derivatives. First derivative results are

found in Reference [20]. The second derivative results are developed here.

Consider the discrete model written in the state space form (2-6). For solutions of the form

$$\eta(t) = \mathbf{r_i}\, e^{\lambda_i t}, \tag{2-7}$$

we obtain the eigenvalue problem

$$[\,\mathbf{K} - \lambda_i \mathbf{M}\,]\,\mathbf{r_i} = \mathbf{0}. \tag{2-8}$$

λ_i and $\mathbf{r_i}$ are the eigenvalues and eigenvectors of the closed loop system. The eigenvalues will be real or occur in complex conjugate pairs. The corresponding eigenvectors will be real or occur in complex conjugate pairs. The eigenvectors from (2-8) are called right eigenvectors. The left eigenvectors, determined by

$$\mathbf{l_i}^t\,[\,\mathbf{K} - \lambda_i \mathbf{M}\,]' = \mathbf{0}^t\,, \tag{2-9}$$

or

$$[\,\mathbf{K} - \lambda_i \mathbf{M}\,]^t\,\mathbf{l_i} = \mathbf{0}\,,$$

are also useful. The left and right eigenvalues are equal since the transpose of a singular matrix is singular. Derivatives of the eigenvalues are determined by differentiating (2-8) by a gain g_j. Assuming the derivatives of **K** and **M** exist, we have

$$[\,\partial\mathbf{K}/\partial g_j - \partial\lambda_i/\partial g_j\,\mathbf{M} - \lambda_i\,\partial\mathbf{M}/\partial g_j\,]\,\mathbf{r_i} + [\,\mathbf{K} - \lambda_i\,\mathbf{M}\,]\,\partial\mathbf{r_i}/\partial g_j = \mathbf{0}. \tag{2-10}$$

Premultiplying by $\mathbf{l_i}^t$ and applying (2-9) results in

$$\frac{\partial \lambda_i}{\partial g_j} = \frac{\mathbf{l}_i^t \left[\partial \mathbf{K}/\partial g_j - \lambda_i \, \partial \mathbf{M}/\partial g_j \right] \mathbf{r}_i}{\mathbf{l}_i^t \mathbf{M} \mathbf{r}_i} \qquad (2\text{–}11)$$

provided

$$\mathbf{l}_i^t \mathbf{M} \mathbf{r}_i \neq 0.$$

The condition in (2-11) is met if all of the eigenvalues are distinct. We can see this by premultiplying (2-8) by $\mathbf{l}_k^t$.

$$\mathbf{l}_k^t \left[\mathbf{K} - \lambda_i \mathbf{M} \right] \mathbf{r}_i = 0 \qquad (2\text{-}12)$$

Since $\mathbf{l}_k^t$ is a left eigenvector, we can substitute

$$\mathbf{l}_k^t \mathbf{K} = \lambda_k \, \mathbf{l}_k^t \mathbf{M} \qquad (2\text{-}13)$$

Then, from (2-12),

$$(\lambda_k - \lambda_i) \, \mathbf{l}_k^t \mathbf{M} \mathbf{r}_i = 0. \qquad (2\text{-}14)$$

If the eigenvalues are distinct, then

$$\mathbf{l}_k^t \mathbf{M} \mathbf{r}_i = 0 \quad \text{for} \quad k \neq i. \qquad (2\text{-}15)$$

For a system with distinct eigenvalues, the vectors $\mathbf{l}_k$ span the vector space. Since $\mathbf{M}$ has full rank and the matrix $[\mathbf{l}_1 \mid \mathbf{l}_2 \mid \mathbf{l}_3 \mid \ldots \mid \mathbf{l}_{2n}]$ has full rank, the vectors $\mathbf{M}^t \mathbf{l}_k$ span the vector space. Therefore, $\mathbf{l}_k^t \mathbf{M} \mathbf{r}_k$ must be nonzero since (2-15) holds and $\mathbf{r}_k$ is nonzero.

In the problems considered here, **M** does not depend on the gains and (2-11) reduces to

$$\frac{\partial \lambda i}{\partial g_j} = \frac{\mathbf{l}_i^t \left[\partial \mathbf{K}/\partial g_j \right] \mathbf{r}_i}{\mathbf{l}_i^t \mathbf{M} \mathbf{r}_i}. \qquad (2\text{–}16)$$

In summary, when **M** is not dependent on th gains, **K** is differentiable with respect to the gains, and all the eigenvalues are distinct, then (2-16) gives the sensitivities of the eigenvalues to the gains.

Many meaningful performance measures include a term with sensitivities of the eigenvalues. This allows a robust design process. As discussed in Chapters Three and Four, the two design methods in this thesis would then require the second derivatives $\partial^2 \lambda_k / \partial g_i \partial g_j$. The calculation of these derivatives in turn require the sensitivities of the eigenvectors to the gains. The eigenvector sensitivities are determined by using an expansion in terms of eigenvectors. Consider

$$\partial \mathbf{r}_i / \partial g_j = \sum_{k=1}^{2n} a_k \mathbf{r}_k \qquad (2\text{-}17)$$

where $\{ a_k \}$ is a set of complex scalars and $\mathbf{r}_k$ are right eigenvectors. If the eigenvalues are distinct, then the set of eigenvectors is complete and (2-17) is meaningful. Now substitute this expression into equation (2-10). Premultiplication by $\mathbf{l}_p^t$ and manipulation of terms results in

$$\mathbf{l}_p^t \left[\partial \mathbf{K}/\partial g_j - \partial \lambda_i / g_j \mathbf{M} - \lambda_i \partial \mathbf{M}/\partial g_j \right] \mathbf{r}_i + \qquad (2\text{-}18)$$

$$\sum_{k=1}^{2n} a_k \left[\mathbf{l_p}^t \mathbf{K}\, \mathbf{r_k} - \lambda_i \mathbf{l_p}^t \mathbf{M}\, \mathbf{r_k} \right] = 0.$$

Now recall that

$$[\mathbf{K} - \lambda_k \mathbf{M}]\, \mathbf{r_k} = 0. \tag{2-19}$$

Premultiplying by $\mathbf{l_p}^t$,

$$\mathbf{l_p}^t \mathbf{K}\, \mathbf{r_k} = \lambda_k\, \mathbf{l_p}^t \mathbf{M}\, \mathbf{r_k}. \tag{2-20}$$

If the eigenvalues are distinct, then equations (2-14) and (2-20) indicate

$$\mathbf{l_p}^t \mathbf{M}\, \mathbf{r_k} = 0 \quad \text{if} \quad p \neq k$$

and (2-21)

$$\mathbf{l_p}^t \mathbf{K}\, \mathbf{r_k} = 0 \quad \text{if} \quad p \neq k\ .$$

Substituting these results into (2-18) yields

$$a_p = \frac{-\mathbf{l_p}^t \left[\partial \mathbf{K}/\partial g_j - \partial \lambda_i/\partial g_j\, \mathbf{M} - \lambda_i\, \partial \mathbf{M}/\partial g_j \right] \mathbf{r_i}}{\mathbf{l_p}^t \left[\mathbf{K} - \lambda_i \mathbf{M} \right] \mathbf{r_p}} \quad \text{if } p \neq i. \tag{2–22}$$

The assumption of distinct eigenvalues implies that the denominator is nonzero. The coefficient a_p corresponding to p=i is still undetermined. The unknown is resolved by scaling the eigenvectors so that

$$\mathbf{r_i}^t \mathbf{M}\, \mathbf{r_i} = 1 \qquad i=1,2,3, \ldots ,2n\ . \tag{2-23}$$

From this constraint,

$$\partial \mathbf{r}_i^t/\partial g_j \, \mathbf{M} \, \mathbf{r}_i + \mathbf{r}_i^t \, \partial \mathbf{M}/\partial g_j \, \mathbf{r}_i + \mathbf{r}_i^t \, \mathbf{M} \, \partial \mathbf{r}_i/\partial g_j = 0 . \tag{2-24}$$

Substituting into (2-17) and recalling that **M** is symmetric results in

$$a_i = - \sum_{\substack{k=1 \\ k \neq i}}^{2n} a_k \, \mathbf{r}_k^t \, \mathbf{M} \, \mathbf{r}_i - 1/2 \, \mathbf{r}_i^t \, \partial \mathbf{M}/\partial g_j \, \mathbf{r}_i . \tag{2-25}$$

Equations (2-22) and (2-25) provide the coefficients to calculate the sensitivities of the right eigenvectors to the gains. The sensitivities of the left eigenvectors are found in a similar way. The expansion

$$\partial \mathbf{l}_i/\partial g_j = \sum_{k=1}^{2n} b_k \, \mathbf{l}_k \tag{2-26}$$

and the constraints

$$\mathbf{l}_i^t \, \mathbf{M}^t \, \mathbf{l}_i = 1 \cdot \quad i=1,2,3, \ldots ,2n \tag{2-27}$$

result in

$$b_p = \frac{- \mathbf{l}_i^t \left[\partial \mathbf{K}/\partial g_j - \partial \lambda_i/\partial g_j \, \mathbf{M} - \lambda_i \, \partial \mathbf{M}/\partial g_j \right] \mathbf{r}_p}{\mathbf{l}_p^t \left[\mathbf{K} - \lambda_i \, \mathbf{M} \right] \mathbf{r}_p} \quad \text{if } p \neq i \tag{2–28}$$

$$b_i = - \sum_{\substack{k=1 \\ k \neq i}}^{2n} b_k \, \mathbf{l}_k^t \, \mathbf{M} \, \mathbf{l}_i - 1/2 \, \mathbf{l}_i^t \, \partial \mathbf{M}/\partial g_j \, \mathbf{l}_i . \tag{2-29}$$

Again, in the problems considered here, **M** does not depend on the gains. Therefore, (2-22), (2-25), (2-28) and (2-29) reduce to

$$a_p = \frac{-\mathbf{l_p}^t [\, \partial \mathbf{K}/\partial g_j - \partial \lambda_i/\partial g_j \, \mathbf{M} \,] \, \mathbf{r_i}}{\mathbf{l_p}^t [\, \mathbf{K} - \lambda_i \, \mathbf{M} \,] \, \mathbf{r_p}} \qquad \text{if } p \neq i \tag{2-30}$$

$$a_i = - \sum_{\substack{k=1 \\ k \neq i}}^{2n} a_k \, \mathbf{r_k}^t \mathbf{M} \, \mathbf{r_i} \tag{2-31}$$

$$b_p = \frac{-\mathbf{l_i}^t [\, \partial \mathbf{K}/\partial g_j - \partial \lambda_i/\partial g_j \, \mathbf{M} \,] \, \mathbf{r_p}}{\mathbf{l_p}^t [\, \mathbf{K} - \lambda_i \, \mathbf{M} \,] \, \mathbf{r_p}} \qquad \text{if } p \neq i \tag{2-32}$$

$$b_i = - \sum_{\substack{k=1 \\ k \neq i}}^{2n} b_k \, \mathbf{l_k}^t \mathbf{M} \, \mathbf{l_i} \tag{2-33}$$

when

$$\mathbf{r_i}^t \mathbf{M} \, \mathbf{r_i} = 1 \qquad i=1,2,3, \ldots ,2n \tag{2-34}$$

$$\mathbf{l_i}^t \mathbf{M}^t \mathbf{l_i} = 1 \qquad i=1,2,3, \ldots ,2n \tag{2-35}$$

$$\partial \mathbf{r_i}/\partial g_j = \sum_{k=1}^{2n} a_k \, \mathbf{r_k} \tag{2-36}$$

$$\partial \mathbf{l}_i/\partial g_j = \sum_{k=1}^{2n} b_k \mathbf{l}_k \ . \qquad (2\text{-}37)$$

These equations hold when **M** does not depend on the gains and is symmetric, **K** is differentiable, and all of the eigenvalues are distinct.

The second derivatives $\partial^2\lambda/\partial g_k \partial g_k$ can now be determined. Taking the derivative of equation (2-11) yields

$$\begin{aligned}
\frac{\partial^2 \lambda_i}{\partial g_j \partial g_k} = \frac{1}{(\mathbf{l}_i^t \mathbf{M} \mathbf{r}_i)^2} & \Big[-\mathbf{l}_i^t [\, \partial \mathbf{K}/\partial g_j - \lambda_i \partial \mathbf{M}/\partial g_j \,] \mathbf{r}_i \\
& [\, \partial \mathbf{l}_i^t/\partial g_k \mathbf{M} \mathbf{r}_i + \mathbf{l}_i^t \partial \mathbf{M}/\partial g_k \mathbf{r}_i + \mathbf{l}_i^t \mathbf{M} \partial \mathbf{r}_i/\partial g_k] \Big] \\
& + \frac{1}{\mathbf{l}_i^t \mathbf{M} \mathbf{r}_i} \Big[\partial \mathbf{l}_i^t/\partial g_k [\, \partial \mathbf{K}/\partial g_j - \lambda_i \partial \mathbf{M}/\partial g_j \,] \mathbf{r}_i \qquad (2\text{-}38) \\
& + \mathbf{l}_i^t [\, \partial^2 \mathbf{K}/\partial g_j \partial g_k - \partial \lambda_i/\partial g_k \partial \mathbf{M}/\partial g_j - \lambda_i \partial^2 \mathbf{M}/\partial g_j \partial g_k \,] \mathbf{r}_i \\
& + \mathbf{l}_i^t [\, \partial \mathbf{K}/\partial g_j - \lambda_i \partial \mathbf{M}/\partial g_j] \partial \mathbf{r}_i/\partial g_k \Big]
\end{aligned}$$

When **M** does not depend on the gains and is symmetric, **K** is differentiable, and the eigenvalues are distinct, then

$$\frac{\partial^2 \lambda_i}{\partial g_j \partial g_k} = \frac{-l_i^t [\, \partial K/\partial g_j \,] r_i \; [\, \partial l_i^t/\partial g_k \, M r_i \; + l_i^t M \, \partial r_i/\partial g_k]}{(\, l_i^t M r_i \,)^2}$$

$$+ \frac{1}{l_i^t M r_i} \left[\partial l_i^t/\partial g_k \, [\, \partial K/\partial g_j \,] r_i \; + \; l_i^t \, [\, \partial^2 K/\partial g_j \partial g_k \,] r_i \right. \qquad (2\text{-}39)$$

$$\left. + \, l_i^t \, [\, \partial K/\partial g_j \,] \, \partial r_i/\partial g_k \right]$$

This completes the calculations of the sensitivities.

As noted above, all of the sensitivity equations require a full set of unique eigenvalues. Reference [20] tells us that the eigenvalues are continuous functions of the parameters. However, the derivatives are not continuous or bounded when repeated eigenvalues occur. The methods of this thesis require bounded derivatives of the eigenvalues to evaluate derivatives of the performance measure. The repeated eigenvalue is therefore a barrier which cannot be crossed. As a result, the basic structure of the set of eigenvalues cannot be changed; each real eigenvalue must remain real, and each complex eigenvalue must remain complex. This artificial constraint on the eigenvalue structure is one of the greatest drawbacks of the algorithms in the following chapters. This limitation is discussed again as it appears in the examples that follow.

2.3 *Finite Element Model of a Bending Beam*

This thesis discusses several design stategies for the control of flexible structures. Demonstration of the strategies requires the development of a discrete model. In this section, a finite element model is developed to study a cantilevered beam in bending. Figure 2.1 shows the beam and beam coordinate system. The presentation here is brief

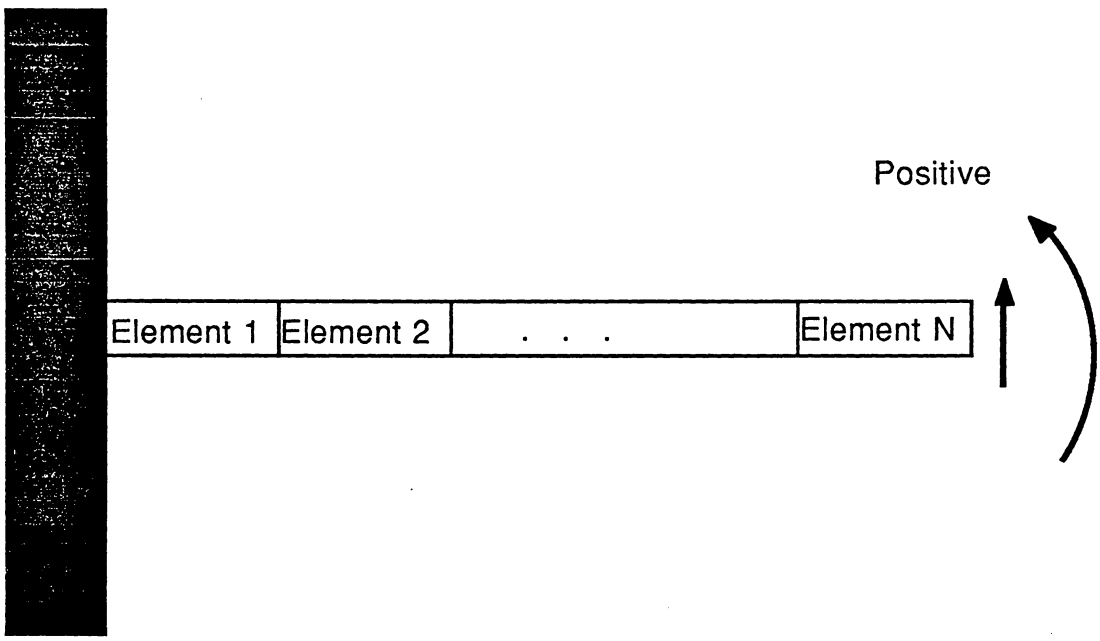


Figure 2.1 A Flexible Beam

and closely follows Leonard Meirovitch's Elements of Vibration Analysis [Ref.21]. Interested readers should see the reference for a detailed explanation. The beam is analyzed by first describing the dynamics of a small beam element. The beams are then assembled to construct a (moderately) high dimension model. This model is converted to a first order system to conform with the developments in earlier sections. Finally, open loop natural frequencies and mode shapes are determined and compared to results from the continuous beam equations. This comparison indicates how many eigenvalues are accurately modeled. Only the accurate eigenvalues should be included in a performance measure.

As the first step in the finite element procedure, the dynamics of a single beam element are described. Consider the element in Figure 2.2. The element has mass per unit length m, bending stiffness EI and length L. w(x,t) represents the deflection of the beam. $\mathbf{u}(t)$ is the displacement vector of the beam ends. u_1 and u_3 are linear displacements, while u_2 and u_4 are angular displacements. $\mathbf{f}(t)$ is the joint force vector. f_1 and f_3 are forces, and f_2 and f_4 are moments. The distributed nonconservative force f(x,t) is not shown in the figure. Using separation of variables, w(x,t) is expressed as

$$w(x,t) = \sum_{i=1}^{4} \phi_i(x)\, u_i(t) . \tag{2-40}$$

Since w(x,t) must match the joint displacements $\mathbf{u}(t)$,

$$\mathbf{u}(t) = \begin{bmatrix} w(0,t) \\ \partial w(x,t)/\partial x \,|_{x=0} \\ w(L,t) \\ \partial w(x,t)/\partial x \,|_{x=L} \end{bmatrix} . \tag{2-41}$$

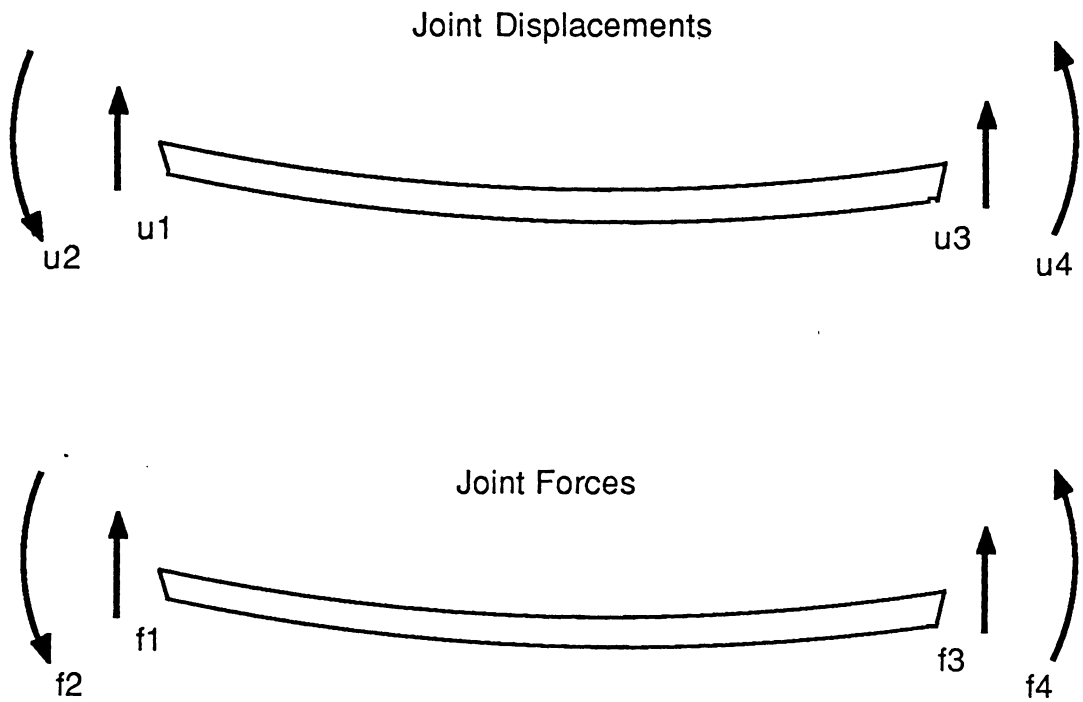


Figure 2.2 A Beam Element

This leads to the boundary conditions

$$\phi(0) = [\ 1, 0, 0, 0\ ]^t \tag{2-42}$$

$$\phi(L) = [\ 0, 0, 1, 0\ ]^t$$

$$\partial\phi/\partial x\,|_{x=0} = [\ 0, 1, 0, 0\ ]^t$$

$$\partial\phi/\partial x\,|_{x=L} = [\ 0, 0, 0, 1\ ]^t\ .$$

Now the equation of static bending for a uniform beam is

$$d^4\, w(x)/dx^4 = 0, \quad 0 < x < L\ . \tag{2-43}$$

Solutions of the static equations are cubics. Using the separation of variables in (2-40) and the boundary conditions in (2-42) results in

$$\phi(x) = \begin{bmatrix} 1 - 3(x/L)^2 + 2(x/L)^3 \\ x - 2x^2/L + x^3/L^2 \\ 3(x/L)^2 - 2(x/L)^3 \\ -x^2/L + x^3/L^2 \end{bmatrix} \tag{2-44}$$

Lagrangian dynamics is used to develop the element equation of motion. First the kinetic energy is

$$T(t) = 1/2 \int_0^L m\,(\partial w(x,t)/\partial t)^2\, dx\ . \tag{2-45}$$

The potential energy is then

$$V(t) = 1/2 \int_0^L EI \left(\partial^2 w(x,t)/\partial x^2 \right)^2 dx \ . \tag{2-46}$$

The virtual work is

$$\delta W = \sum_{i=1}^{4} f_i(t)\, \delta u_i(t) \ , \tag{2-47}$$

where

$$\mathbf{f_i}(t) = \int_0^L f(x,t)\, \phi_i(x)\, dx + \mathbf{f^*_i}(t) \ . \tag{2-48}$$

f(x,t) is the distributed nonconservative forces and $\mathbf{f}^*(t)$ is the force vector exerted by adjacent elements. The component of $\mathbf{f}(t)$ from the distributed nonconservative forces is referred to as $\mathbf{f_{applied}}(t)$. Then

$$\mathbf{f}(t) = \mathbf{f_{applied}}(t) + \mathbf{f}^*(t) \ . \tag{2-49}$$

Substituting these expressions into Lagrange's equation of motion,

$$\frac{d}{dt}\left[\frac{\partial T}{\partial q_i'} \right] - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = Q_i \ , \quad i = 1,2,3, \ldots ,n, \tag{2-50}$$

results in

$$\mathbf{m_{elem}}\, \mathbf{u}''(t) + \mathbf{k_{elem}}\, \mathbf{u}(t) = \mathbf{f}(t). \tag{2-51}$$

The element mass and stiffness matrices are

$$\mathbf{m_{elem}} = m\,L/420 \begin{bmatrix} 156 & 22L & 54 & -13L \\ 22L & 4L^2 & 13L & -3L^2 \\ 54 & 13L & 156 & -22L \\ -13L & -3L^2 & -22L & 4L^2 \end{bmatrix} \quad (2\text{–}52)$$

$$\mathbf{k_{elem}} = EI/L^3 \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \quad (2\text{–}53)$$

Equation (2-51) provides the equations of motion of the individual element. Now the elements must be assembled. Figure 2.3 shows the joint and element numbering system. Each element has a set of equations of motion in the form of (2-51). Since the joints are massless points, adjacent elements exert equal and opposite joint forces. Equating the joint forces results in the system equation of motion

$$\mathbf{M_a}\,\mathbf{x}'' + \mathbf{K_a}\,\mathbf{x} = \mathbf{f_a} \quad (2\text{-}54)$$

where $\mathbf{f_a}$ is the system force vector. The description of the elements in $\mathbf{M_a}$, $\mathbf{K_a}$, and $\mathbf{f_a}$ is slightly complicated.

$$\mathbf{M_a} = \sum_{i=1}^{n_{elem}} \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & 2(i-1)\text{ rows of zeros} & \cdot \\ \cdots & \cdots & \cdots \\ \cdot & \cdot & \cdot \\ 2(i-1)\text{ columns of zeros} & \mathbf{m_{elem\ \#\ i}} & 2n_{elem} - 2i - 6\text{ columns of zeros} \\ \cdot & \cdot & \cdot \\ \cdots & \cdots & \cdots \\ \cdot & 2n_{elem} - 2i - 6\text{ rows of zeros} & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \quad (2\text{–}55)$$

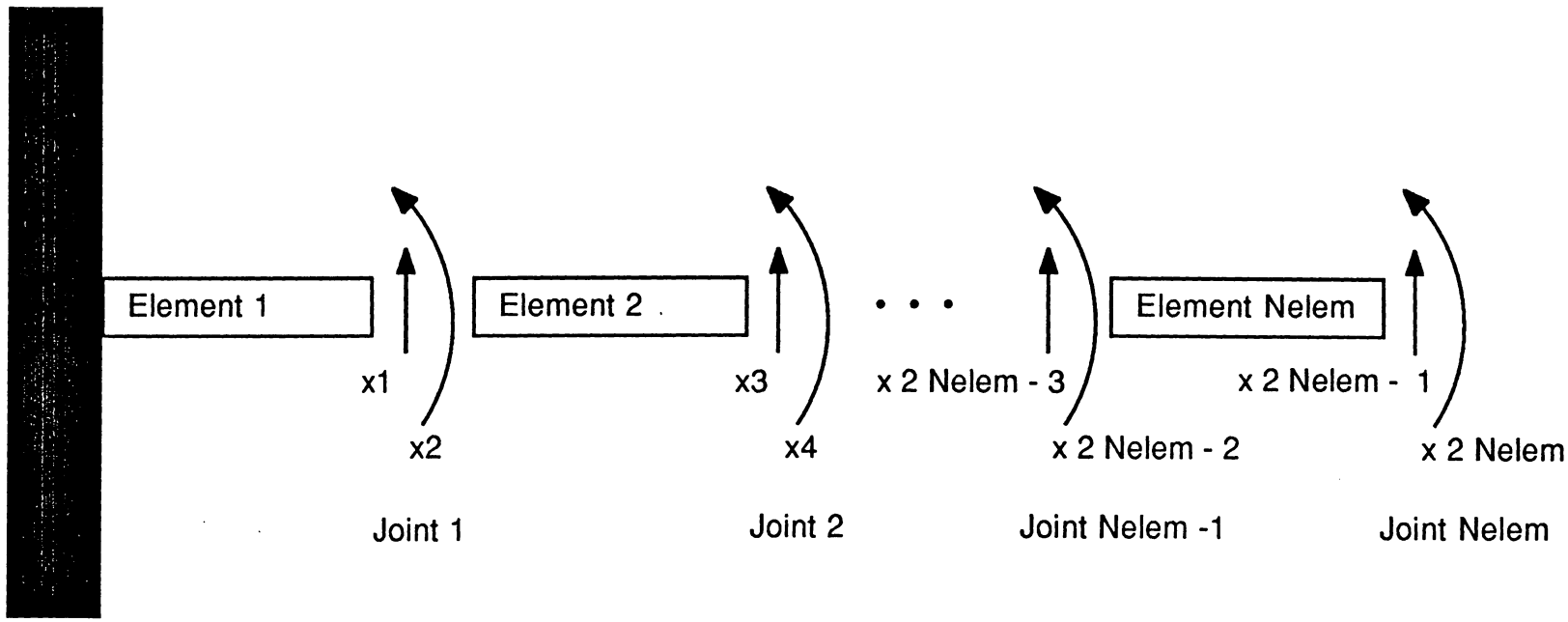


Figure 2.3 Element Numbering Scheme

Similiarly,

$$\mathbf{K_a} = \sum_{i=1}^{n_{elem}} \begin{bmatrix} \\ \text{2(i-1) columns of zeros} \\ \\ \hline \\ \mathbf{k_{elem\ \#\ i}} \\ \\ \hline \\ \text{}2n_{elem} - 2i - 6 \text{ columns of zeros} \\ \\ \end{bmatrix} \quad (2\text{–}56)$$

The force vector $\mathbf{f_a}$ is

$$\mathbf{f_a} = \sum_{i=1}^{n_{elem}} \begin{bmatrix} \text{2(i-1) zeros} \\ \hline \\ \mathbf{f_{applied\ \#i}} \\ \\ \hline \\ 2n_{elem} - 2i - 6 \text{ zeros} \end{bmatrix} . \quad (2\text{–}57)$$

Applying the constraints of zero displacement and angle at the base of the beam results in the model matrices $\mathbf{K_s}$ and $\mathbf{M_s}$ in equation (2-1).

This thesis requires a particular implementation of the flexible beam model. Here, a twelve element model is chosen, resulting in a model of dimension forty eight. Values for EI, M and L are chosen so that range of the open loop eigenvalues are appropraite for a

highly flexible structure. Table 1.1 lists the values. Since a uniform beam is modeled, the open loop eigenvalues can be calculated theoretically. [Ref. 21] Numerical results are included in Appendix A. Table A.1 compares the first few analytical and approximate eigenvalues. Finally, Figures A.1 through A.12 show the open loop mode shapes. As seen from Table A.1, only the lower eigenvalues and modes are accurately modeled. Only these lower eigenvalues will be used in performance measures based on the discrete model.

Table 2.1

Beam Physical Constants

mass per unit length	1 kg/m
length	1 m
bending stiffness	0.1 kg m^3 / s^2

Chapter Three

The Optimization Method

3.1 *The Optimization Algorithm*

The optimization method designs a control law for (2-1) by minimizing a measure of performance f(**g**). Minimizing f can present a significant challenge. The function f is often "busy" and contains very sensitive eigenvalues and partial derivatives. Repeated attempts to use a discrete optimization procedure often result in numerical overflows and other difficulties. A simple continuous gradient algorithm combined with a modern integration routine provides reliable results. The routine's automatic step size control and order estimation dramatically improve computational efficiency.

The algorithm solves

$$d\mathbf{g}/d\tau = - \partial f/\partial \mathbf{g}^t \ , \quad 0 \le \tau \le \infty \tag{3-1}$$

for $\mathbf{g}(\tau)$, with the initial condition

$$\mathbf{g}(0) = \mathbf{g}_0 \ .$$

τ is simply a scalar parameter of the trajectory, related to arclength along the space curve. The initial condition can be chosen in a number of ways. $\mathbf{g}_0$ can be chosen, for lack of a better choice, as zero. $\mathbf{g}_0$ may also be determined by using an existing control law. The continuous gradient method is then used to improve an existing design by implementing meaningful performance measures. Applications of optimization methods to the fine tuning of complex systems is extensively used, and discussed in references [8,9]. In any case, a careful choice is necessary. As discussed in Section 2.2, the eigenvalue structure

contains singularities which prevent a continuous method from arbitrarily placing the eigenvalues.

The gradient method converges slowly near the minimum. A second order or conjugate gradient method could be used to accelerate convergence in this region. However, this may be unnecessary since we are more interested in finding a good nominal design than in exactly identifying the minimum of f. Fortunately, for most practical examples, the minima are fairly broad. This is usually indicative of a robust design.

The following sections contain four examples using the gradient method. These examples illustrate the flexibility of the design procedure, but they also illustrate the problems which may occur. Appendices B and D contain sample listings of the computer code used in these sections.

3.2 *Using a Small Number of Sensors and Actuators*

As discussed in the Introduction, several design procedures are available when numerous sensors and actuators may be used. These design procedures give great flexibility in system design. However, in most practical applications, only a few sensors and actuators can be implemented. The substantial sensor and actuator requirements are the greatest weakness of these elegant analytical methods. The gradient design method can be applied without restrictions in sensor and actuator placement. Unfortunately, this flexibility comes at the cost of analytic results; no general conclusions can be drawn a priori about the method's success. This section provides an example of design using a small number of sensors and actuators.

Consider the finite element model of a flexible beam developed in Chapter 2. Figure 2.3 shows the joint displacement and angle numbering scheme. In this section, we will consider a two sensor and two actuator configuration. Figure 3.1 shows the

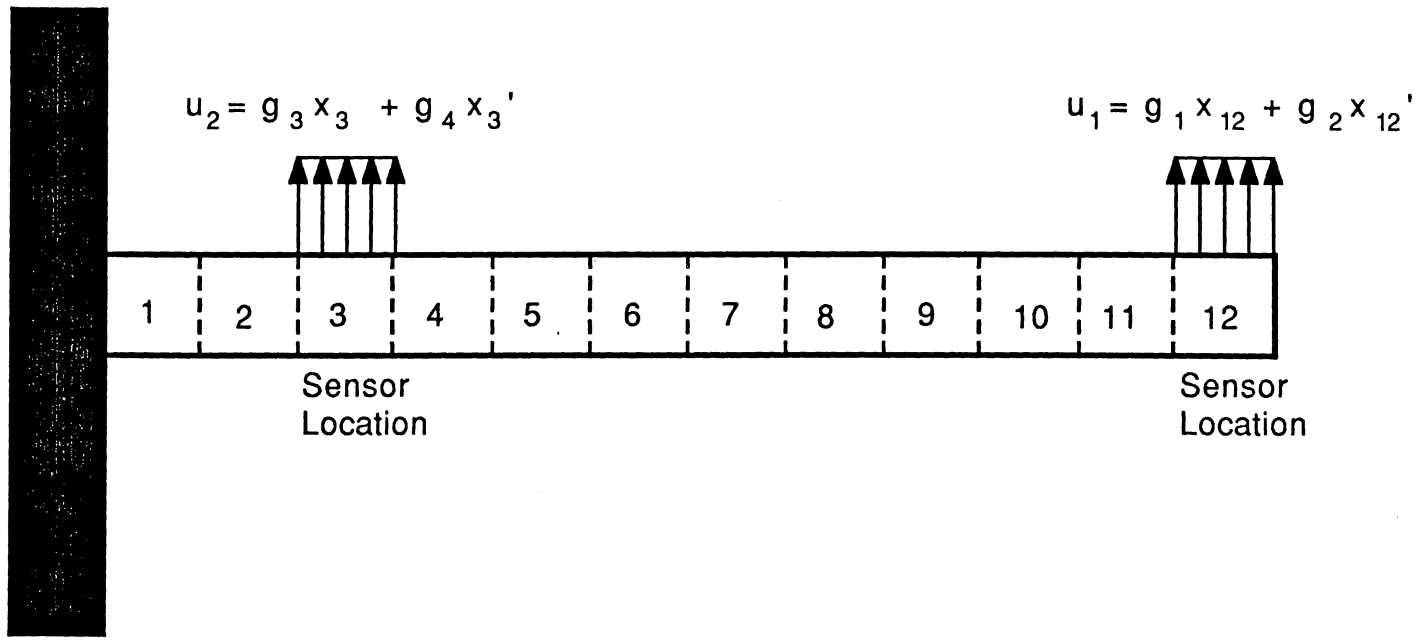


Figure 3.1 The Sensor and Actuator Configuration

configuration. The first actuator provides a uniformly distributed force on the last element through feedback on the end position and rate. The second actuator provides a uniformly distributed force on the third element through feedback on the third joint position and rate.

$$u_1 = g_1 x_{12} + g_2 x_{12}' \qquad (3\text{-}2)$$

$$u_2 = g_3 x_3 + g_4 x_3' ,$$

with

u_i = distributed force on element 3 or 12

g_i = gain i

x_i = position of joint i

x_i' = derivative of x_i.

Notice that the actuators use only local information; the last element is forced on the basis of last element information, and the third element is forced on the basis of third element information. This colocation also simplifies practical implementation, since no communication between widely separated sensors and actuators is necessary.

The gradient method requires choice of a performance measure. In this example, the first eight eigenvalues, arranged in order of increasing imaginary part, of equation (2-6) are included. The goal is to drive the real part of these eigenvalues to the left of -0.5 rad./sec. A penalty function approach is used to place the eigenvalues in an acceptable portion of the complex plane. This is accomplished through choice of the performance measure

$$f(g) = \sum_{i=1}^{8} f_i(g) \qquad (3\text{-}3)$$

where

$$f_i(g) = \begin{cases} [\ \text{real}(\lambda_i) + 0.5\ ]^3 & \text{if real}(\lambda_i) > -0.5 \text{ rad./sec.} \\ 0 & \text{otherwise.} \end{cases}$$

Equations (2-16) from Chapter 2 provided the needed derivatives in equation (3-1). Initial conditions are provided by using an initial design of

$$g_0 = \{\ 0, 0, 0, 0\ \}^t .$$

The equations are integrated numerically using a Hammings modified predictor-corrector method and automatic stepsize control [Ref. 22].

The gradient algorithm drives the cost to (very near) zero, thereby driving the appropriate eigenvalues into the desired subdomain of the complex plane. The numerical results and graphs are included in Appendix A. Table A.2 gives the initial and final gain values. Table A.3 gives the initial and final eigenvalues. The cost is also included. Note that the cost is driven virtually to zero. This is best illustrated through plots of various trajectories. Figure A.13 contain plots of the gains. Notice that the plots flatten out, as expected. Figure A.14 shows the performance measure, which goes to zero. The real parts of the eigenvalues, shown in Figure A.15, are driven to the left of -0.5 rad./sec., as desired. The imaginary components of the eigenvalues, shown in Figure A.16 change very little. This is not surprising, since the cost does not explicitly contain the imaginary components. Note also that gains one and three, the position feedback gains, remain near zero. This is again not surprising when the imaginary parts are not included in the cost.

In the example in this section, the performance measure is actually driven to zero. When more complicated measures are used, this may not be possible. The performance measure may have a minimum which is not in the zero region of the penalty functions. In this case, some judgment is necessary when choosing relative weights of various portions

of the performance measure. This is the case in the examples which follow.

3.3 *Performance Measures in Noise*

In many applications, feedback control is used to stabilize a system in the presence of noise. The methods considered here are useful for the design of feedback control laws which are "tuned" to the statistical characteristics of the noise. If the second order statistics of this noise are known a priori, then the interaction between the noise and the flexible structure can be explicitly included in the design process.

Consider the linear system in equation (2-6), with the gain matrices $\mathbf{G_1}$ and $\mathbf{G_2}$ assumed known. We need to characterize the performance of this feedback system in noise. Our interest here will be limited to the single input - single output case. This will be adequate to illustrate the design method. First formalize the problem with input n(t) and output o(t) as

$$\eta' = \mathbf{A}\,\eta + \mathbf{b}\,n(t) \qquad (3\text{-}4)$$

$$o(t) = \mathbf{c}^t \eta$$

with

$\mathbf{A} = \mathbf{M}^{-1}\mathbf{K}$

$\eta = \{\, \mathbf{x}' \mid \mathbf{x} \,\}$,

$\mathbf{b}$ = 2n x 1 noise influence vector, and

$\mathbf{c}$ = 2n x 1 measurement vector.

The input noise n(t) is assumed to be a wide sense stationary process with zero mean and covariance $R_n(t)$. We then know, from results on the linear filtering of wide sense stationary noise, that the output o(t) is zero mean with covariance function

$$R_o(t) = \int_{-\infty}^{\infty} R_n(t - (\alpha - \beta))\, h(\alpha)\, h(\beta)\, d\alpha\, d\beta \qquad (3\text{-}5)$$

with

$h(t)$ = the impulse response of the linear system.

Then, since **M** is positive definite,

$$h(t) = \mathbf{c}^t \exp(\mathbf{A}\, t)\, \mathbf{b} \qquad \text{when } t \geq 0, \text{ and} \qquad (3\text{-}6)$$

$$= 0 \qquad \text{when } t < 0.$$

Another form of equation (3-6) will prove more convenient. Define the matrix **V** in terms of the right eigenvectors:

$$\mathbf{V} = [\ \mathbf{r_1} \mid \mathbf{r_2} \mid \ldots \mid \mathbf{r_{2n}}\ ].$$

Assuming that **A** has distinct eigenvalues,

$$\exp(\mathbf{A}\, t) = \exp[\ \mathbf{V}\ \text{diagonal}\ (\ \lambda_1, \lambda_2, \ldots, \lambda_{2n}\ )\ \ \mathbf{V}^{-1}\ ]$$

$$= \mathbf{V}\ \text{diagonal}(\ e^{\lambda_1 t}, e^{\lambda_2 t}, \ldots, e^{\lambda_{2n} t}\ )\ \ \mathbf{V}^{-1}\ .$$

Thus

$$h(t) = \mathbf{c}^t\, \mathbf{V}\ \text{diagonal}(\ e^{\lambda_1 t}, e^{\lambda_2 t}, \ldots, e^{\lambda_{2n} t}\ )\ \ \mathbf{V}^{-1}\ \mathbf{b}\ . \qquad (3\text{-}7)$$

The convolution in (3-5) is most easily performed using Fourier transforms. Letting $H(f)$ be the transform of the impulse response $h(t)$ and applying (3-7),

$$H(f) = \int_0^{\infty} \mathbf{c}^t \exp(\mathbf{A}\, t)\, \mathbf{b}\, e^{-2\pi f t j}\, dt\ . \qquad (3\text{-}8)$$

From equation (3-7) and linearity, we then see

$$H(f) = \mathbf{c}^t \mathbf{V} \text{ diagonal}\left(\int_0^\infty e^{\lambda_i t - 2\pi f t j} \, dt \right) \mathbf{V}^{-1} \mathbf{b} \, . \qquad (3\text{-}9)$$

We require that the real part of all of the eigenvalues be strictly negative. This is a strong requirement since, as a result of the discretization, only the smaller eigenvalues accurately model the flexible beam. With this assumption, equation (3-9) immediately yields

$$H(f) = \mathbf{c}^t \mathbf{V} \text{ diagonal}\left(-(\lambda_i - 2\pi f j)^{-1} \right) \mathbf{V}^{-1} \mathbf{b} \, . \qquad (3\text{-}10)$$

The formulas are now developed to apply Fourier transform methods to equation (3-5). Letting $H_n(f)$ and $H_o(f)$ be the transform of $R_n(t)$ and $R_o(t)$, the input and output covariance functions,

$$H_o(f) = H(f) \; H(f)^* \; H_n(f) \qquad (3\text{-}11)$$

where * denotes complex conjugate.

The function $H_o(f)$ can be inverse transformed to yield $R_o(t)$.

A simpler application is made here. Note the variance of the output is

$$E(o(t)^2) = \int_{-\infty}^{\infty} H_o(f) \, df \quad , \qquad (3\text{-}12)$$

where E( ) denotes expectation. The goal is to make this integral small. In the examples of interest, the input n(t) will be bandpass and characterized by a center frequency f_c. In light of (3-11), choosing gains to minimize $H(f_c) \; H(f_c)^*$ is a reasonable goal. In that case, the flexible structure itself will filter out much of the power of n(t).

Minimizing $H(f_c) \; H(f_c)^*$ requires derivatives with respect to the gains. These are

developed below.

$$\partial\,(H(f_c)\,H(f_c)^*)/\partial g_k = \partial\,H(f_c)/\partial g_k\;H(f_c)^* + H(f_c)\,\partial\,H(f_c)^*/\partial g_k$$

$$= \partial\,H(f_c)/\partial g_k\;H(f_c)^* + H(f_c)\,[\,\partial\,H(f_c)/\partial g_k\,]^*. \qquad (3\text{-}13)$$

Applying $\partial/\partial g_k$ to equation (3-10),

$$\partial\,H(f_c)/\partial g_k = \mathbf{c}^t\,\partial\mathbf{V}/\partial g_k\;\text{diagonal}(\,-(\lambda_i - 2\pi f j)^{-1}\,)\,\mathbf{V}^{-1}\,\mathbf{b} \qquad (3\text{-}14)$$

$$+\,\mathbf{c}^t\,\mathbf{V}\;\text{diagonal}(\,(\lambda_i - 2\pi f j)^{-2}\,\partial\lambda_i/\partial g_k\,)\,\mathbf{V}^{-1}\,\mathbf{b}$$

$$+\,\mathbf{c}^t\,\mathbf{V}\;\text{diagonal}(\,-(\lambda_i - 2\pi f j)^{-1}\,)\,\partial\mathbf{V}^{-1}/\partial g_k\;\mathbf{b}.$$

One more equation is required. Note, differentiating

$$\mathbf{V}\,\mathbf{V}^{-1} = \mathbf{I}$$

yields

$$\partial\mathbf{V}^{-1}/\partial g_k = -\,\mathbf{V}^{-1}\,\partial\mathbf{V}/\partial g_k\,\mathbf{V}^{-1}\,. \qquad (3\text{-}15)$$

Finally,

$$\partial\,H(f_c)/\partial g_k = \mathbf{c}^t\,\partial\mathbf{V}/\partial g_k\;\text{diagonal}(\,-(\lambda_i - 2\pi f j)^{-1}\,)\,\mathbf{V}^{-1}\,\mathbf{b} \qquad (3\text{-}16)$$

$$+\,\mathbf{c}^t\,\mathbf{V}\;\text{diagonal}(\,(\lambda_i - 2\pi f j)^{-2}\,\partial\lambda_i/\partial g_k\,)\,\mathbf{V}^{-1}\,\mathbf{b}$$

$$-\,\mathbf{c}^t\,\mathbf{V}\;\text{diagonal}(\,-(\lambda_i - 2\pi f j)^{-1}\,)\,\mathbf{V}^{-1}\,\partial\mathbf{V}/\partial g_k\,\mathbf{V}^{-1}\,\mathbf{b}\,,$$

with

$$\partial\mathbf{V}/g_k = [\;\partial\mathbf{r_1}/\partial g_k \mid \partial\mathbf{r_2}/g_k \mid \ldots \mid \partial\mathbf{r_{2n}}/g_k\;].$$

The expressions $\partial\lambda_i/\partial g_k$ and $\partial \mathbf{r_i}/\partial g_k$ were developed earlier. Equations (3-13) and (3-16) provide the tools needed to include input noise characteristics in the control law design process.

The development above requires that all the eigenvalues of the model have negative real part. This is, of course, a requirement for stability of the linear system. Unfortunately, as discussed in Section 2.3, only the lower frequency eigenvalues accurately model the beam. The higher frequency modes are not physical; they are a result of the discretization process and do not describe the actual motion of the beam. Numerous sensor and actuator configurations were attempted to adequately stabilize these poorly modeled modes. None of the configurations led to an appropriate set of eigenvalues. The performance measure in this section is not appropriate for the model used here, since the method puts an artificial constraint on the poorly modeled modes. However, the performance measure is appropriate for other modeling procedures which lead to discrete models with only accurately modeled modes.

3.4 *A Measure of Robustness: Using Sensitivities*

The methods developed in this thesis allow great flexibility in system design. Measures of robustness can be included in the cost function. In principle, a variety of measures of robustness could be applied: sensitivities of the eigenvalues to poorly modeled system parameters, sensitivity of the eigenvalues to actuator and sensor placement, singular values, etc. Robustness is demonstrated here through the inclusion of the sensitivities of the eigenvalues to the gains. Some of the limitations of the gradient method are demonstrated here; the penalty function approach may not place all the selected eigenvalues in the desired portion of the complex plane. However, the derived control law may still have adequate performance.

Again consider the flexible beam model from Chapter 2. In this example, we will consider four pairs of colocated actuators and sensors. This configuration will require no communication between widely separated sensors and actuators. Figure 2.3 shows the state numbering scheme, and Figure 3.2 shows the controller configuration. Each of the four actuators imparts a uniformly distributed force over an element, using position and rate feedback. More rigorously,

$$u_1 = g_1 x_{12} + g_2 x_{12}' \tag{3-17}$$

$$u_2 = g_3 x_3 + g_4 x_3'$$

$$u_3 = g_5 x_6 + g_6 x_6'$$

$$u_4 = g_7 x_9 + g_8 x_9' ,$$

with

u_i = uniform force on element 3, 6, 9 or 12.

g_i = gain i

x_i = position of joint i

x_i' = derivative of x_i.

The performance measure will consist of two parts: a penalty function component to place real parts of eigenvalues, and a sensitivity component to increase robustness of the final controller design. Choose

$$f(g) = \sum_{i=1}^{18} f_i(g) + \sum_{i=1}^{10} h_i(g) \tag{3-18}$$

where

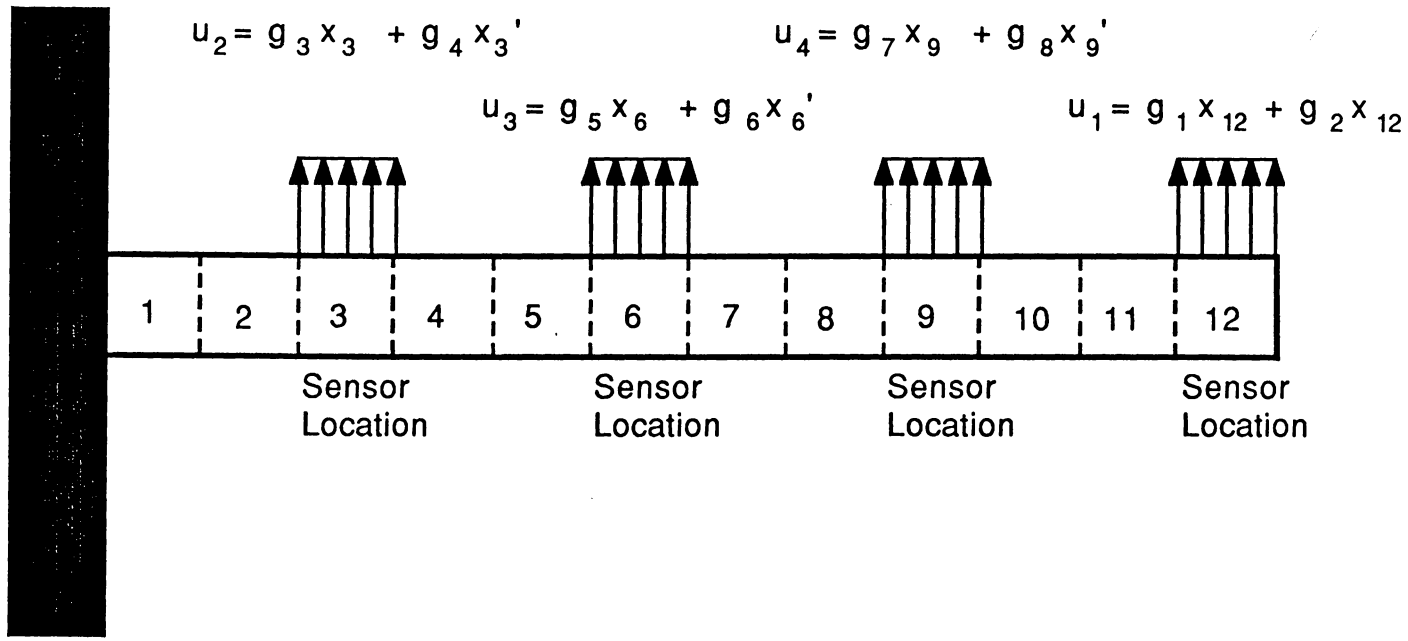


Figure 3.2 The Sensor and Actuator Configuration

$$f_i(\mathbf{g}) = \begin{cases} [\ \text{real}(\lambda_i) + 0.5\ ]^3 & \text{if real}(\lambda_i) > -0.5 \text{ rad./sec.} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$h_i(\mathbf{g}) = 0.001 \sum_{j=1}^{8} \frac{d}{dg_j} \{\ \text{Real}(\lambda_i)\ \}^2 \ .$$

The eigenvalues are ordered with increasing magnitude of the imaginary component. The real parts of the first eighteen eigenvalues are placed to the left of -0.5 rad./sec. by the penalty functions $f_i(\mathbf{g})$. Meanwhile, sensitivities of the first eight eigenvalues to the gains are reduced through the functions $h_i(\mathbf{g})$. Only the first ten eigenvalues are included because of the substantial computations required to calculate second derivatives. The scale factor of 0.001 is chosen after some experimentation. Selecting weights of different portions of the cost is one of the principle difficulties of the optimization method. In the next chapter, the continuation method addresses this problem.

The design problem uses a nominal initial design of

$$\mathbf{g}_0 = \{\ 0, 0, 0, 0\ \}^t \ .$$

Chapter Two contains the necessary derivative formulas for implementing the algorithm in equation (3-1), and a Hammings modified predictor-corrector is used for the integration [Ref. 22]. Numerical results are summarized in Appendix A. Although the gradient method does succeed in moving the selected eigenvalues left, it does not drive the real parts of the eigenvalues to the left of -0.5 rad./sec., as shown in Tables A.4 and A.5. Instead, the method stops as the imaginary parts of the first two eigenvalues are driven to zero. The algorithm stops on a local minimum of the performance measure. This behavior is best

illustrated with the trajectory plots. Figure A.17 contains the gains. In this example, the gains associated with both position and rate feedback are important. Figure A.18 shows the performance measure, and Figure A.19 shows the real parts of the eigenvalues. The most interesting plots are the imaginary components plotted in Figure A.20. Note that the imaginary parts of eigenvalues one and two are going to zero, thereby causing $\lambda_1 \rightarrow \lambda_2$. In this limit, the derivatives of the eigenvalues (and therefore f) go to infinity. The algorithm stops because the automatic stepsize control cannot find a sufficiently large step resulting in a decrease in f. Of course, the large sensitivities in the imaginary component associated with $\lambda_1 \approx \lambda_2$ is not acceptable. An earlier point in the trajectory, showing adequate performance, could be used.

This example also illustrates the restrictions discussed in Section 2.2 on structure of the set of eigenvalues. As discussed in that section, the assumption of distinct eigenvalues is required for the continuous methods discussed in this thesis. As a result, the structure of the set of eigenvalues is fixed. If all of the initial eigenvalues occur in complex pairs, then the continuous methods cannot lead to a final design with real eigenvalues. The double eigenvalue presents a barrier which the methods cannot cross. The above example shows that this barrier can occur in practice; the structure of the initial eigenvalues present a practical limitation of continuous methods.

3.5 *Placing Both Real and Imaginary Components of Eigenvalues*

In this section, two ideas are demonstrated. First, both the real and imaginary components of the eigenvalues are placed through use of penalty functions. The placement of both real and imaginary components allows shaping of the structure's frequency response. In the example here, the structure is made more stable in the presence of low

frequency noise by increasing the imaginary components of the eigenvalues. The example also demonstrates performance of the optimization design method on systems involving a large number of gains. The large number of gains is handled easily; the only cost is the additional computation time.

This example again uses the flexible beam from Chapter Two. See Figure 2.3 for the numbering scheme. Consider sensors and actuators located on elements three, six, nine and twelve, as shown in Figure 3.3. Each of the four actuators applies a uniformly distibuted force over the element, using full feedback of the measurements. This results in thirty-two gains, and the equations

$$\begin{aligned} u_1 &= g_1 x_{12} + g_2 x_{12}' + g_9 x_3 + g_{10} x_3' \\ &\quad + g_{11} x_6 + g_{12} x_6' + g_{13} x_9 + g_{14} x_9' \\ u_2 &= g_3 x_3 + g_4 x_3' + g_{15} x_{12} + g_{16} x_{12}' \\ &\quad + g_{17} x_6 + g_{18} x_6' + g_{19} x_9 + g_{20} x_9' \\ u_3 &= g_5 x_6 + g_6 x_6' + g_{21} x_{12} + g_{22} x_{12}' \\ &\quad + g_{23} x_3 + g_{24} x_3' + g_{25} x_9 + g_{26} x_9' \\ u_4 &= g_7 x_9 + g_8 x_9' + g_{27} x_{12} + g_{28} x_{12}' \\ &\quad + g_{29} x_3 + g_{30} x_3' + g_{31} x_6 + g_{32} x_6' \end{aligned} \tag{3-19}$$

with

u_i = uniformly distributed force on element 3, 6, 9 or 12.

g_i = gain i

x_i = position of joint i

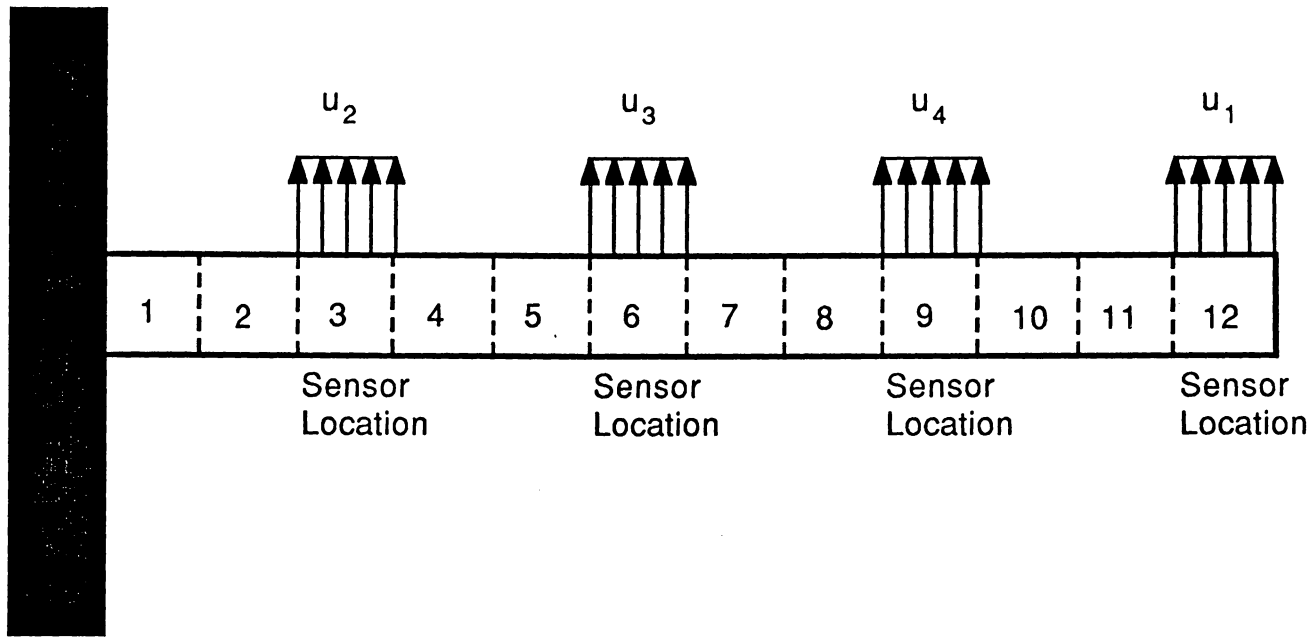


Figure 3.3 The Sensor and Actuator Configuration

x_i' = derivative of x_i.

Two parts are used in the performance measure: a term for placing real components of the eigenvalues, and a term for placing imaginary components of the eigenvalues. Let

$$f(\mathbf{g}) = \sum_{i=1}^{18} f_i(\mathbf{g}) + \sum_{i=1}^{18} h_i(\mathbf{g}) \qquad (3\text{-}20)$$

where

$$f_i(\mathbf{g}) = \begin{cases} [\ \text{real}(\lambda_i) + 0.5\ ]^3 & \text{if real}(\lambda_i) > -0.5 \text{ rad./sec.} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$h_i(\mathbf{g}) = \begin{cases} 0.002\ [\ 5 - |\ \text{imag}(\lambda_i)\ |\ ]^3 & \text{if } |\ \text{imag}(\lambda_i)\ | < 5 \text{ rad./sec.} \\ 0 & \text{otherwise.} \end{cases}$$

The 0.002 is a weighting factor to balance the magnitude and importance of the terms. As in previous examples, the first eighteen eigenvalues are pushed to real parts less than -0.5 by the penalty function f_i. The penalty function h_i forces the imaginary parts of the eigenvalues to magnitudes greater than five.

The sample problem is executed using a Hammings modified predictor - corrector, [Ref. 22], with

$$g_0 = \underline{0}.$$

Appendix A summarizes the numerical results. Tables A.6 and A.7 show the initial and final values of various parameters. Note in Figure A.21 that the cost approaches zero, as

desired. Figures A.22 to A.25 show the trajectories of the gains. Figures A.26 and A.27 show the real and imaginary eigenvalues. The optimization method is completely successful in placing the eigenvalues in the desired regions.

Chapter Four

The Continuation Method

The continuation method designs a control law for equation (2-1) by solving a set of nonlinear constraint equations

$$\mathbf{F}(\mathbf{g}) = \mathbf{0}, \qquad \mathbf{F}: \mathbf{R}^m \rightarrow \mathbf{R}^n \tag{4-1}$$

where

$$m \leq n.$$

$\mathbf{F}(\mathbf{g})$ can be functions of eigenvalues and eigenvectors, sensitivities, and other robustness measures. Choosing $\mathbf{F}(\mathbf{g})$ to restrict the eigenvalues to specific areas of the complex plane will prove particularly useful. Again, any performance measure which can be described as a differentiable function of the gain can be included. This chapter describes in detail the use of a continuation method to design control laws. The first section discusses the continuous minimum norm algorithm and establishes its continuation properties. Then several examples are given which use the bending beam problem to illustrate the method. Appendices C and D contain sample listings of the computer code used in these sections.

4.1 *The Minimum Norm Algorithm*

A solution is attempted for (4-1) by imbedding the equation in a continuum of problems dependant on a continuation variable t. First, the equation is embedded in the set of intermediate problems

$$\mathbf{H}(\mathbf{g}, t) = \mathbf{F}(\mathbf{g}) + h(t)\,\mathbf{F}(\mathbf{g_0}) = \mathbf{0} \tag{4-2}$$

where $\mathbf{g_0}$ is an initial design point and h(t) has some special properties. The choice of $\mathbf{g_0}$ is critical because of the limitations of continuous methods when placing eigenvalues. This

is discussed in Section 2.3. The continuation algorithm, through the choice of $\mathbf{g_0}$, can be used to refine an existing and adequate design. Or it can be used to substantially improve a primitive design. h is chosen continuous and differentiable with the properties

$$h(0) = -1 \tag{4-3}$$

and

$$h(t_{end}) = 0 \quad \text{for some } t_{end} \in (0, \infty].$$

The statement

$$h(t_{end}) = 0 \quad \text{for } t_{end} = \infty \tag{4-4}$$

should be interpreted as

$$h(t) \rightarrow 0 \text{ as } t \rightarrow \infty \ . \tag{4-5}$$

This gives the continuation **H** some very special characteristics. When t=0, we have

$$\mathbf{H}(\mathbf{g_0}, 0) = 0. \tag{4-6}$$

The continuation method attempts to solve (4-1) by generating trajectories $\mathbf{g}(t_{end})$ in such a way that

$$\mathbf{H}(\mathbf{g}(t), t) = \mathbf{0} \tag{4-7}$$

for all $t \in [0, t_{end}]$. Then the terminal vector $\mathbf{g}(t_{end})$ is the solution of (4-1) since

$$\mathbf{H}(\mathbf{g}(t_{end}), t_{end}) = \mathbf{F}(\mathbf{g}(t_{end})) + 0\ \mathbf{F}(\mathbf{g_0}) = \mathbf{0}. \tag{4-8}$$

The trajectories are generated by solving the differential equation

$$d\mathbf{g}(t)/dt = \mathbf{v} \tag{4-9}$$

$$\mathbf{g}(0) = \mathbf{g_0},$$

where the derivatives **v** are solutions of

$$dH/dt = \partial F/\partial g \; \mathbf{v} + dh/dt \; \mathbf{F}(\mathbf{g_0}) = \mathbf{0}. \tag{4-10}$$

Note that (4-10) must have rank m for a solution to exist. Insufficient rank is particularly a problem when n=m. This case calls for the use of a much more advanced and mathematically elegant procedure called the Chow-Yorke homotopy method [Ref. 17,18]. In the control problems here, the number of gains (n) will generally be far greater than the number of constraint equations (m). In the author's experience, insufficient rank has only occurred when one of the components has reached an extrema. In this case, the user must reevaluate the initial design $\mathbf{g_0}$ or the design constraints **F**(**g**).

The solutions **v** of (4-10) are not uniquely determined. The vector **v** must be chosen so that the differential equation (4-9) has a solution. The author was unsuccessful in extending the Chow-Yorke theorems to the under determined case. Instead, a minimum norm algorithm is used. This is motivated by a desire to meet the constraint equations (4-1) while moving a small distance from the initial design $\mathbf{g_0}$. Consider the minimization of

$$1/2 \; \mathbf{v}^t \; \mathbf{W} \; \mathbf{v} \; , \tag{4-11}$$

where **W** is a positive definite diagonal weighting matrix, subject to the linear constraint of equation (4-10)

$$\partial \mathbf{F}/\partial \mathbf{g} \; \mathbf{v} = - dh/dt \; \mathbf{F}(\mathbf{g_0}) . \tag{4-12}$$

Using Lagrange multipliers, the augmented function is

$$\Psi = 1/2 \, \mathbf{v}^t \mathbf{W} \mathbf{v} + \lambda^t (\partial \mathbf{F}/\partial \mathbf{g} \; \mathbf{v} + dh/dt \; \mathbf{F}(\mathbf{g_0})) . \tag{4-13}$$

Differentiating Ψ with respect to **v** results in

$$\mathbf{W}\,\mathbf{v} + (\partial \mathbf{F}/\partial \mathbf{g})^t\,\lambda = \mathbf{0}\ . \tag{4-14}$$

Differentiating Ψ with respect to λ results in

$$\partial \mathbf{F}/\partial \mathbf{g}\ \mathbf{v} = -\,dh/dt\ \mathbf{F}(\mathbf{g_0})\ . \tag{4-15}$$

Then, from (4-14) and (4-15),

$$\mathbf{v} = -\,\mathbf{W}^{-1}\,(\partial \mathbf{F}/\partial \mathbf{g})^t\,[\,\partial \mathbf{F}/\partial \mathbf{g}\ \mathbf{W}^{-1}\,(\partial \mathbf{F}/\partial \mathbf{g})^t\,]^{-1}\ dh/dt\ \mathbf{F}(\mathbf{g_0})\ . \tag{4-16}$$

In summary, the continuation method takes the initial condition $\mathbf{g_0}$ and integrates (4-9) using (4-16), resulting in

$$d\mathbf{g}/dt = -\,\mathbf{W}^{-1}\,(\partial \mathbf{F}/\partial \mathbf{g})^t\,[\,\partial \mathbf{F}/\partial \mathbf{g}\,\mathbf{W}^{-1}\,(\partial \mathbf{F}/\partial \mathbf{g})^t\,]^{-1}\ dh/dt\ \mathbf{F}(\mathbf{g_0}) \tag{4-17}$$

$$\mathbf{g}(0) = \mathbf{g_0}\ .$$

When t equals (or approaches) t_{end}, then $\mathbf{g}(t_{end})$ is the final design.

Any number of choices for h(t) is possible.

$$h(t)=(t-1) \tag{4-19}$$

results in an extension of Smale's Algorithm to the solution of sets of under determined equations. Reference [19] describes Smale's algorithm.

4.2 A Robust Continuation Map

The author has discovered a particular choice of h(t) which will prove especially robust and useful. Let

$$h(t) = e^{-t}. \tag{4-19}$$

Then

$$dh/dt = -\,h \tag{4-20}$$

and the continuum **H** in (4-2) is

$$\mathbf{H}(\mathbf{g},t) = \mathbf{F}(\mathbf{g}) + e^{-t}\mathbf{F}(\mathbf{g_0}) = \mathbf{0}\ . \tag{4-21}$$

Note that

$$\mathbf{F}(\mathbf{g}) = -e^{-t}\mathbf{F}(\mathbf{g_0}) = dh/dt\ \mathbf{F}(\mathbf{g_0})\ . \tag{4-22}$$

Substituting this into (4-17) results in

$$d\mathbf{g}/dt = -\mathbf{W}^{-1}(\partial\mathbf{F}/\partial\mathbf{g})^t\ [\partial\mathbf{F}/\partial\mathbf{g}\ \mathbf{W}^{-1}(\partial\mathbf{F}/\partial\mathbf{g})^t\ ]^{-1}\mathbf{F}(\mathbf{g}) \tag{4-23}$$

$$\mathbf{g}(0) = \mathbf{g_0}\ .$$

This equation of this version of the algorithm is not explicitly dependant on either $\mathbf{g_0}$ or t. During the numerical integration of (4-23), errors will accumulate in $\mathbf{g}(t)$. Say, for example, that the errors are described by

$$\mathbf{g_{numerical}}(t_{mid}) = \mathbf{g_{ideal}}(t_{mid}) + \mathbf{error}(t_{mid}) \tag{4-24}$$

at some

$$t_{mid} \in (0,\infty).$$

The algorithm (4-23) can simply be viewed as restarting with

$$\mathbf{g_0} = \mathbf{g_{numerical}}(t_{mid})\ . \tag{4-25}$$

The numerical trajectory $\mathbf{g_{numerical}}(t)$ is moving along a trajectory near $\mathbf{g_{ideal}}(t)$ which still converges to the solution as $t\rightarrow\infty$. This robustness to numerical errors is of great practical importance. As a result, the version of the continuation algorithm in (4-23) will be used during the following illustrations.

4.3 *Placing Eigenvalues with a Large Number of Gains*

The optimization methods of Chapter Three present a problem in implementation. In most practical problems, more than one performance measure must be included in the

design process. The optimization method requires that these different measures be combined into a single cost function, usually through the use of weights. These weights may be very difficult to determine a priori, so considerable experimentation is necessary. The examples in this section demonstrate the greater flexibility allowed by the continuation method. In both examples, two cost functions are chosen: the first controls the real parts of the eigenvalues, and the second controls the imaginary parts. In the first example, the method is wholly successful. In the second example, one of the common difficulties of the continuation method is demonstrated.

Consider again the flexible beam of Chapter Two, with the state numbering system of Figure 2.3. These examples will use the same sensor and actuator configuration as section 3.5 and Figure 3.3. See equation (3-19) for a complete description of the situation. There are four actuators, four sensors and thirty-two gains. Complete feedback is implemented. In the first example of this section, the cost functions are

$$f_1(g) = \sum_{i=1}^{18} h_i(g) \qquad (4\text{-}26)$$

with

$$h_i(g) = \begin{cases} [\,\mathrm{Real}(\lambda_i) + 0.5\,]^3 & \text{if } \mathrm{Real}(\lambda_i) > -0.5 \text{ rad./sec.} \\ 0 & \text{otherwise ,} \end{cases}$$

and

$$f_2(g) = \sum_{i=1}^{18} p_i(g)$$

with

$$p_i(g) = \begin{cases} [\,5 - |\,\mathrm{Imag}(\lambda_i)\,|\,]^3 & \text{if } |\,\mathrm{Imag}(\lambda_i)\,| < 5 \text{ rad./sec.} \\ 0 & \text{otherwise .} \end{cases}$$

The cost functions are used to drive the real parts of the first eighteen eigenvalues to the left of -0.5 rad./sec. and the magnitude of the imaginary component above 5rad./sec. A nominal design of

$$g_0 = \underline{0}$$

is combined with a Hammings modified predictor - corrector integration scheme [Ref. 22]. Chapter Two provides the necessary derivative formulas to implement equation (4-23), and Appendix A contains the numerical results. The continuation method successfully uses the above penalty functions, as shown in Tables A.8 and A.9. Figure A.28 shows the cost functions, Figures A.29 through A.32 the gains, and Figures A.33 and A.34 the eigenvalues.

The sensor and actuator configuration above can be used to illustrate one of the common difficulties of continuous design methods. The cost functions are now used to drive the real parts of the first eighteen eigenvalues to the left of -0.5 rad./sec. and the magnitude of the imaginary component above 20 rad./sec. Consider

$$f_1(g) = \sum_{i=1}^{18} h_i(g) \qquad (4\text{-}27)$$

with

$$h_i(g) = \begin{cases} [\,\mathrm{Real}(\lambda_i) + 0.5\,]^3 & \text{if } \mathrm{Real}(\lambda_i) > -0.5 \text{ rad./sec.} \\ 0 & \text{otherwise ,} \end{cases}$$

and

$$f_2(g) = \sum_{i=1}^{18} p_i(g)$$

with

$$p_i(g) = \begin{cases} [\,20 - |\,\mathrm{Imag}(\lambda_i)\,|\,]^3 & \text{if } |\,\mathrm{Imag}(\lambda_i)\,| < 20 \text{ rad./sec.} \\ 0 & \text{otherwise .} \end{cases}$$

Now the continuation method reduces the cost by two orders of magnitude, but fails to drive the costs to zero. Tables A.10 and A.11 and Figures A.35 through A.41 show the initial conditions and output. The algorithm fails because of step size problems; the numerical integration package reaches an area in the gain space in which small changes in gains result in large changes in eigenvalues. The stepsize is reduced until the algorithm is making no progress along the continuation map, and execution is stopped. At this intermediate point in the continuation map, several eigenvalues are not even stable. However, as shown in Figure A.42, the determinant of the matrix inverted in equation (4-23) is still clearly nonzero. In theory, the algorithm can move through this region. In practice, the algorithm stops for numerical reasons. This behavior has also been observed when using the optimization ideas in Chapter Three.

4.4 *Robust Design Through Sensitivities*

The continuation method presented in this chapter provides a great deal of flexibility through the choice of multiple cost functions. This flexibility is especially useful when robust design is required. In Section 3.4, the optimization method is used to provide a robust design. The optimization method allows only a single cost, so the penalty

function for eigenvalue placement is combined with a term reflecting sensitivities, as shown in equation (3-29). Note that the optimization method requires a choice of weight between the two cost terms. This weight is chosen through trial and error, a very costly process. The optimization method also fails to drive the eigenvalues into the appropriate portion of the complex plane. The method stops on a local minimum, resulting in an inadequate design. The weight could possibly be adjusted to remove this minimum, but understanding the interactions between cost terms is difficult. This section demonstrates the advantage of the continuation method by considering another example of robust design through sensitivities. The continuation method does not require a weighting of different costs functions.

Once more consider the flexible beam described in Chapter Two. The element and state numbering is provided in Figure 2.3, and the four actuator and sensor configuration of Figure 3.3 in Section 3.5 is applied. Equation (3-19) shows the configuration of the thirty-two gains used in the full feedback. Two costs functions are used. The first implements a penalty function to place the first eighteen eigenvalues, while the second controls eigenvalue sensitivities. Let

$$f_1(\mathbf{g}) = \sum_{i=1}^{18} h_i(\mathbf{g}) \qquad (4\text{-}28)$$

with

$$h_i(\mathbf{g}) = \begin{cases} [\,\mathrm{Real}(\lambda_i) + 0.5\,]^3 & \text{if } \mathrm{Real}(\lambda_i) > -0.5 \text{ rad./sec.} \\ 0 & \text{otherwise ,} \end{cases}$$

and

$$f_2(\mathbf{g}) = \sum_{i=1}^{4} p_i(\mathbf{g}) - 0.01$$

with

$$p_i(\mathbf{g}) = 0.001 \sum_{j=1}^{32} \frac{d}{dg_j} \{ \text{Real}(\lambda_i) \}^2 .$$

The performance measure f_1 drives the real parts of the first eighteen eigenvalues below -0.5 rad./sec. when $f_1(\mathbf{g})=0$. The function f_2 reduces the quadratic measure of sensitivity to approximately a third of the original value when $f_2(\mathbf{g})=0$.

The robustness measure f_2 not only improves the final design, but also improves the performance of the numerical design method. In several cases, in both the optimization and continuation design method, the design method fails because of high sensitivities. The design process results in an intermediate set of eigenvalues which are extremely sensitive to changes in the gains. The algorithm then fails because no appropriate step size can be found. Controlling the sensitivities during the design process avoids this problem and often results in more efficient computations.

The equation (4-23) is implemented using a Hammings modified predictor - corrector, [Ref. 22], and the initial condition

$$\mathbf{g}_0 = \underline{0}.$$

Appendix A contains the numerical results. Tables A.12 to A.13 show the initial and final designs. Note that the design objectives are met. Figure A.43 shows the decrease in costs, while Figures A.44 to A.47 show the gains. Finally, Figures A.48 and A.49 plot

the appropriate real and complex eigenvalues. The eigenvalues are driven by the penalty function f_1 into the desired region of the complex plane, and the sensitivities are controlled by the measure f_2.

Chapter Five

Concluding Remarks

The control of flexible structures presents a number of challenging problems. This thesis considers one aspect of the complete problem. Two closely related methods are developed for the design of feedback controllers for high-dimensioned linear systems. These methods offer three advantages. First, the design laws require no observer. Design and implementation of an observer for a flexible body may be a difficult task. Second, no a priori assumption is made about the number of modes which can be controlled. The methods do not suffer from the restriction of only one mode controlled for each actuator. Finally, the approaches used here allow mission specific performance measures. In principle, any function of the eigenvalues, eigenvectors, sensitivities of the eigenvalues and vectors, gains or other design parameters can be chosen. The design methodology is easy to implement for a variety of performance measures, but substantial computations are required. Unfortunately, this flexibility has a price. No assurance is available that the methods will lead to an acceptable design.

The feedback control problem is approached by first developing a mathematical model and equations for the appropriate sensitivities. A simple finite element model of a cantilevered beam is derived for use in examples. The optimization and continuation methods are developed. Examples demonstrating both successes and failures are provided. Extensive numerical output is available in Appendix A, and sample listings of computer code are included in Appendices B, C and D.

This thesis shows the promise of parameter optimization and continuation design methods. In the examples here, the open loop system is used as the nominal design by setting initial gain values to zero. From this open loop system, feedback controllers with the desired performance are constructed. However, this does not demonstrate the real capability of the optimization methods. These methods would be best used as the second

half of a two part process. First, a more theoretical approach would be used to place poles in the desired locations. Then, using this as a nominal design, the optimization or continuation method would be used to "tune" the design to match specific mission requirements. Thus adequate performance would be assured, while very general performance measures would be applied.

References

1. Balas, M. J., "Some Trends in Large Space Structure Control Theory: Fondest Hopes; Wildest Dreams," Proceedings of the Joint Automatic Control Conference, Denver, Colorado, June 1979, pp. 42-55.

2. Balas, M. J., "Direct Output Feedback Control of Large Space Structures," The Journal of the Astronautical Sciences, Vol. XXVII, No. 2, April-June 1979, pp. 157-180.

3. Kimura, H., "A Further Result on the Problem of Pole Assignment by Output Feedback," IEEE Transactions on Automatic Control, June 1977, pp. 458-463.

4. Wu, Y. W., Rice, R. B., and Juang, J. N., "Control of Large Flexible Space Structures Using Pole Placement Design Techniques", Journal of Guidance and Control, Vol. 4, No. 3, May-June 1981, pp. 298-303.

5. Balas, M. J., "Direct Velocity Feedback Control of Large Flexible Space Structures," Journal of Guidance and Control, Vol. 2, No. 3, May-June 1979, pp. 252-253.

6. Balas, M. J., "Feedback Control of Flexible Systems," IEEE Transactions on Automatic Control, Vol. AC-23, No. 4, August 1978, pp.673-679.

7. Balas, M. J., "Finite Element Models and Feedback Control of Flexible Aerospace Structures", Proceedings of the 1980 Joint Automatic Control Conference, Vol. II, pp. FP1-D.

8. Polak, E., et al., "DELIGHT.MEMO: An Interactive, Optimization-Based Multivarate Control System Design Package," IEEE Control Systems Magazine, Vol. 2, No. 4, December 82, pp. 9-14.

9. Nye, B., Tits, A., "DELIGHT.SPICE: An Optimization-Based System for the Design of Integrated Circuits," IEEE 1983 Custom Integrated Circuits Conference, Rochester, New York, pp. 23-25.

10. Dunyak, J. P., Junkins, J. L., Watson, L. T., "Eigenvalue Optimization for High-Dimensioned Feedback Control Systems," presented at The 22nd IEEE Conference on Decision and Control, San Antonio, Texas, December 1983.

11. Anderson, L., "Direct design of Multivariate Control Systems Through Singular Value Decomposition," presented at the AAS/AIAA Astrodynamics Conference, Lake Placid, NY, 1983.

12. Newsom, J. and Mukhopadhyay, V., "The Use of Singular Value Gradients and Optimization Techniques to Design Robust Controllers for Multiloop Systems," presented at the AAS/AIAA Astrodynamics Conference, Lake Placid, NY, 1983.

13. Davison, E. J., and Ferguson, I. J., "The Design of Controllers for the Multivariable Robust Servomechanism Problem Using Parameter Optimization Methods," IEEE Transactions on Automatic Control, Vol. AC-26, No. 1, February 1981, pp. 93-110.

14. Schulz, G., and Heimbold, G., "Dislocated Actuator / Sensor Positioning and Feedback Design for Flexible Structures," Journal of Guidance, Vol. 6, No. 5, Sept.-Oct. 1983, pp. 361-367.

15. Skaar, S. B., Arbitrary Large-Angle Satellite Attitude Maneuvers Using an Optimal Reaction Wheel Power Critereon, dissertation submitted to Virginia Polytechnic Institute and State University, Blacksburg, Virginia, August 1982.

16. Dunyak, J. P., Junkins, J. L., and Watson, L. T., "Robust Nonlinear Least Squares Estimation Using the Chow-Yorke Homotopy Method," Journal of Guidance, Control, and Dynamics, Vo. 7, No. 6, November-December 1984, pp. 752-754.

17. Watson, L. T., "A Globally Convergent Algorithm for Computing Fixed Points of C^2 Maps," Applied Mathematics and Computation, Vol. 5, 1979, pp. 298-311.

18. Chow, S. N., Mallet-Paret, J., Yorke, J. A., "Finding Zeros of Maps: Homotopy Methods That Are Constructive with Probability One," Mathematics of Computation, Vol. 32, No. 143, July 1978, pp. 887-899.

19. Smale, S., "A Convergent Process of Price Adjustment and Global Newton Methods," Journal of Mathematical Economics, Vol. 3, 1976, pp. 1-14.

20. Fox, R. L., and Kapoor, M. P., "Rates of Change of Eigenvalues and Eigenvectors," AIAA Journal, December 1968, pp. 2426-2429.

21. Meirovitch, L., Elements of Vibrational Analysis, McGraw Hill, 1975.

22. Ralston, Wilf, Mathematical Methods for Digital Computers, Wiley, 1960, pp. 95-109.

Appendix A

Numerical Results

Table A.1

Approximate and Exact Open Loop Frequencies

Approximate rad./sec.	Exact rad./sec.
1.112	1.111
6.968	6.967
19.51	19.51
38.5	38.23
63.3	63.20
94.7	94.41
132	131
177	175
228	225
287	281
353	344
420	413
540	488
631	569
741	656
868	750
1015	850
1182	956
1371	1068
1582	1186
1810	1312
2034	1442
2214	1580
2727	1724

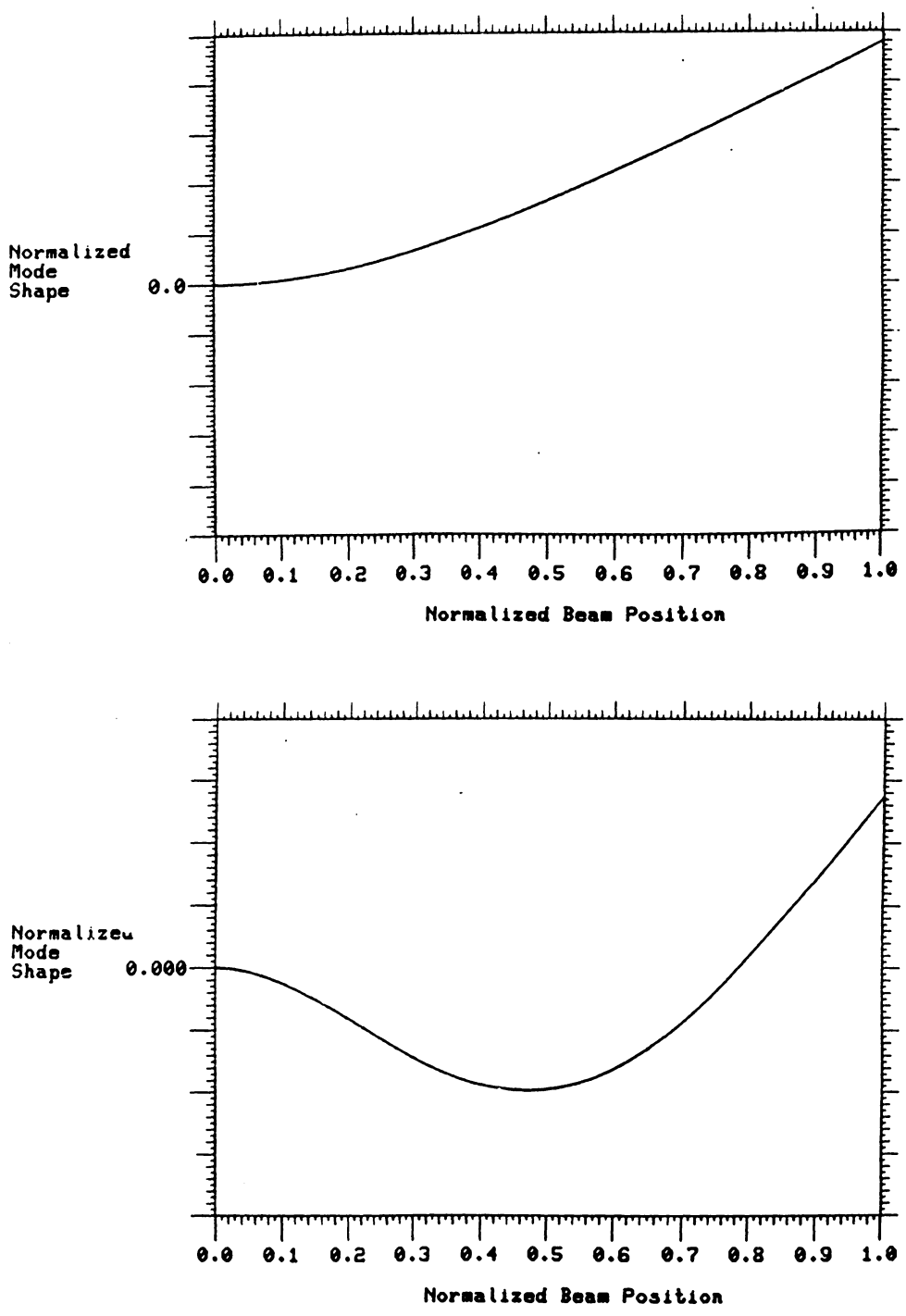


Figure A.1 Modes 1 and 2

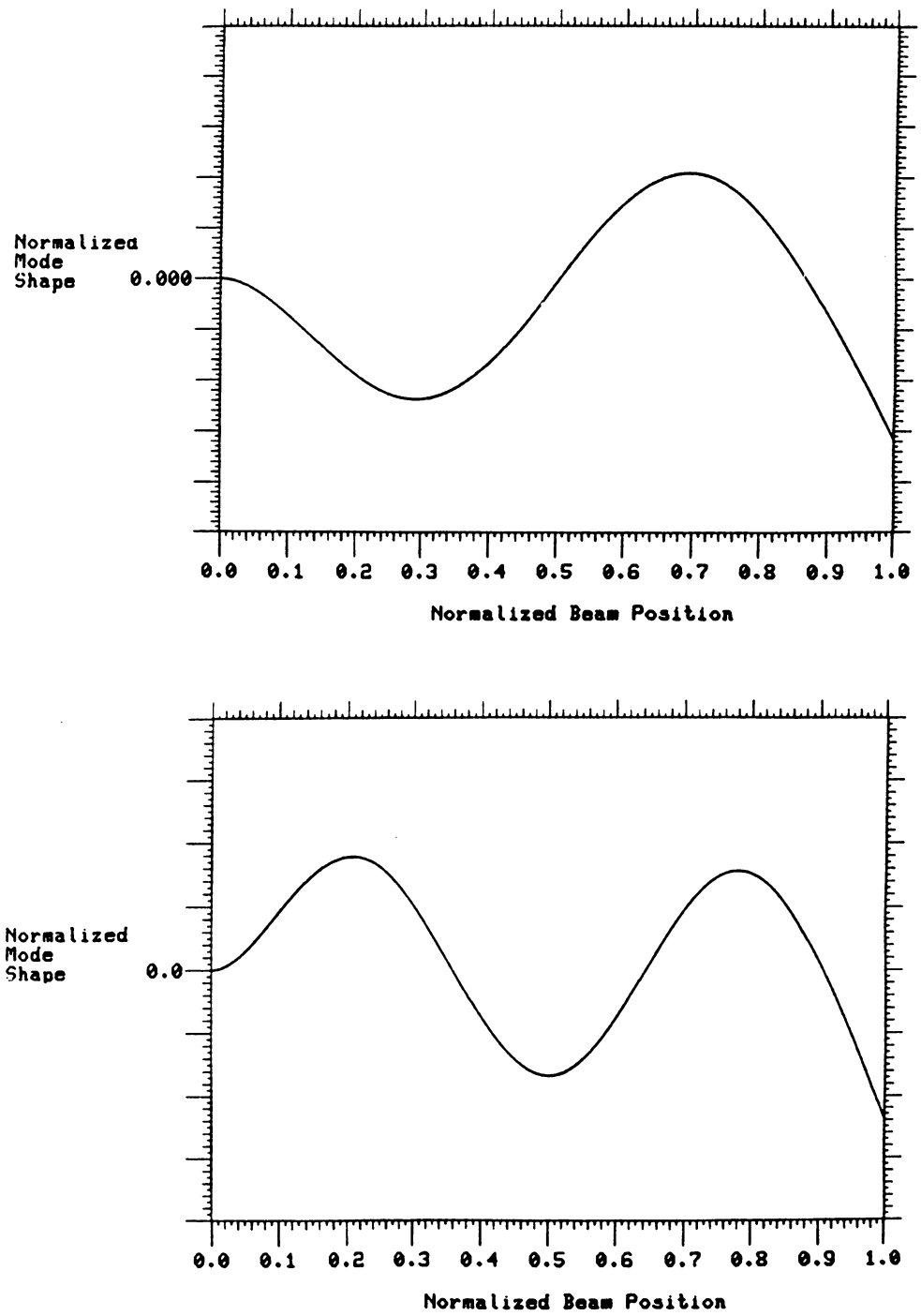


Figure A.2 Modes 3 and 4

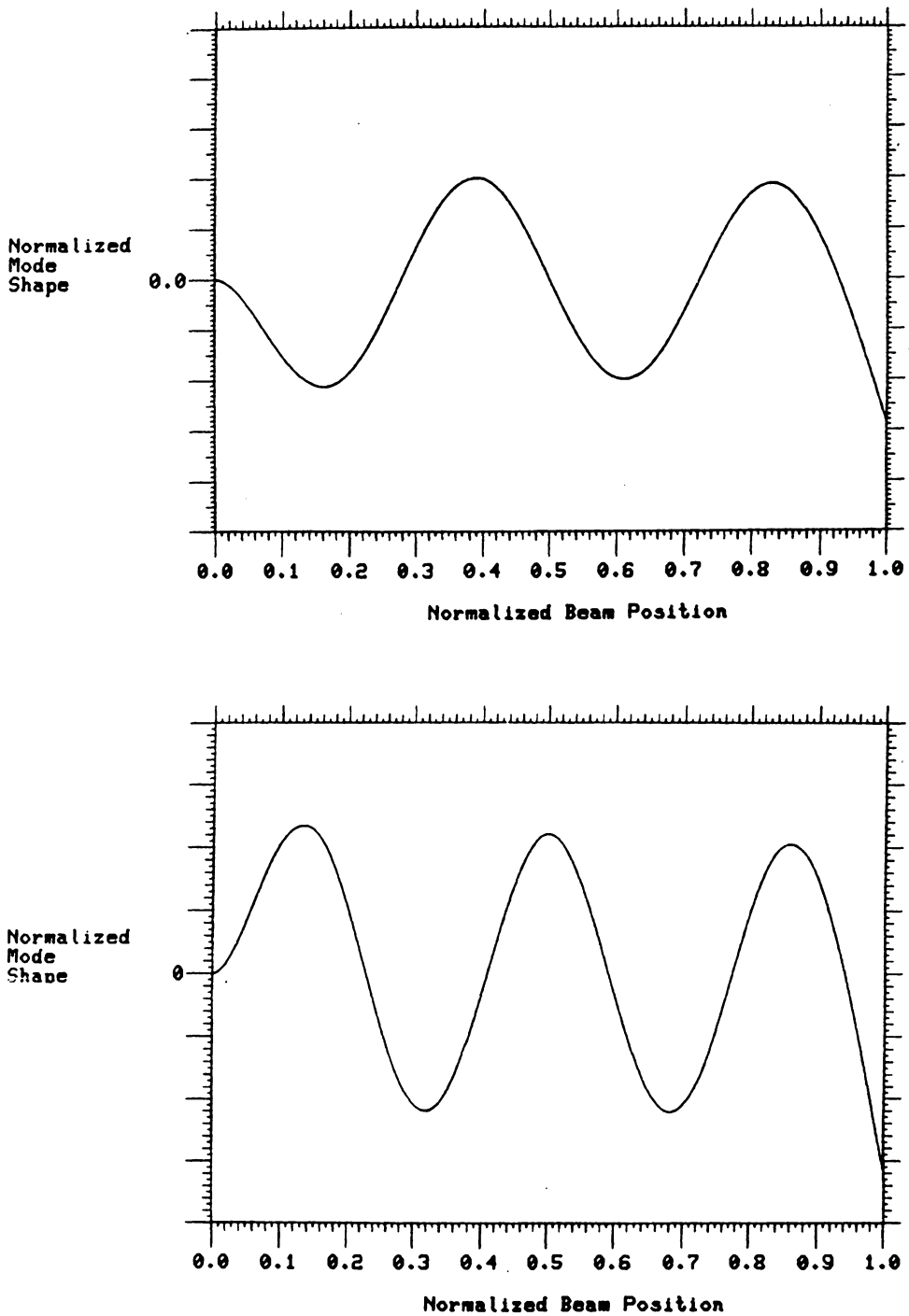


Figure A.3 Modes 5 and 6

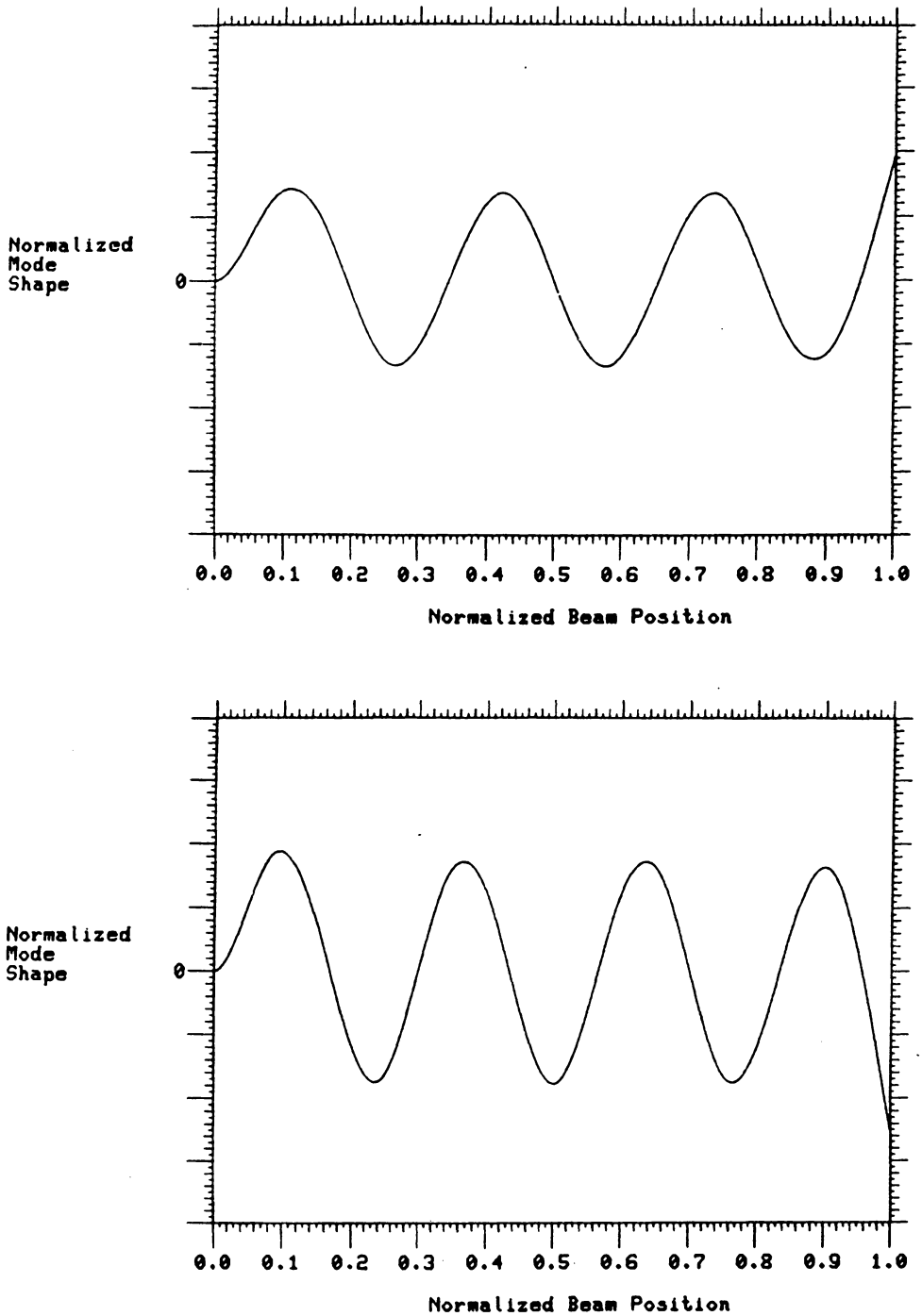


Figure A.4 Modes 7 and 8

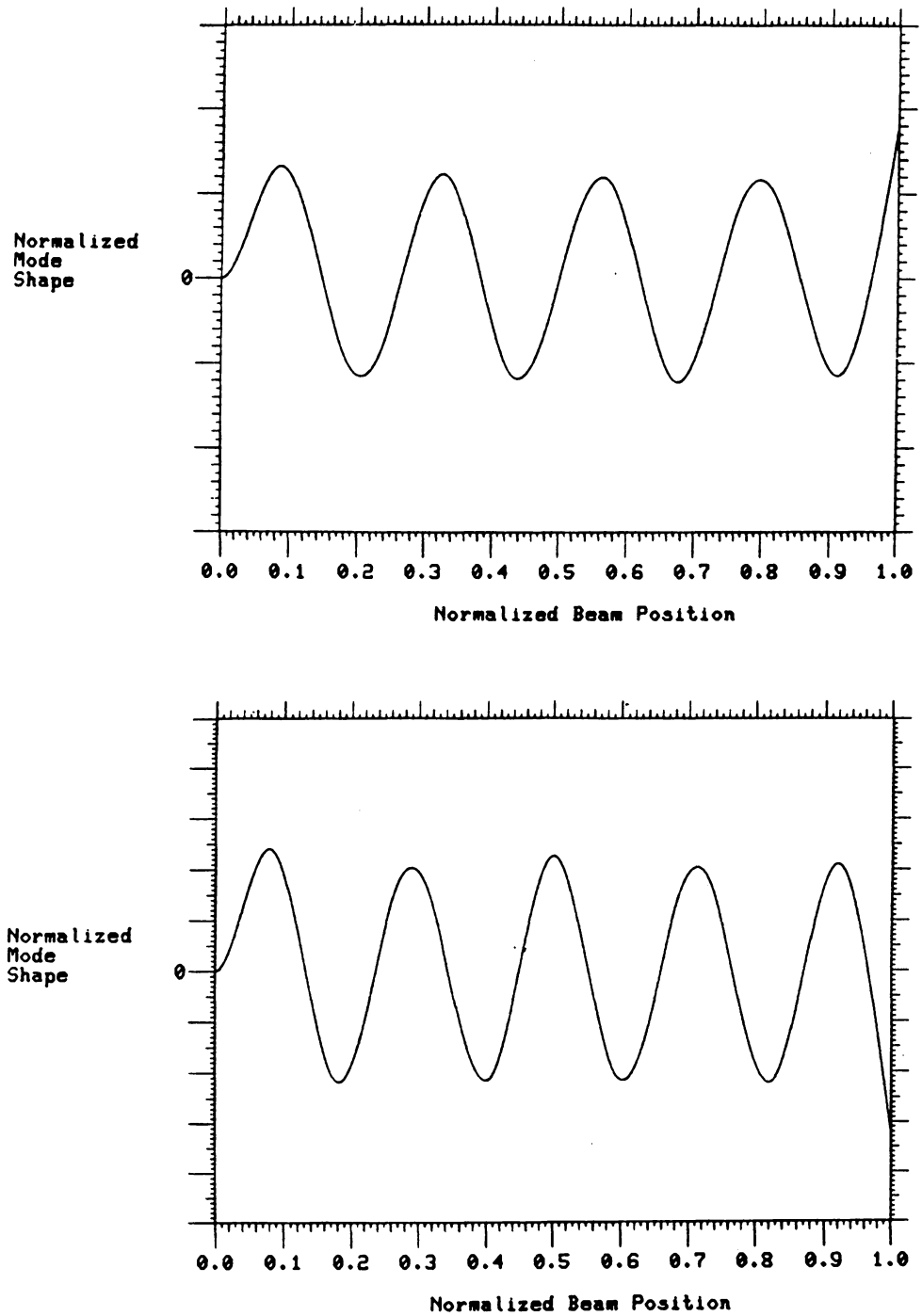


Figure A.5 Modes 9 and 10

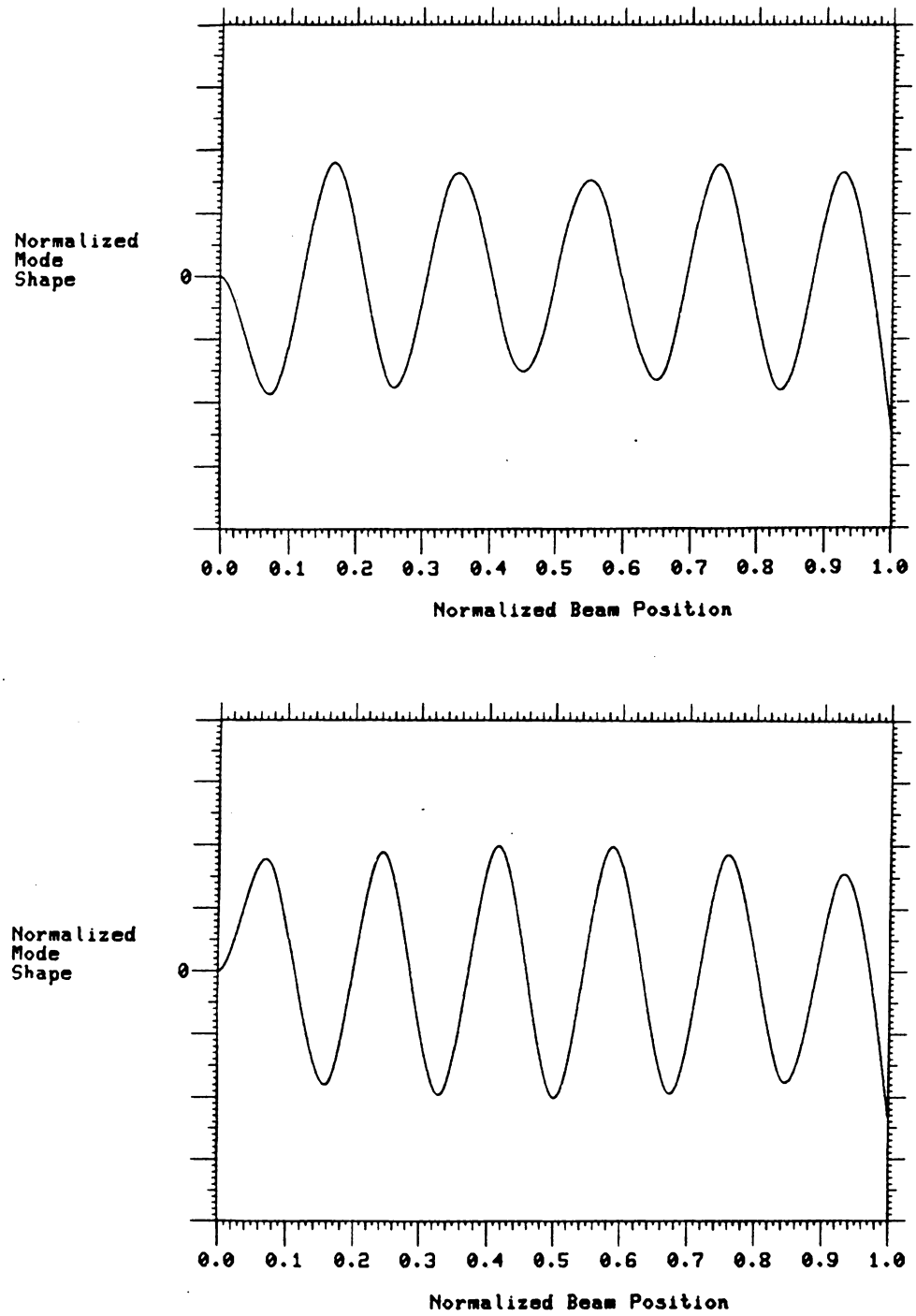


Figure A.6 Modes 11 and 12

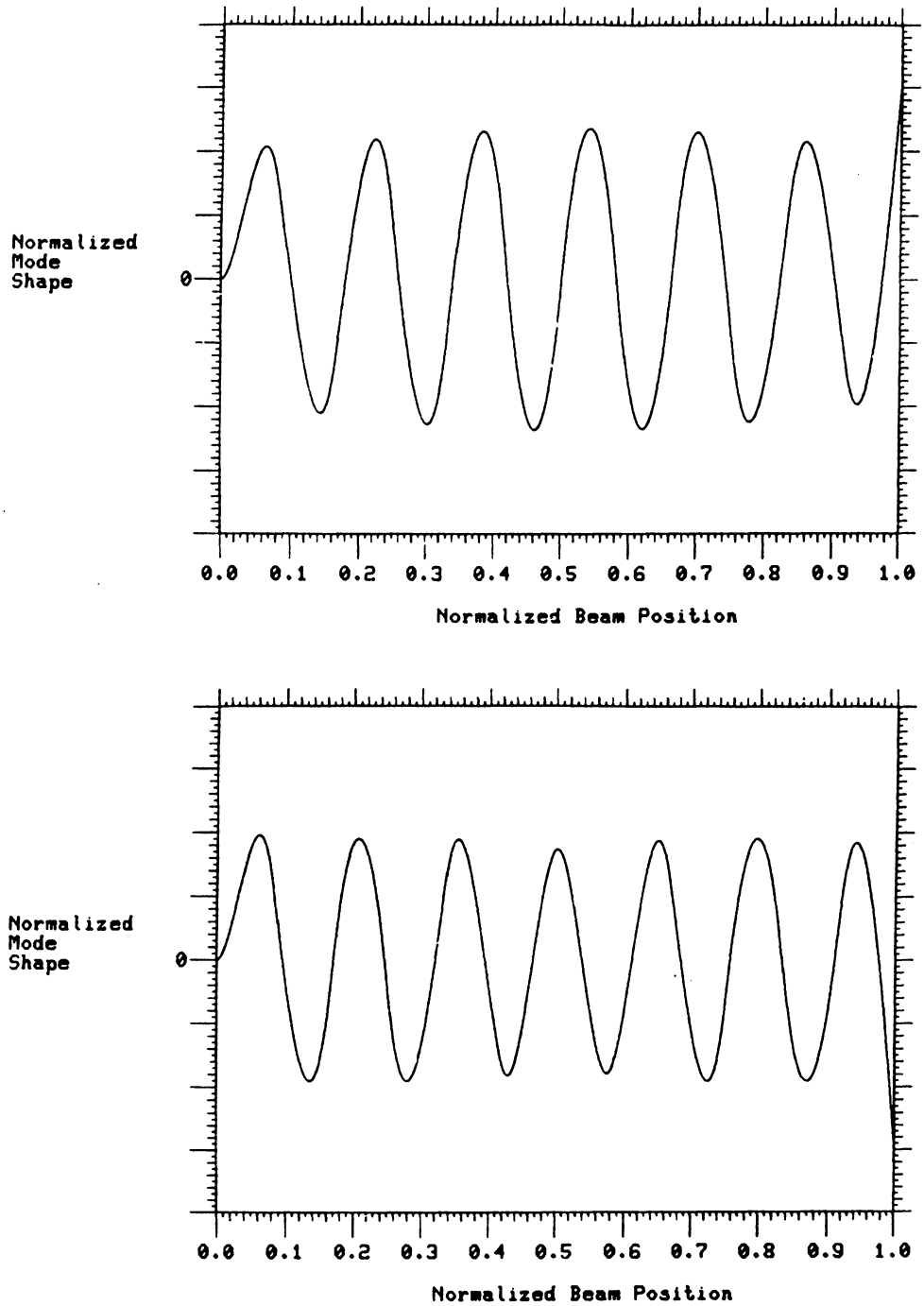
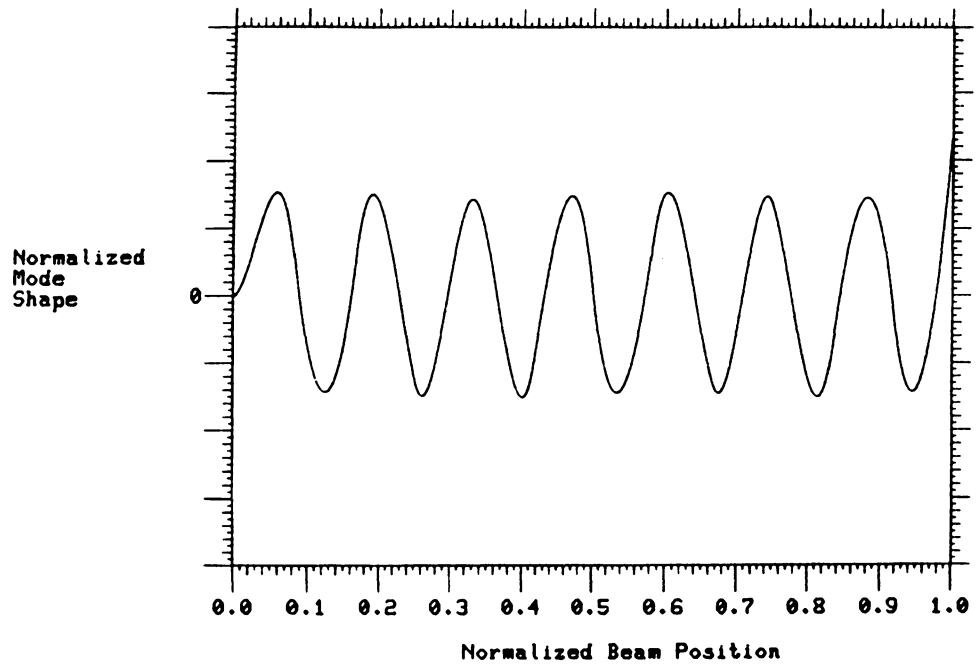


Figure A.7 Modes 13 and 14



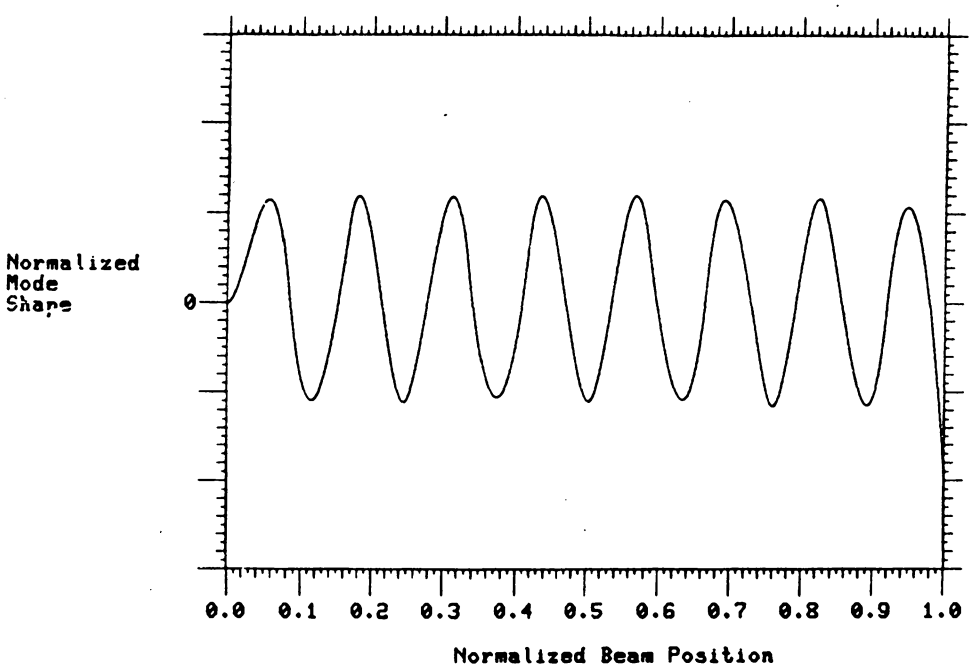


Figure A.8 Modes 14 and 16

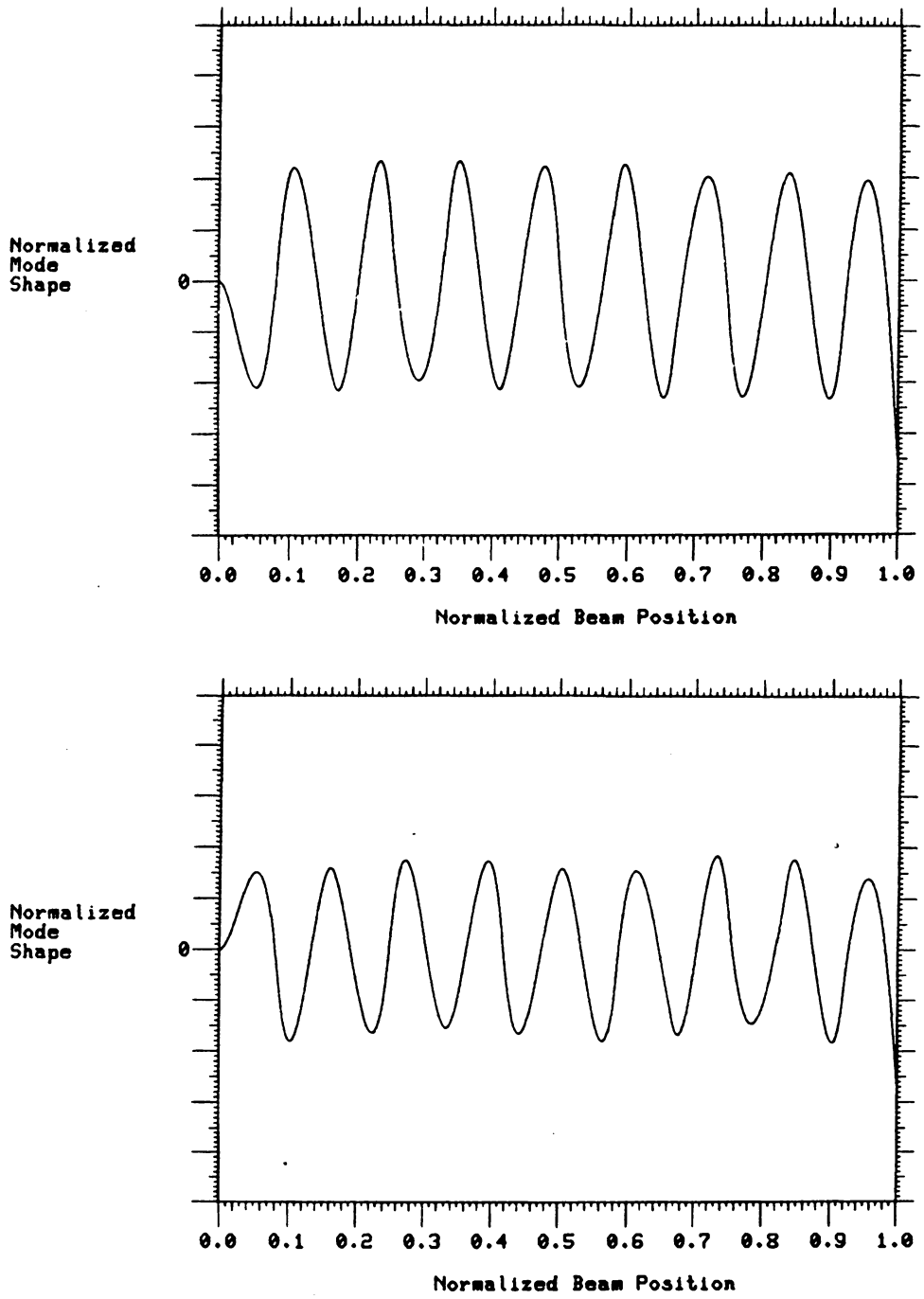
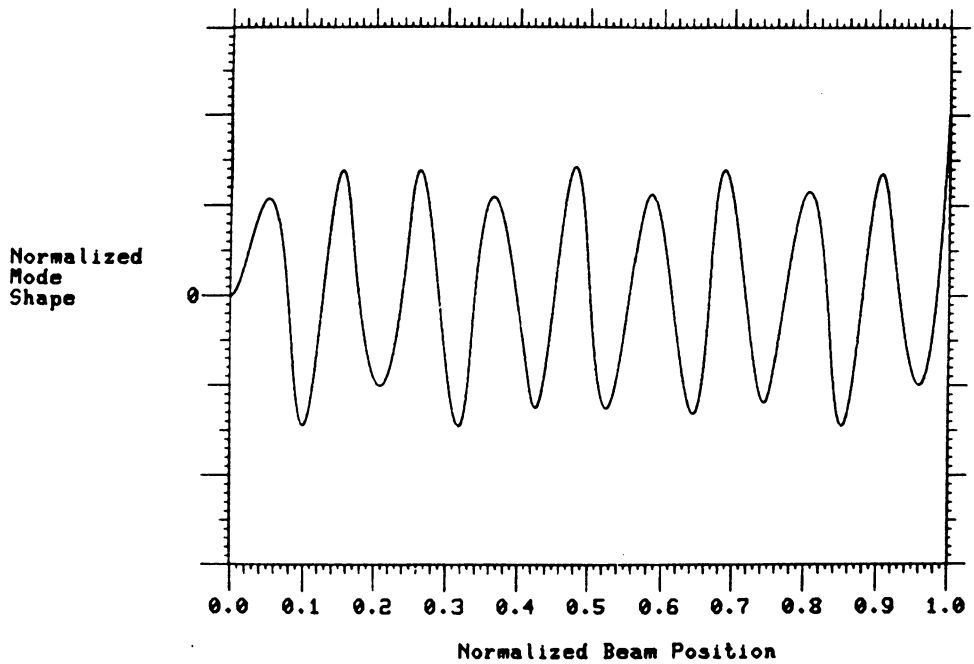


Figure A.9 Modes 17 and 18



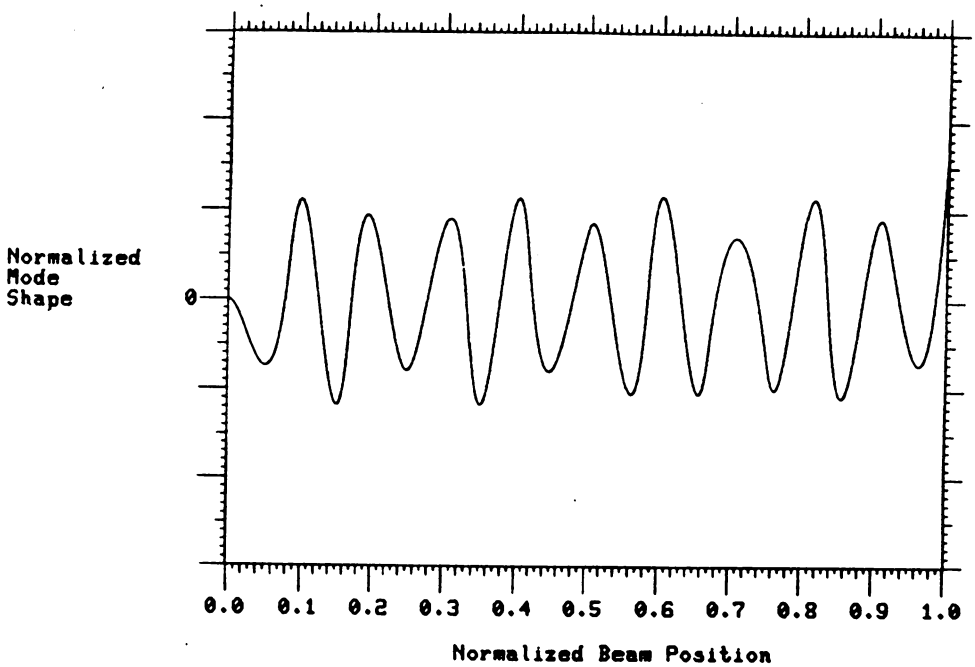
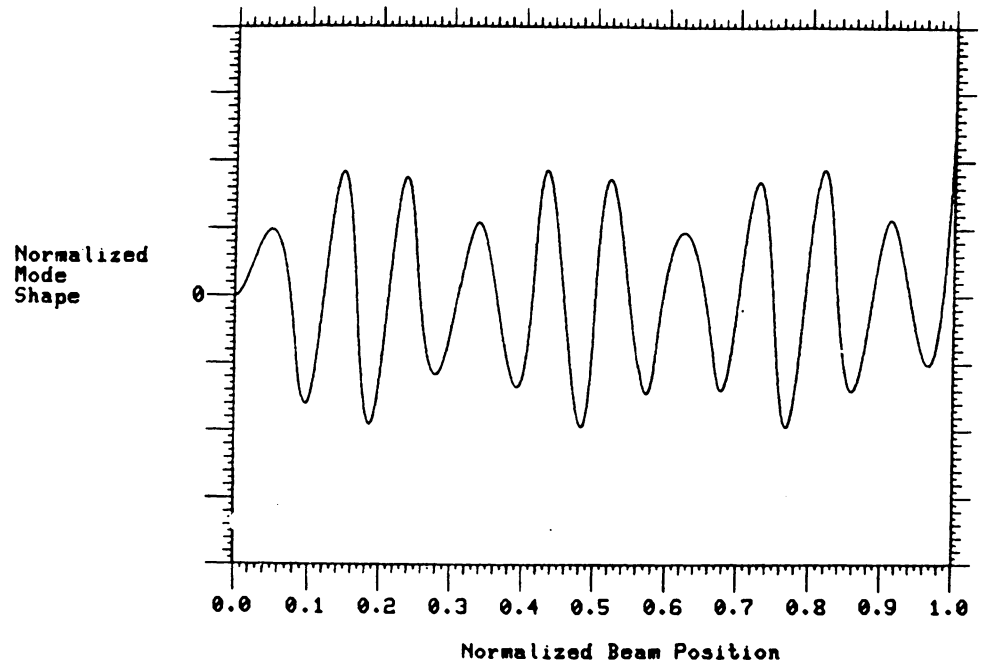


Figure A.10 Modes 19 and 20



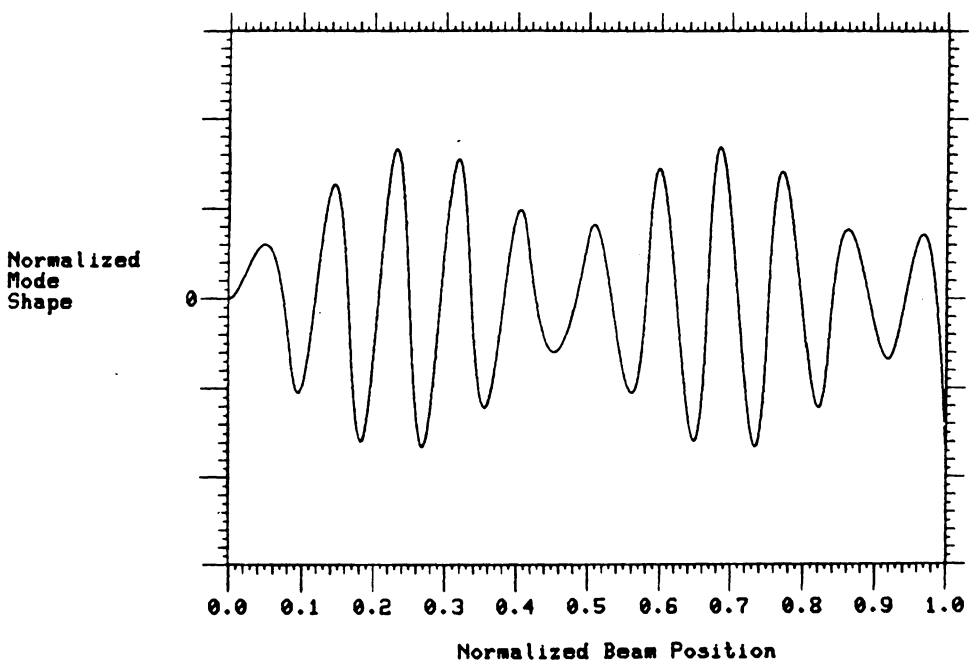


Figure A.11 Modes 21 and 22

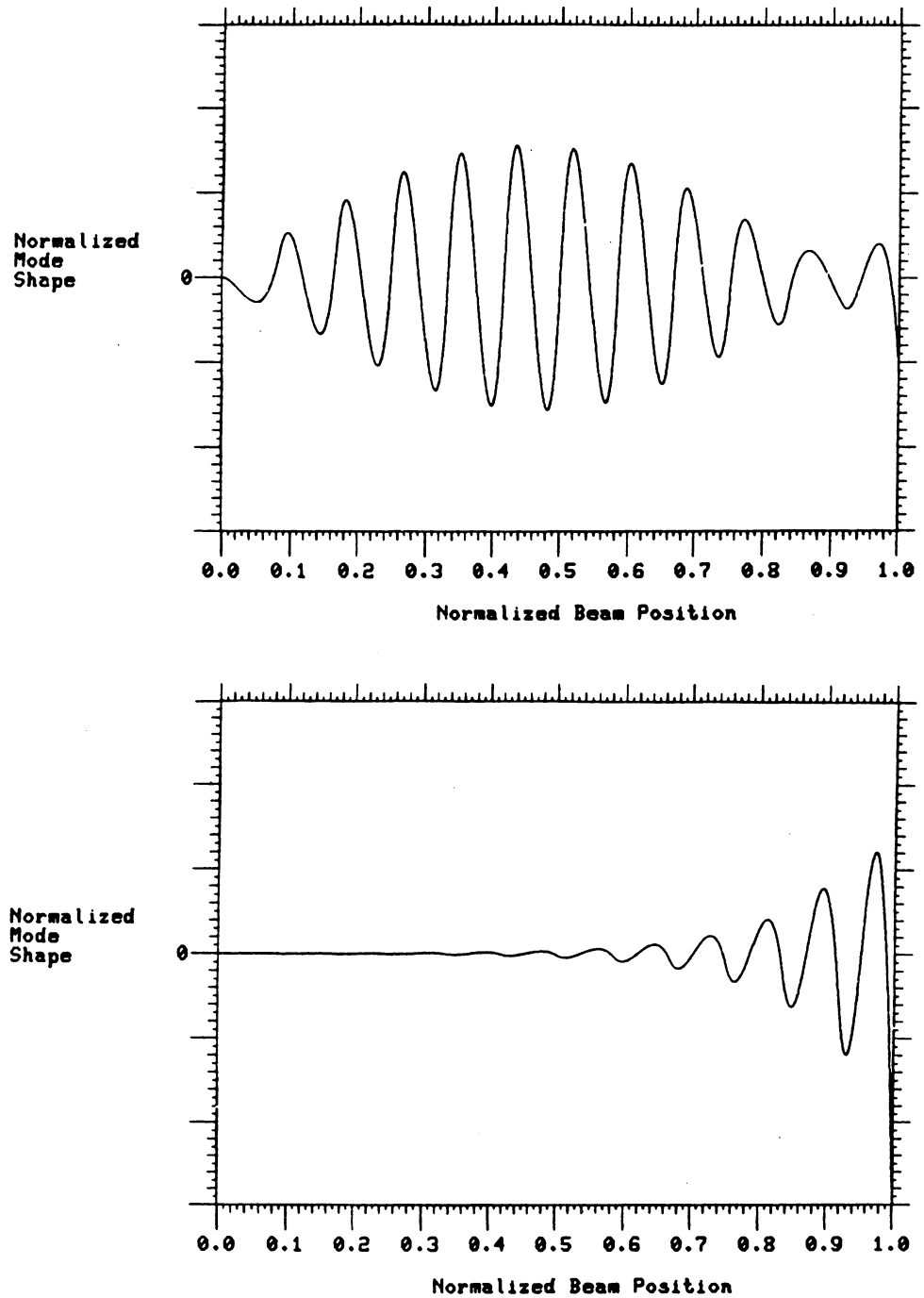


Figure A.12 Modes 23 and 24

Table A.2
Gains and Parameters

Parameter	Initial Value	Final Value
tau	0	121
cost	1.0	5×10^{-5}
g_1	0	-0.001
g_2	0	-0.31
g_3	0	2×10^{-5}
g_4	0	-0.16

Table A.3

Eigenvalue Initial and Final States

Eigenvalue	Initial Value rad./sec.		Final Value rad./sec.	
λ_1	0	+ i 1.11	-0.61	+ i 0.95
λ_2	0	- i 1.11	-0.61	- i 0.95
λ_3	0	+ i 6.97	-0.53	+ i 6.87
λ_4	0	- i 6.97	-0.53	- i 6.87
λ_5	0	+ i 19.5	-0.56	+ i 19.5
λ_6	0	- i 19.5	-0.56	- i 19.5
λ_7	0	+ i 38.5	-0.50	+ i 38.2
λ_8	0	- i 38.5	-0.50	- i 38.2

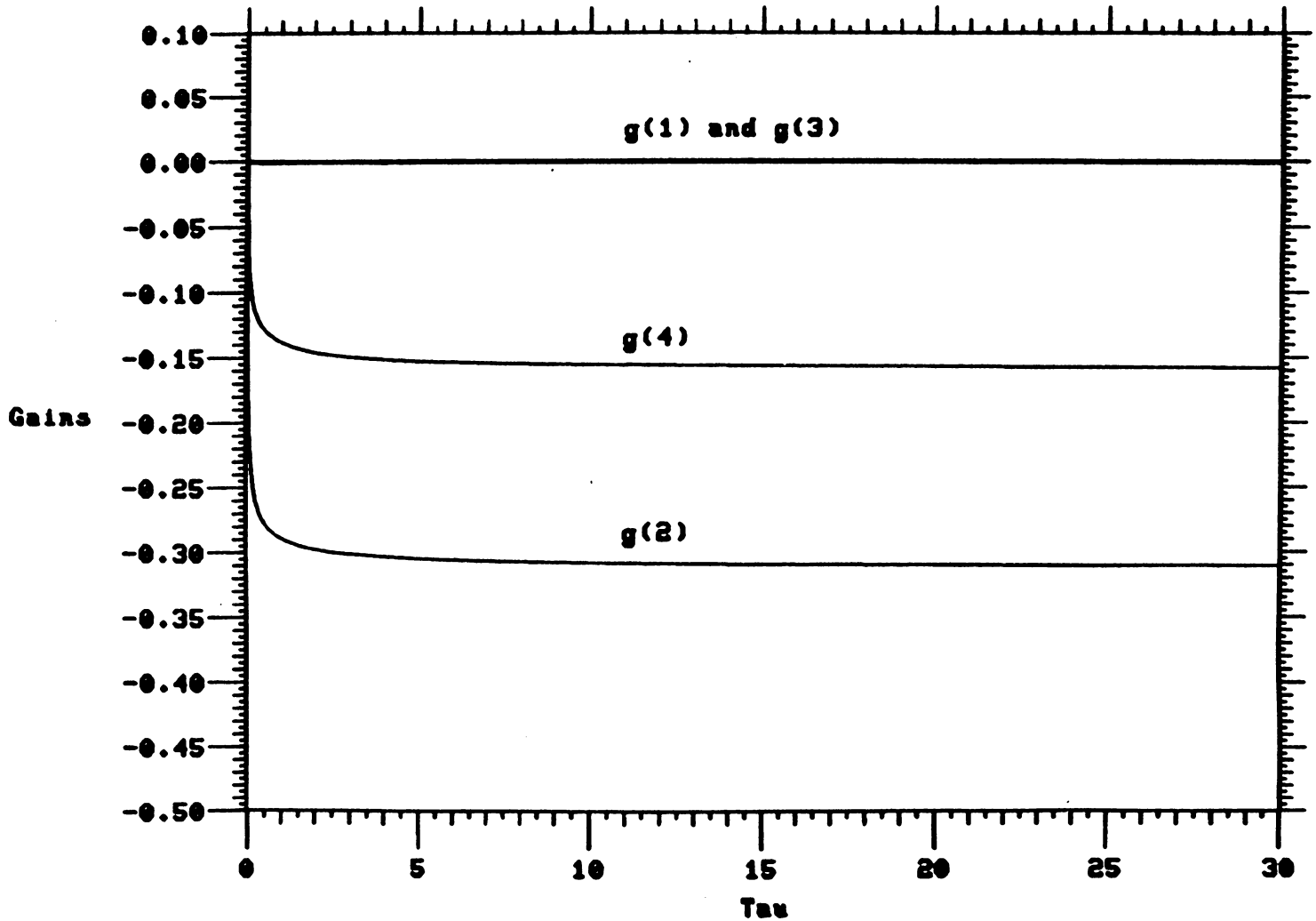


Figure A.13 Gain Plots

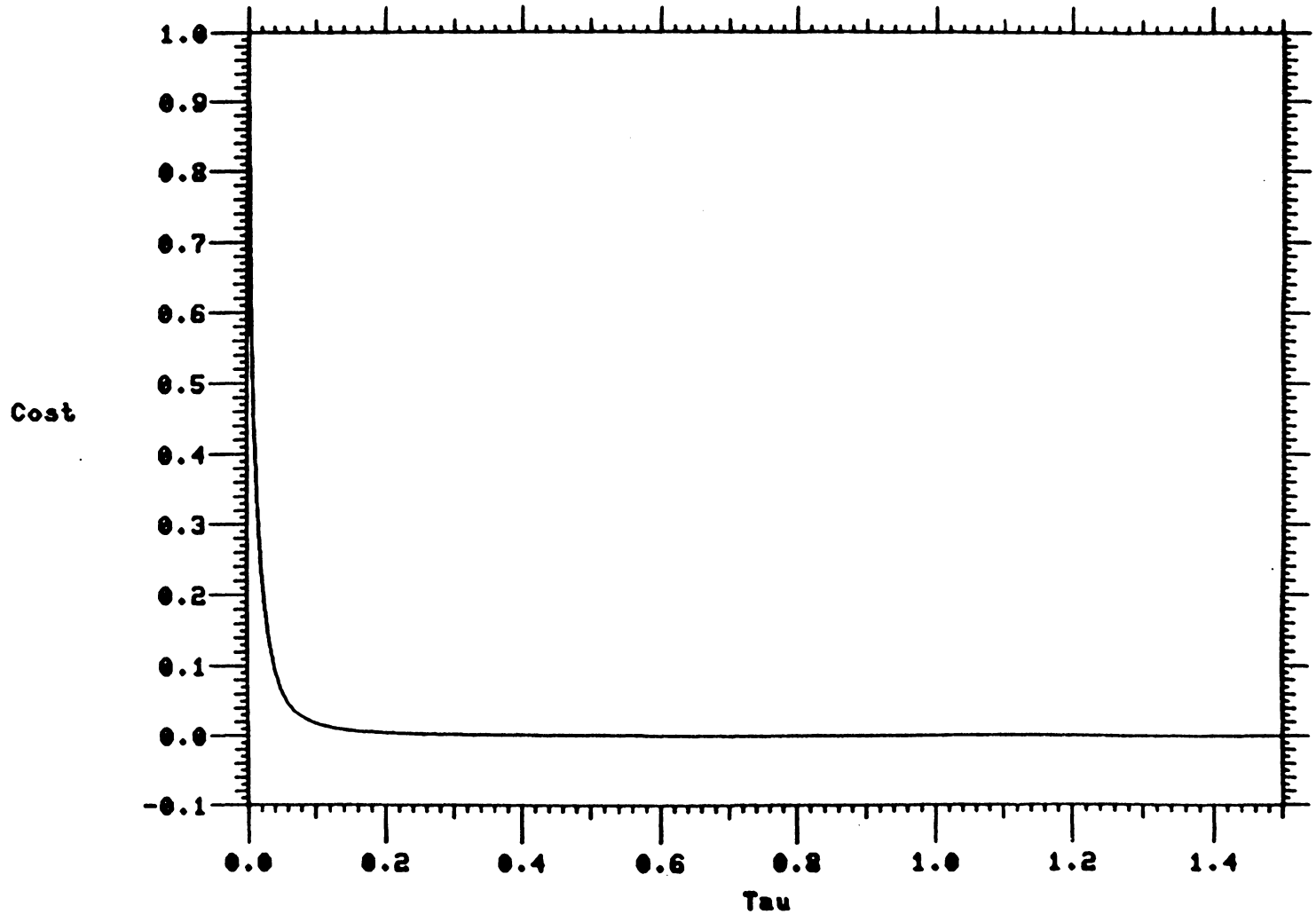


Figure A.14 Cost Trajectory

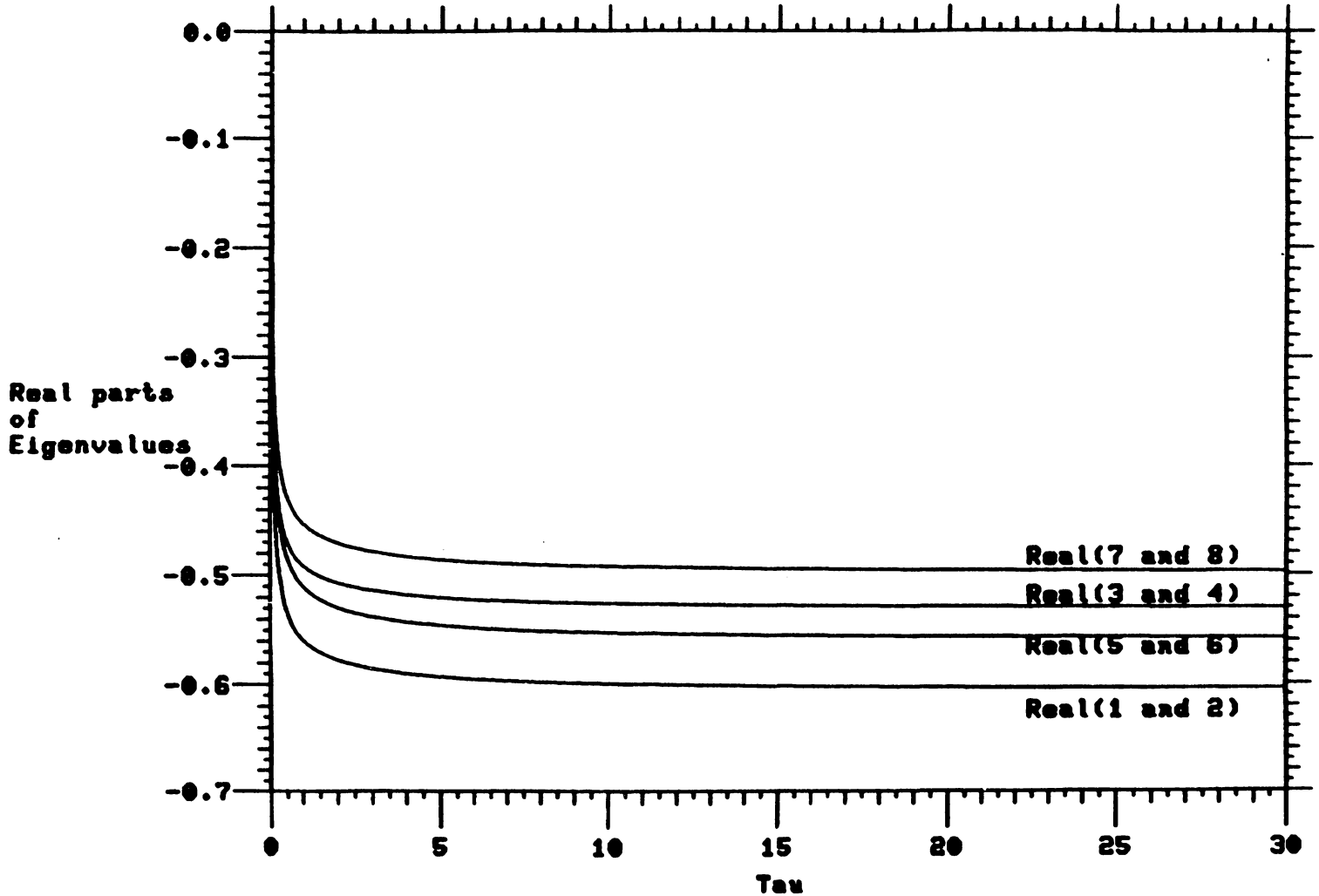


Figure A.15 Real Parts of Eigenvalues

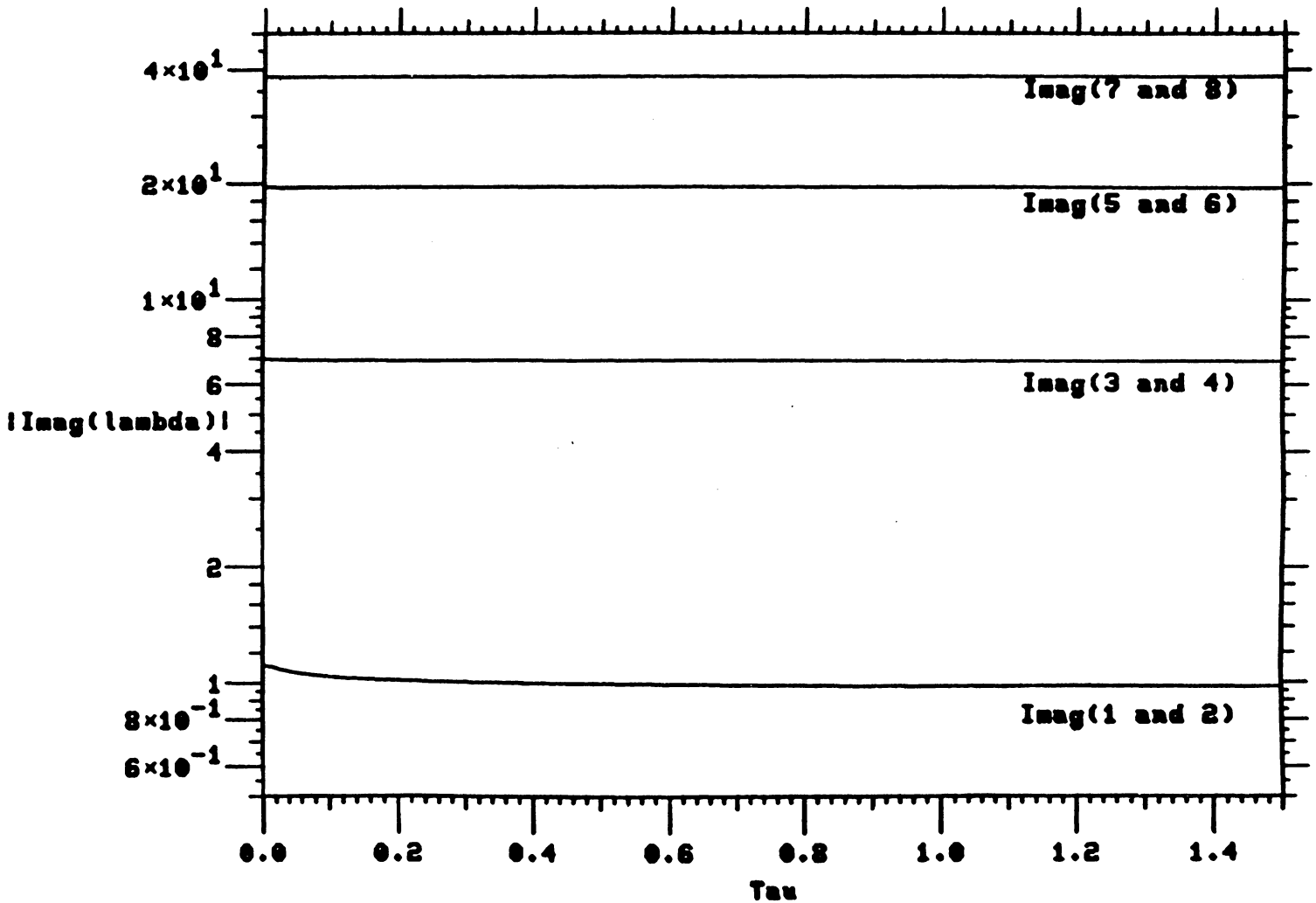


Figure A.16 Imaginary Parts of Eigenvalues

Table A.4

Gains and Parameters

Parameter	Initial Value	Final Value
tau	0	2.68
cost	2.28	0.06
g_1	0	-0.0038
g_2	0	-0.53
g_3	0	-5.4×10^{-5}
g_4	0	-0.79
g_5	0	5.4×10^{-4}
g_6	0	-0.28
g_7	0	7.3×10^{-4}
g_8	0	-0.012

Table A.5

Eigenvalue Initial and Final States

Eigenvalue	Initial Value rad./sec.		Final Value rad./sec.	
λ_1	0	+ i 1.11	-1.15	+ i 0.02
λ_2	0	- i 1.11	-1.15	- i 0.02
λ_3	0	+ i 6.97	-1.31	+ i 6.7
λ_4	0	- i 6.97	-1.31	- i 6.7
λ_5	0	+ i 19.5	-1.43	+ i 19
λ_6	0	- i 19.5	-1.43	- i 19
λ_7	0	+ i 38.5	-1.62	+ i 38
λ_8	0	- i 38.5	-1.62	- i 38
λ_9	0	+ i 63.3	-0.71	+ i 63
λ_{10}	0	- i 63.3	-0.71	- i 63
λ_{11}	0	+ i 94.7	-0.43	+ i 94
λ_{12}	0	- i 94.7	-0.43	- i 94
λ_{13}	0	+ i 132	-0.42	+ i 132
λ_{14}	0	- i 132	-0.42	- i 132
λ_{15}	0	+ i 177	-0.75	+ i 177
λ_{16}	0	- i 177	-0.75	- i 177
λ_{17}	0	+ i 228	-0.26	+ i 228
λ_{18}	0	- i 228	-0.26	- i 228

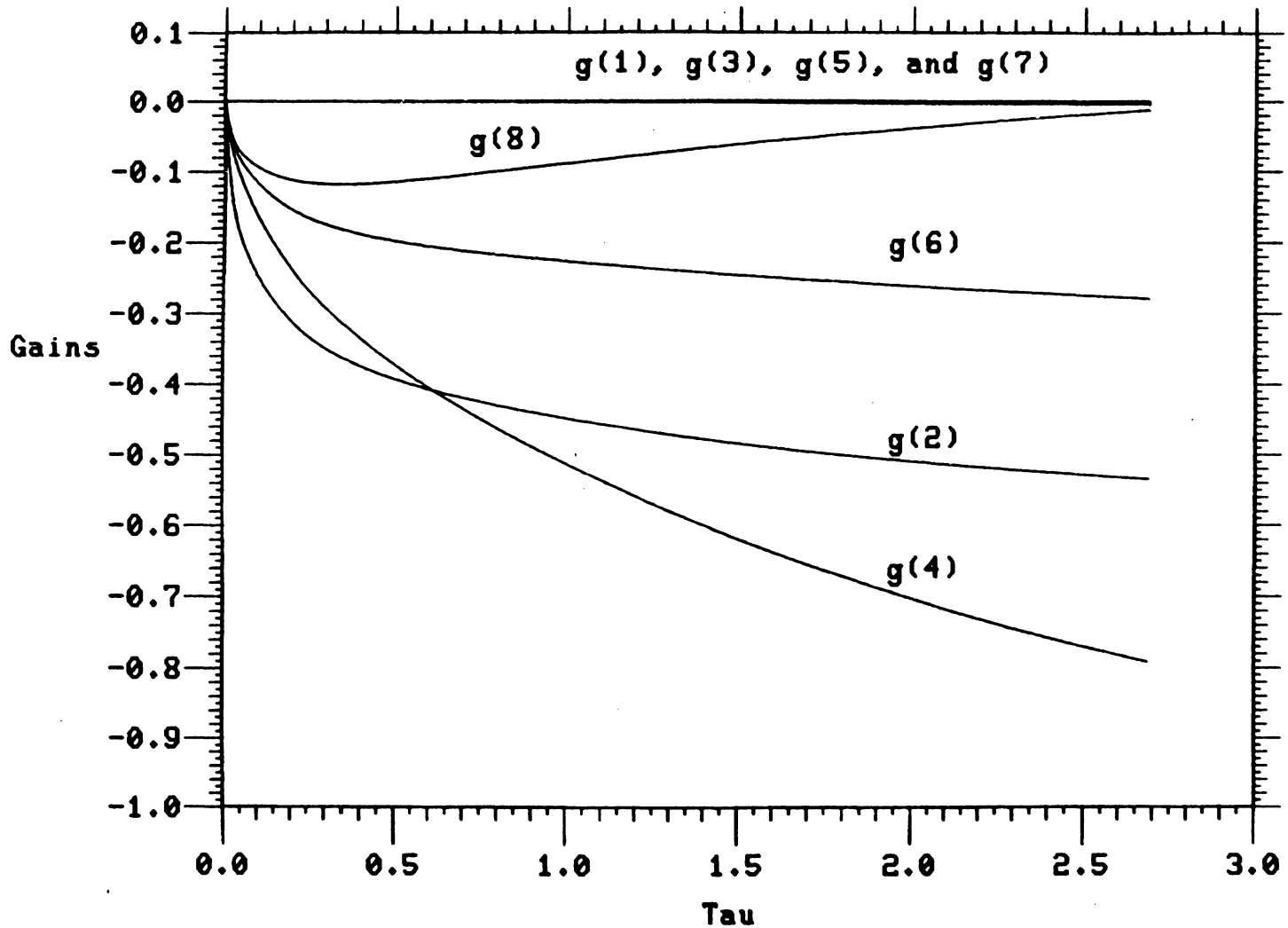


Figure A.17 Gain Plots

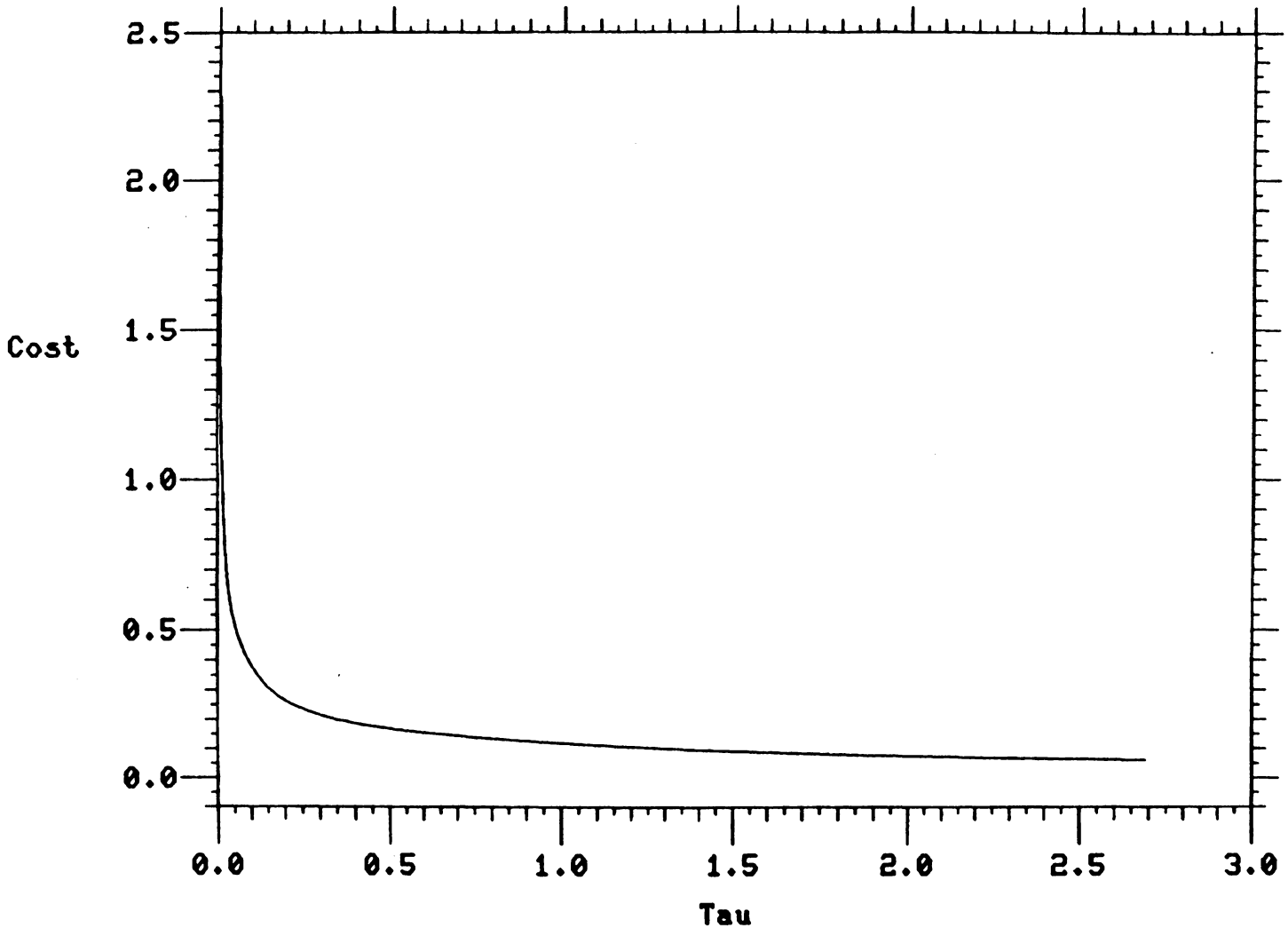


Figure A.18 Cost Trajectory

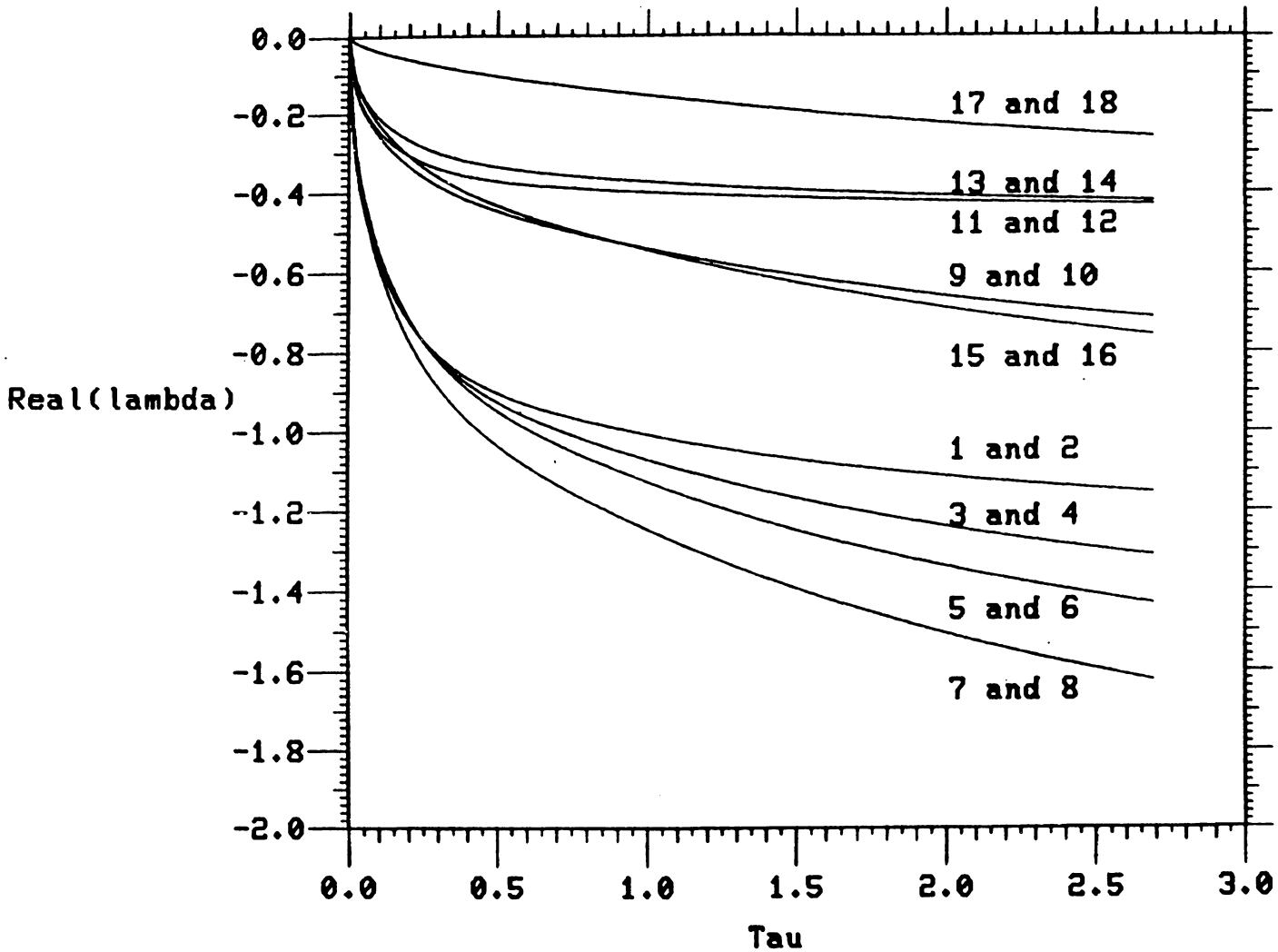


Figure A.19 Real Parts of Eigenvalues

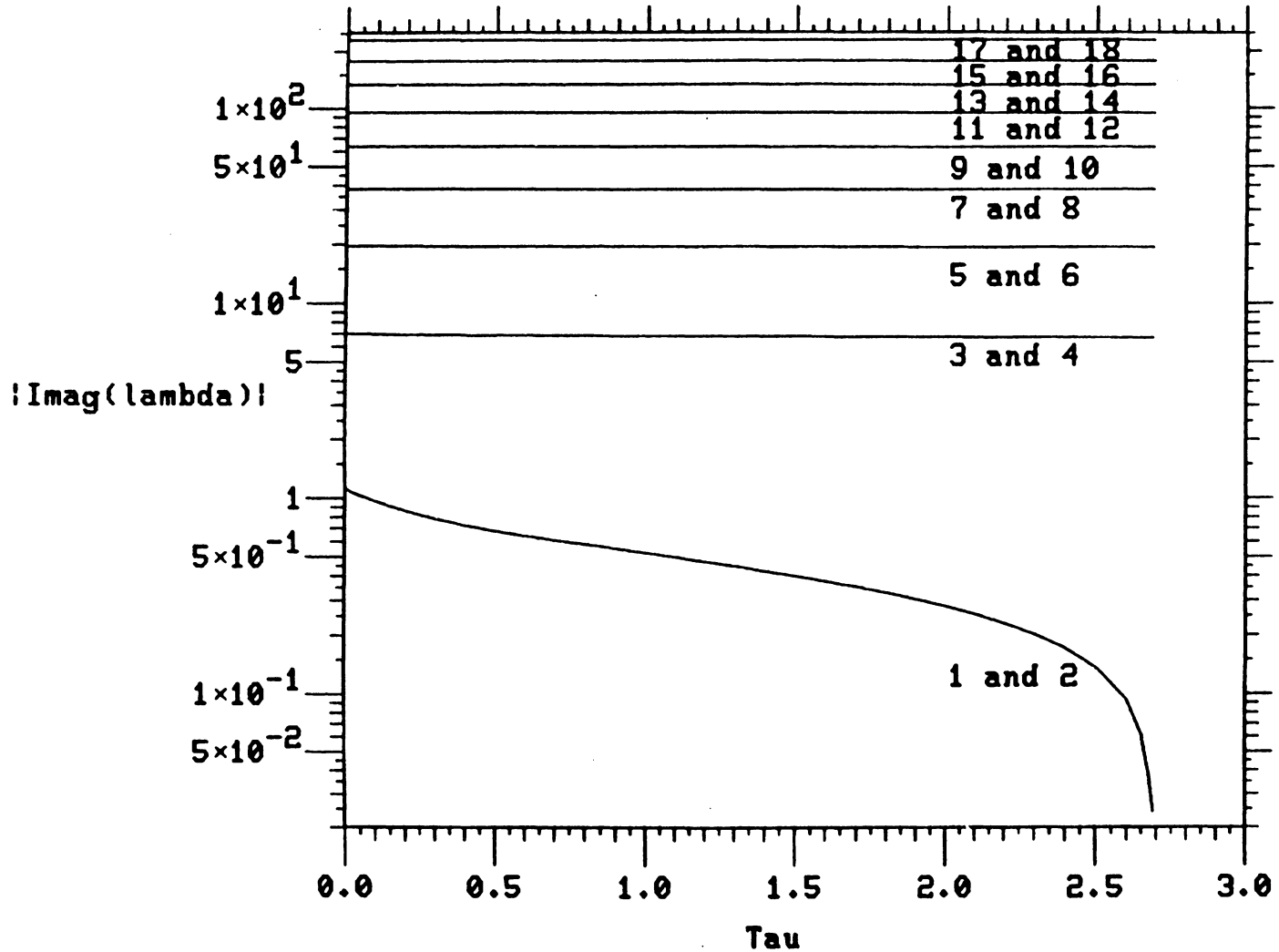


Figure A.20 Imaginary Parts of Eigenvalues

Table A.6
Gains and Parameters

Parameter	Initial Value	Final Value	Parameter	Initial Value	Final Value
tau	0	240			
cost	2.49	3.8×10^{-5}			
g_1	0	-1.27	g_{17}	0	-0.057
g_2	0	-0.78	g_{18}	0	-0.3
g_3	0	-0.02	g_{19}	0	-0.096
g_4	0	-0.76	g_{20}	0	0.084
g_5	0	-0.21	g_{21}	0	-0.5
g_6	0	4.3×10^{-3}	g_{22}	0	0.18
g_7	0	-0.65	g_{23}	0	-0.069
g_8	0	0.042	g_{24}	0	0.022
g_9	0	-0.15	g_{25}	0	-0.36
g_{10}	0	0.0255	g_{26}	0	0.010
g_{11}	0	-0.49	g_{27}	0	-0.91
g_{12}	0	0.034	g_{28}	0	-0.079
g_{13}	0	-0.89	g_{29}	0	-0.12
g_{14}	0	-0.087	g_{30}	0	0.21
g_{15}	0	-0.13	g_{31}	0	-0.37
g_{16}	0	0.11	g_{32}	0	0.31

Table A.7

Eigenvalue Initial and Final States

Eigenvalue	Initial Value rad./sec.		Final Value rad./sec.	
λ_1	0	+ i 1.11	-1.11	+ i 4.83
λ_2	0	- i 1.11	-1.11	- i 4.83
λ_3	0	+ i 6.97	-2.0	+ i 4.84
λ_4	0	- i 6.97	-2.0	- i 4.84
λ_5	0	+ i 19.5	-1.56	+ i 19
λ_6	0	- i 19.5	-1.56	- i 19
λ_7	0	+ i 38.5	-1.09	+ i 38
λ_8	0	- i 38.5	-1.09	- i 38
λ_9	0	+ i 63.3	-0.93	+ i 63
λ_{10}	0	- i 63.3	-0.93	- i 63
λ_{11}	0	+ i 94.7	-1.02	+ i 94
λ_{12}	0	- i 94.7	-1.02	- i 94
λ_{13}	0	+ i 132	-0.64	+ i 132
λ_{14}	0	- i 132	-0.64	- i 132
λ_{15}	0	+ i 177	-0.49	+ i 177
λ_{16}	0	- i 177	-0.49	- i 177
λ_{17}	0	+ i 228	-0.49	+ i 228
λ_{18}	0	- i 228	-0.49	- i 228

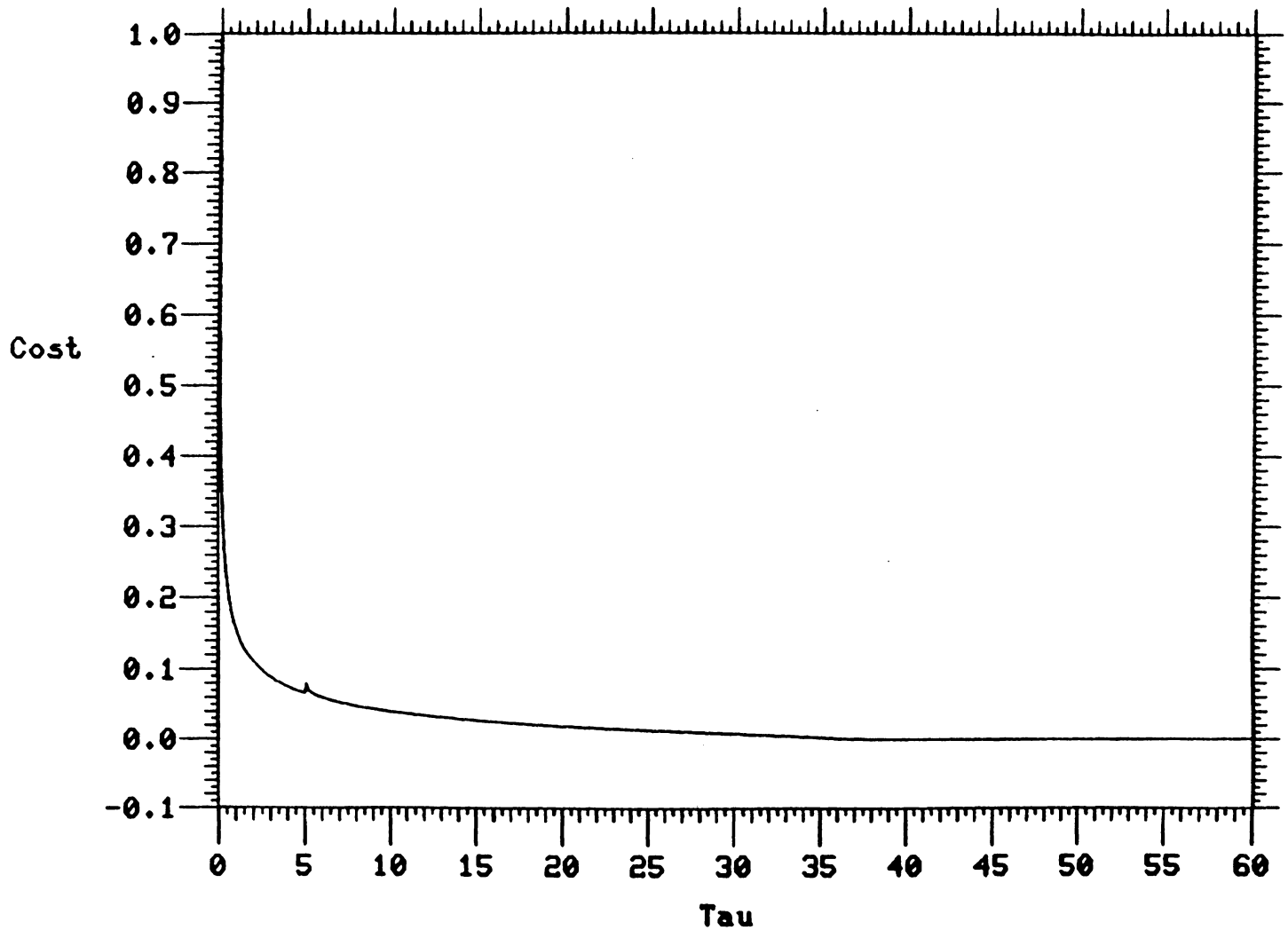


Figure A.21 Cost Trajectory

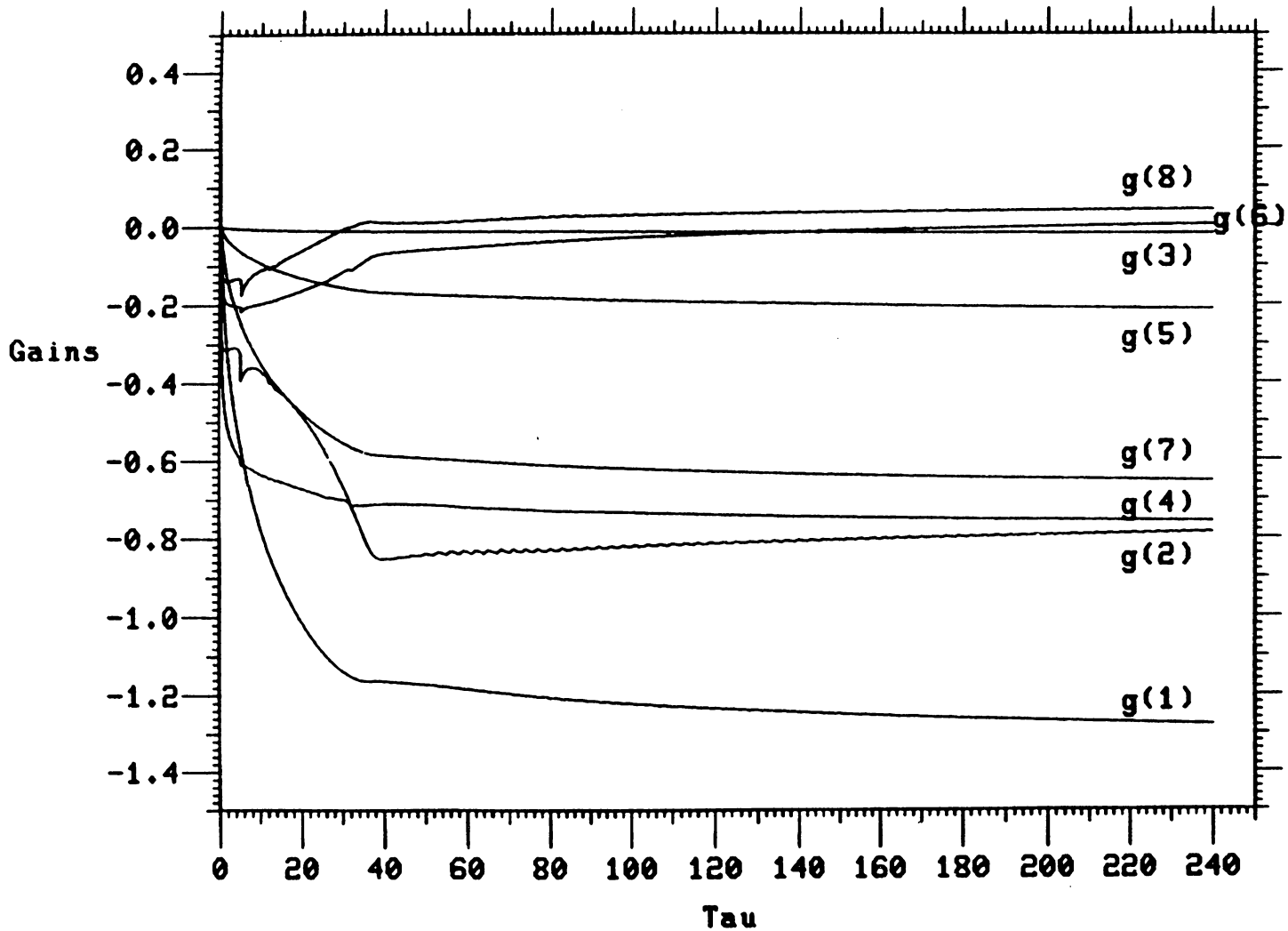


Figure A.22 Gain Plots

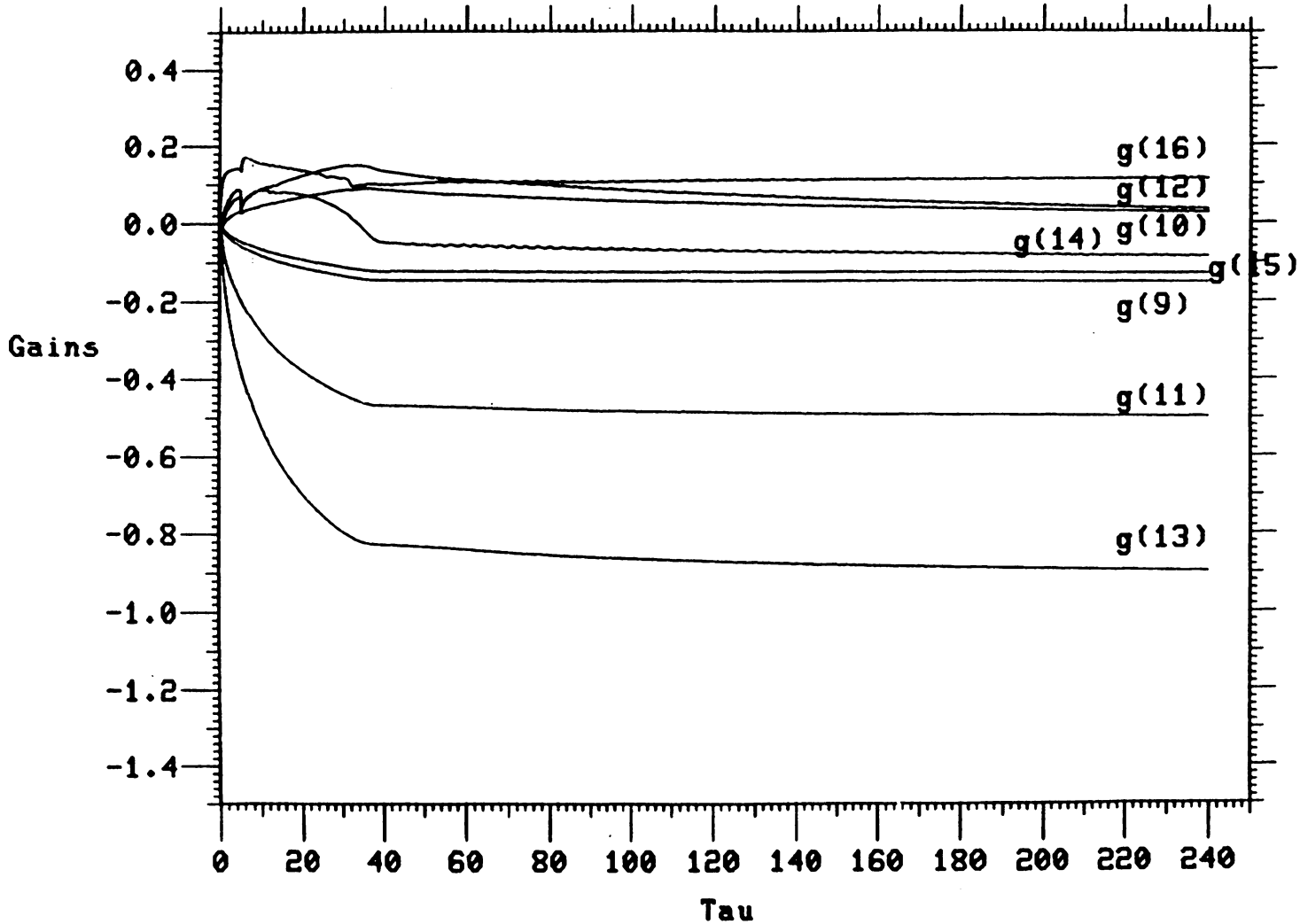


Figure A.23 Gain Plots

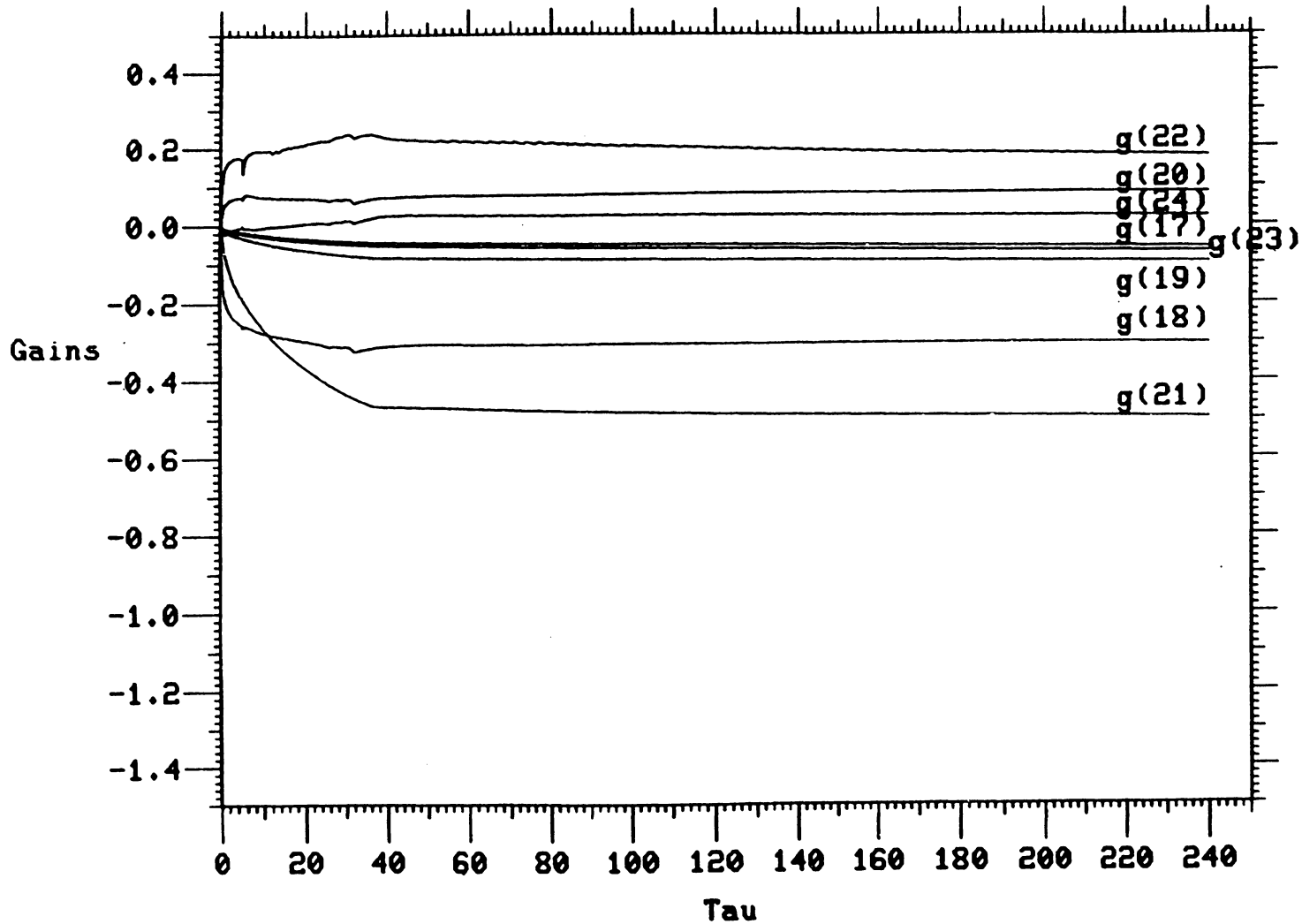


Figure A.24 Gain Plots

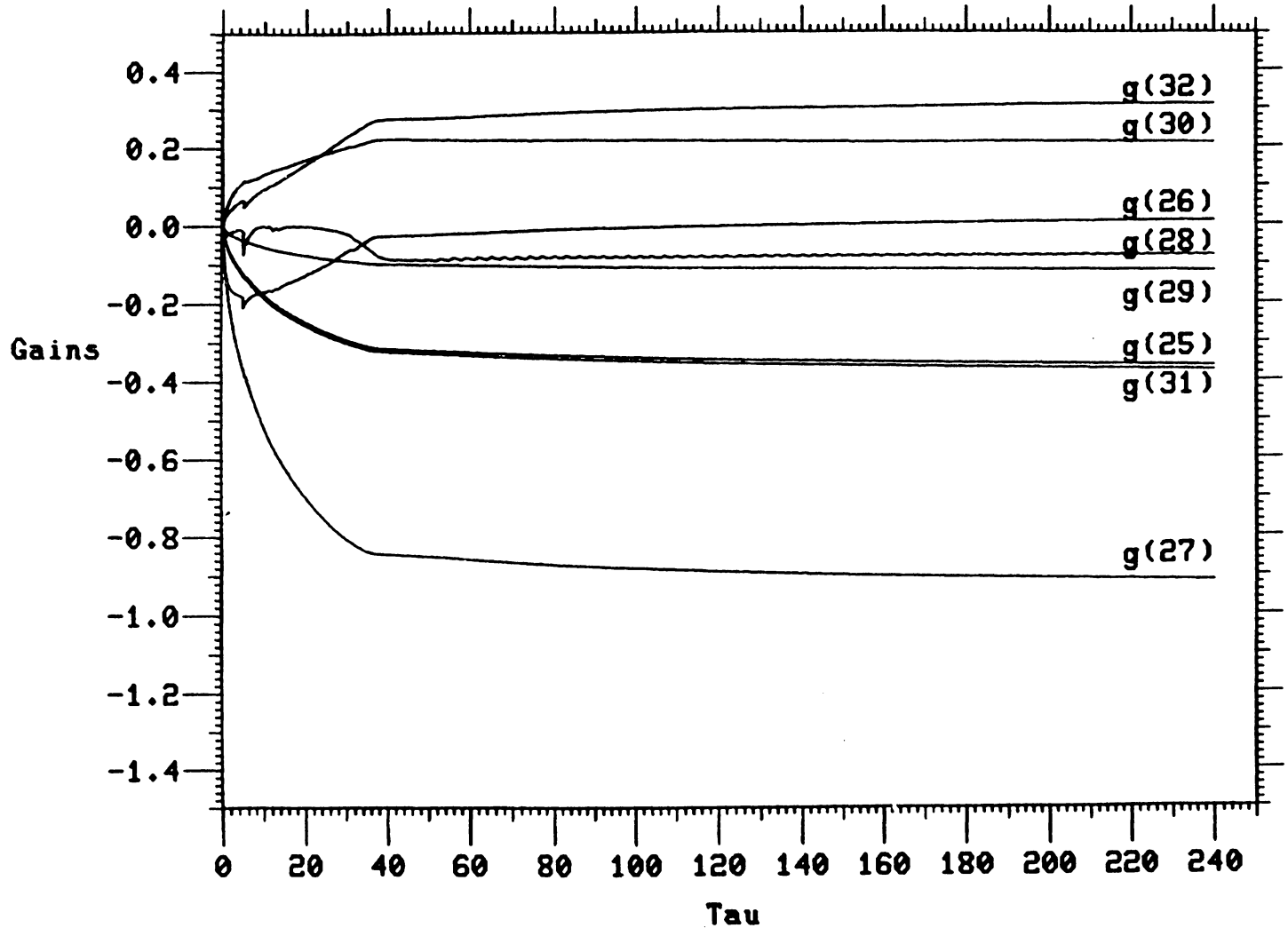


Figure A.25 Gain Plots

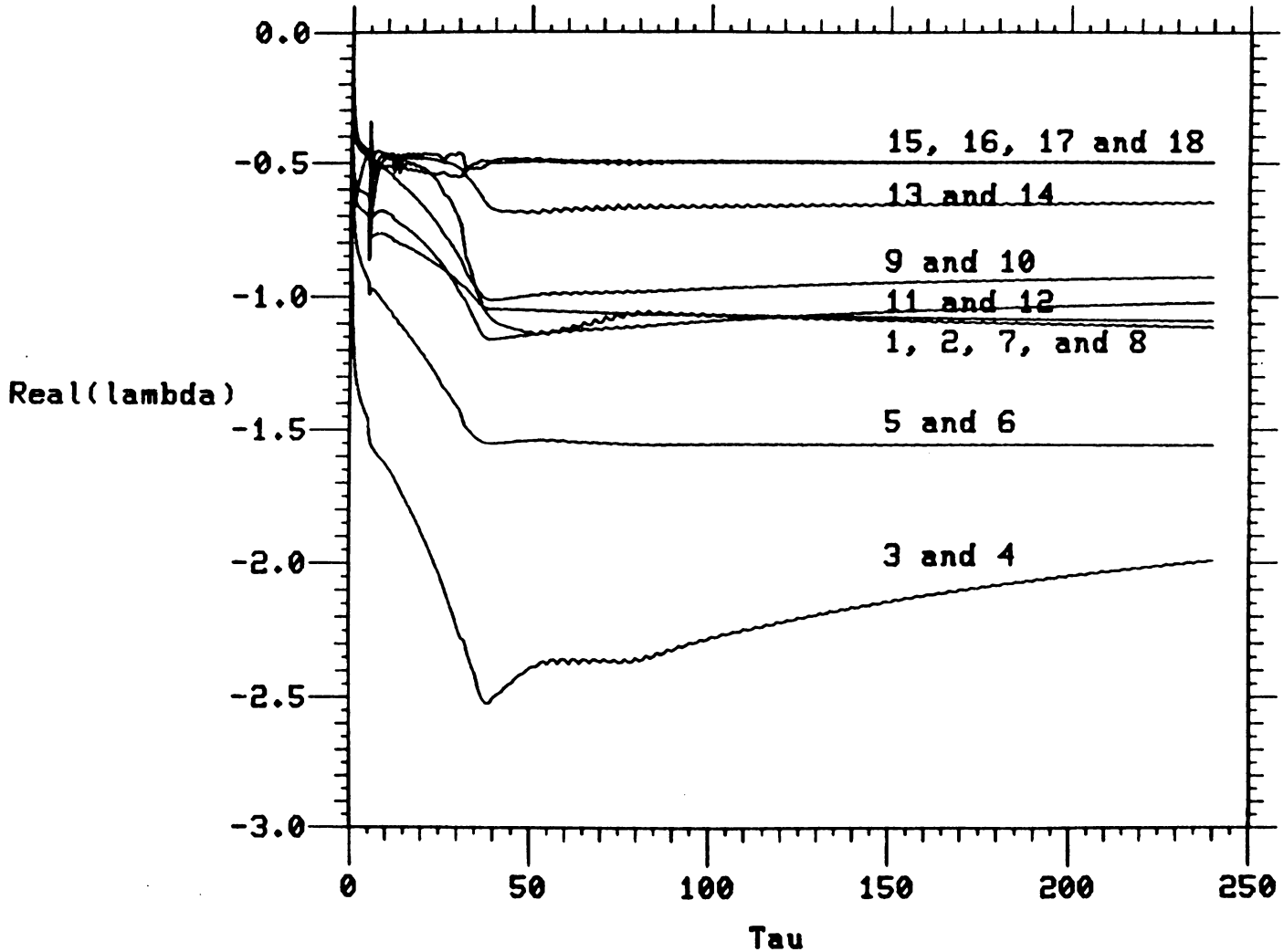


Figure A.26 Real Parts of Eigenvalues

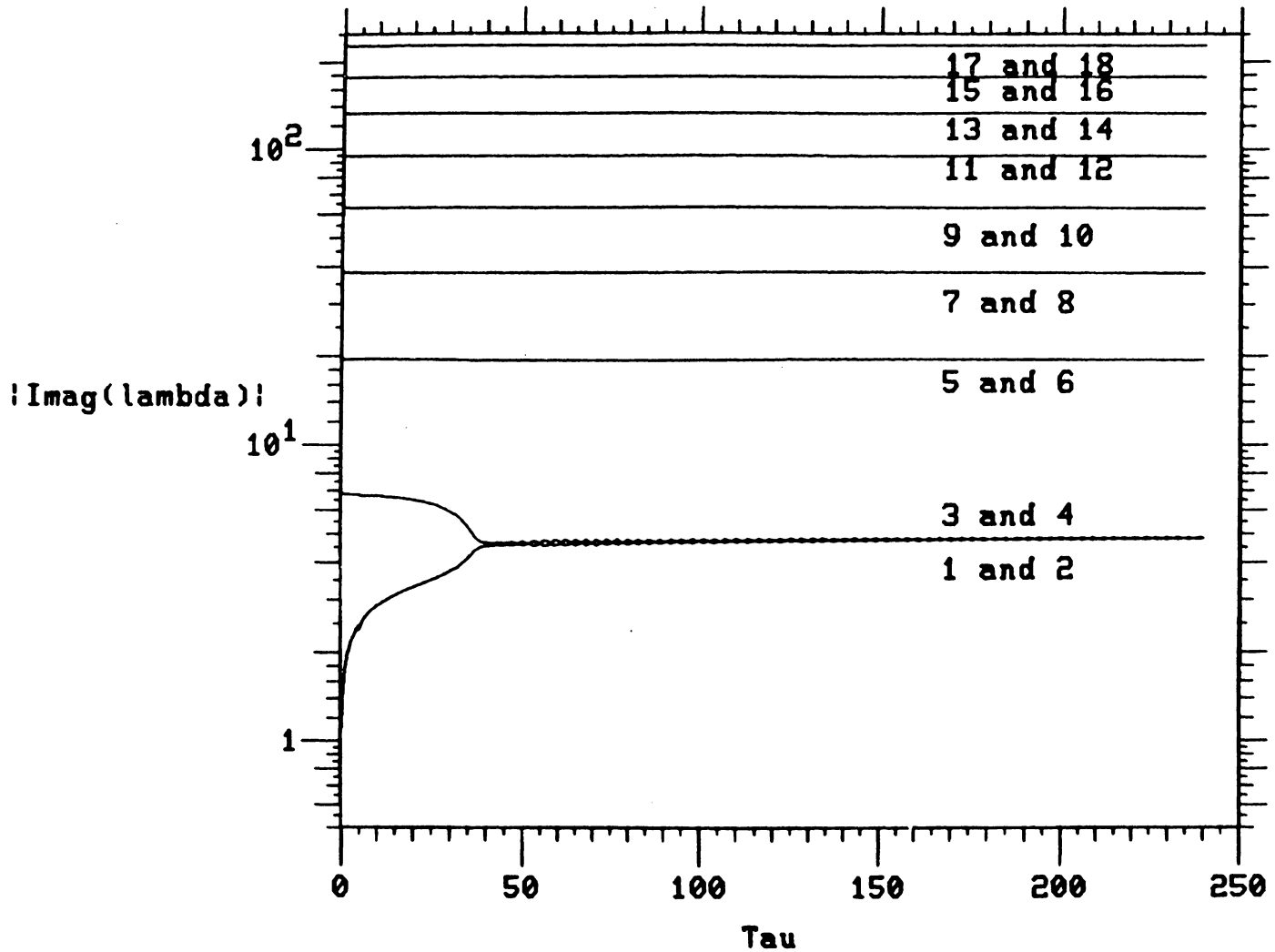


Figure A.27 Imaginary Parts of Eigenvalues

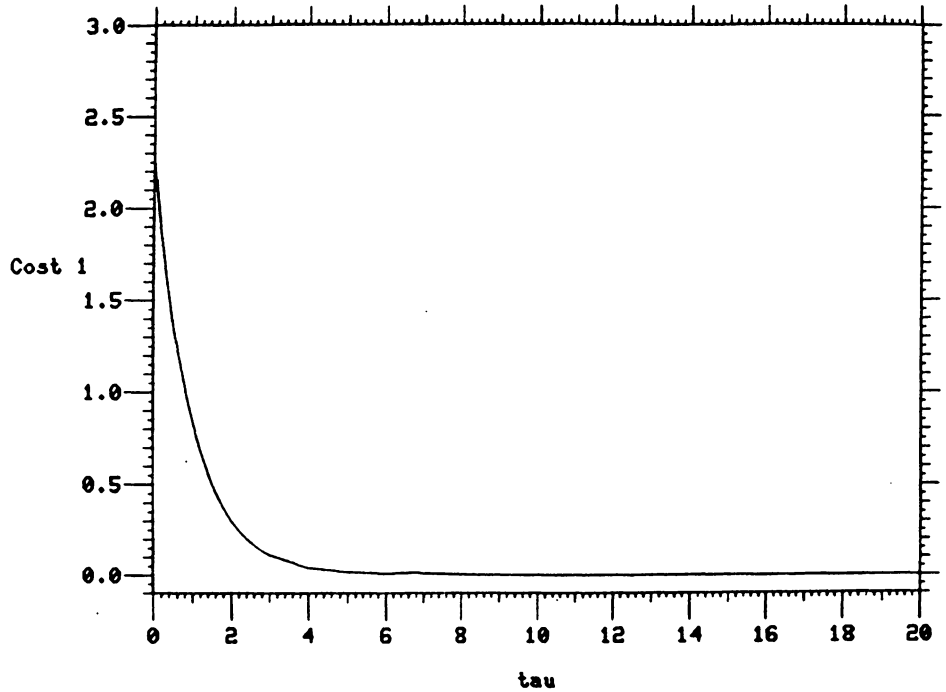
Table A.8
Gains and Parameters

Parameter	Initial Value	Final Value	Parameter	Initial Value	Final Value
tau	0				
cost1	2.25	$2x10^{-7}$	cost2	0.015	$5x10^{-7}$
g_1	0	-1.4	g_{17}	0	-0.057
g_2	0	-0.78	g_{18}	0	-0.28
g_3	0	-0.019	g_{19}	0	-0.097
g_4	0	-0.62	g_{20}	0	-0.0027
g_5	0	-0.21	g_{21}	0	-0.515
g_6	0	-0.015	g_{22}	0	0.11
g_7	0	-0.68	g_{23}	0	-0.68
g_8	0	0.049	g_{24}	0	-0.063
g_9	0	-0.157	g_{25}	0	-0.37
g_{10}	0	0.033	g_{26}	0	0.059
g_{11}	0	-0.52	g_{27}	0	-0.96
g_{12}	0	0.0217	g_{28}	0	-0.038
g_{13}	0	-0.95	g_{29}	0	-0.12
g_{14}	0	-0.16	g_{30}	0	0.13
g_{15}	0	-0.13	g_{31}	0	-0.38
g_{16}	0	0.16	g_{32}	0	0.27

Table A.9

Eigenvalue Initial and Final States

Eigenvalue	Initial Value rad./sec.		Final Value rad./sec.	
λ_1	0	+ i 1.11	-1.35	+ i 4.8
λ_2	0	- i 1.11	-1.35	- i 4.8
λ_3	0	+ i 6.97	-1.8	+ i 4.9
λ_4	0	- i 6.97	-1.8	- i 4.9
λ_5	0	+ i 19.5	-1.3	+ i 19
λ_6	0	- i 19.5	-1.3	- i 19
λ_7	0	+ i 38.5	-1.2	+ i 38
λ_8	0	- i 38.5	-1.2	- i 38
λ_9	0	+ i 63.3	-0.70	+ i 63
λ_{10}	0	- i 63.3	-0.70	- i 63
λ_{11}	0	+ i 94.7	-0.92	+ i 94
λ_{12}	0	- i 94.7	-0.92	- i 94
λ_{13}	0	+ i 132	-0.50	+ i 132
λ_{14}	0	- i 132	-0.50	- i 132
λ_{15}	0	+ i 177	-0.50	+ i 177
λ_{16}	0	- i 177	-0.50	- i 177
λ_{17}	0	+ i 228	-0.50	+ i 228
λ_{18}	0	- i 228	-0.50	- i 228



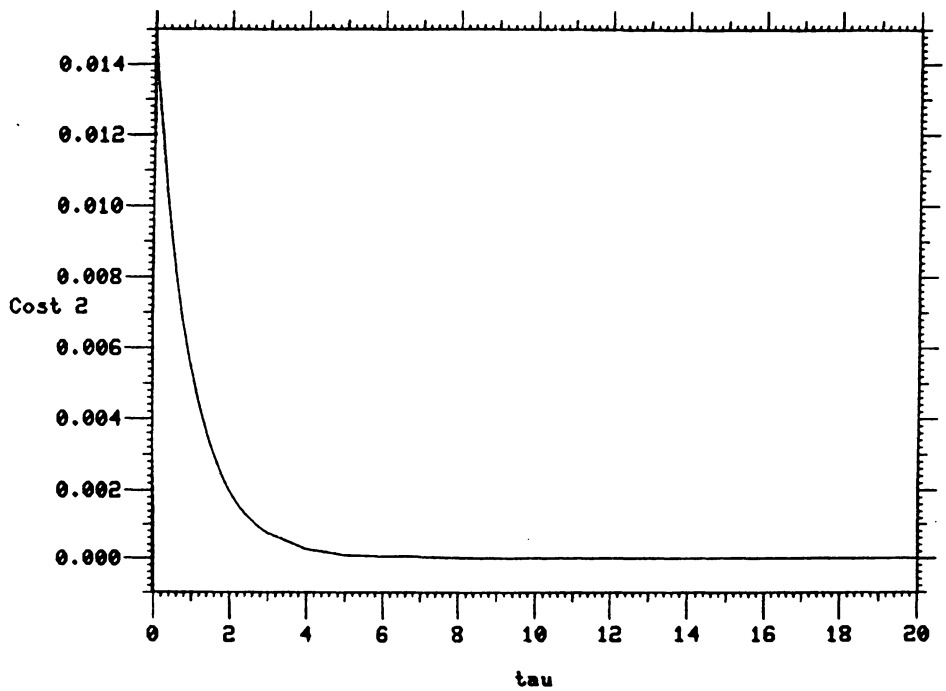


Figure A.28 Cost Trajectories

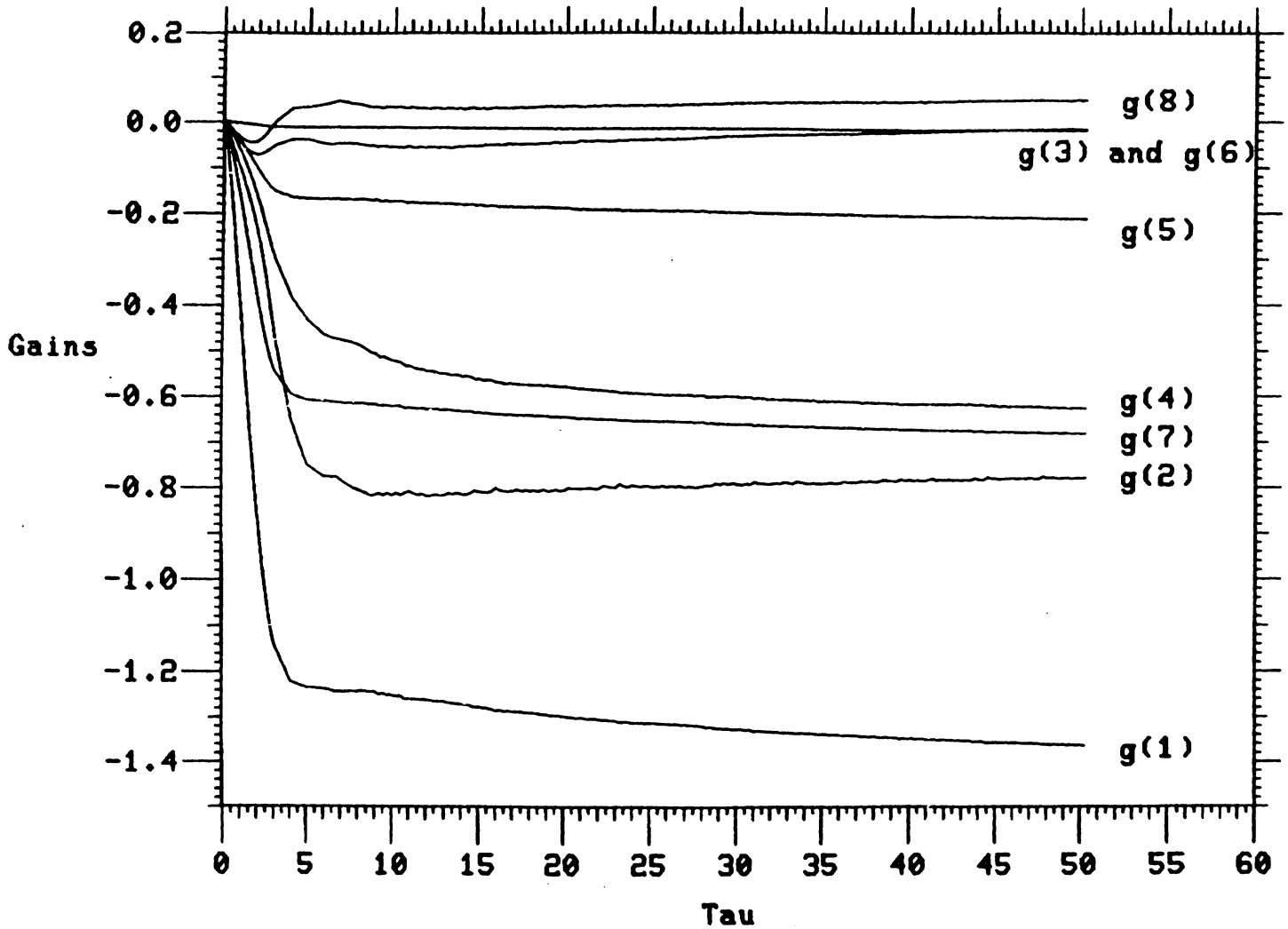


Figure A.29 Gain Plots

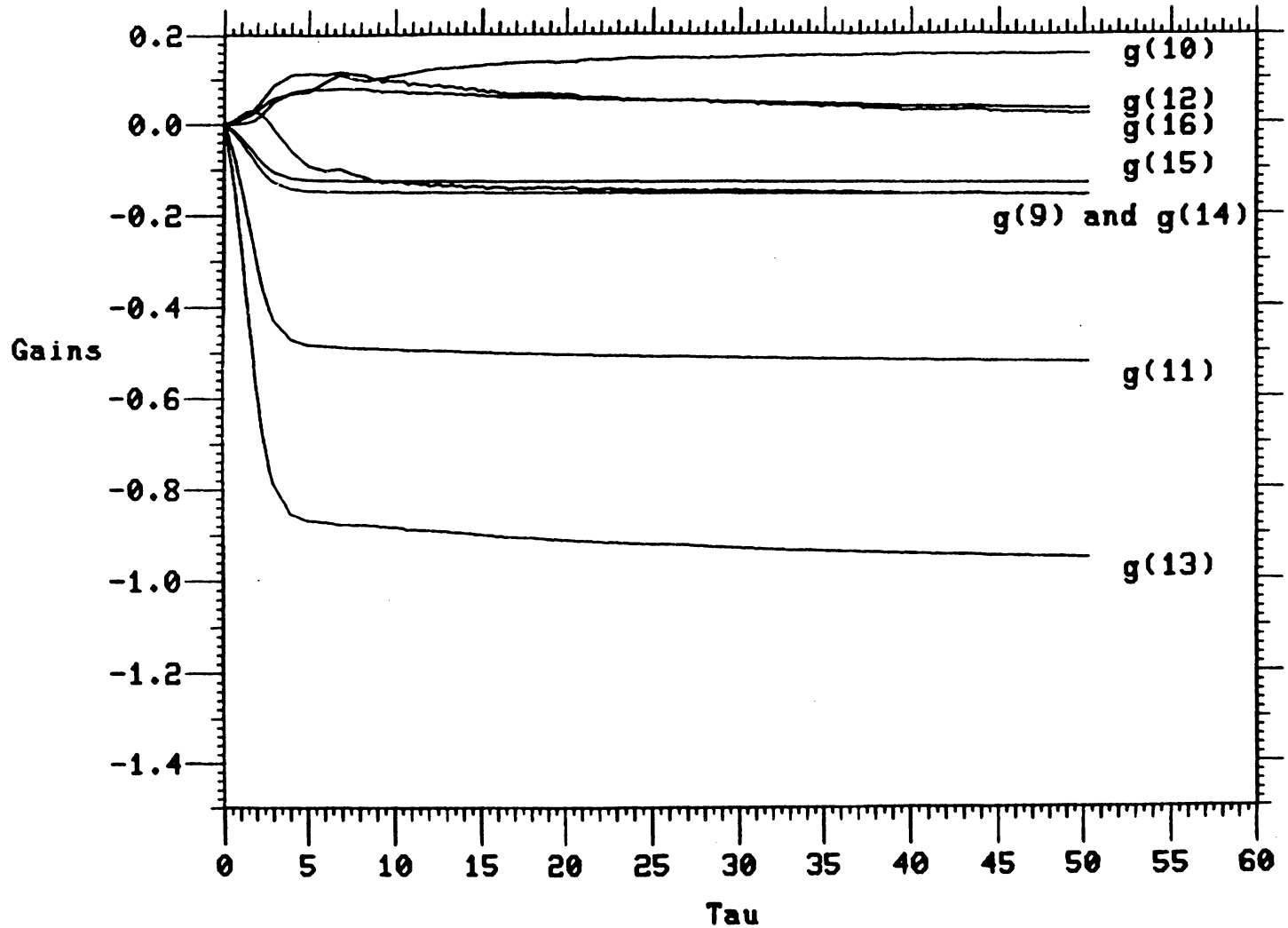


Figure A.30 Gain Plots

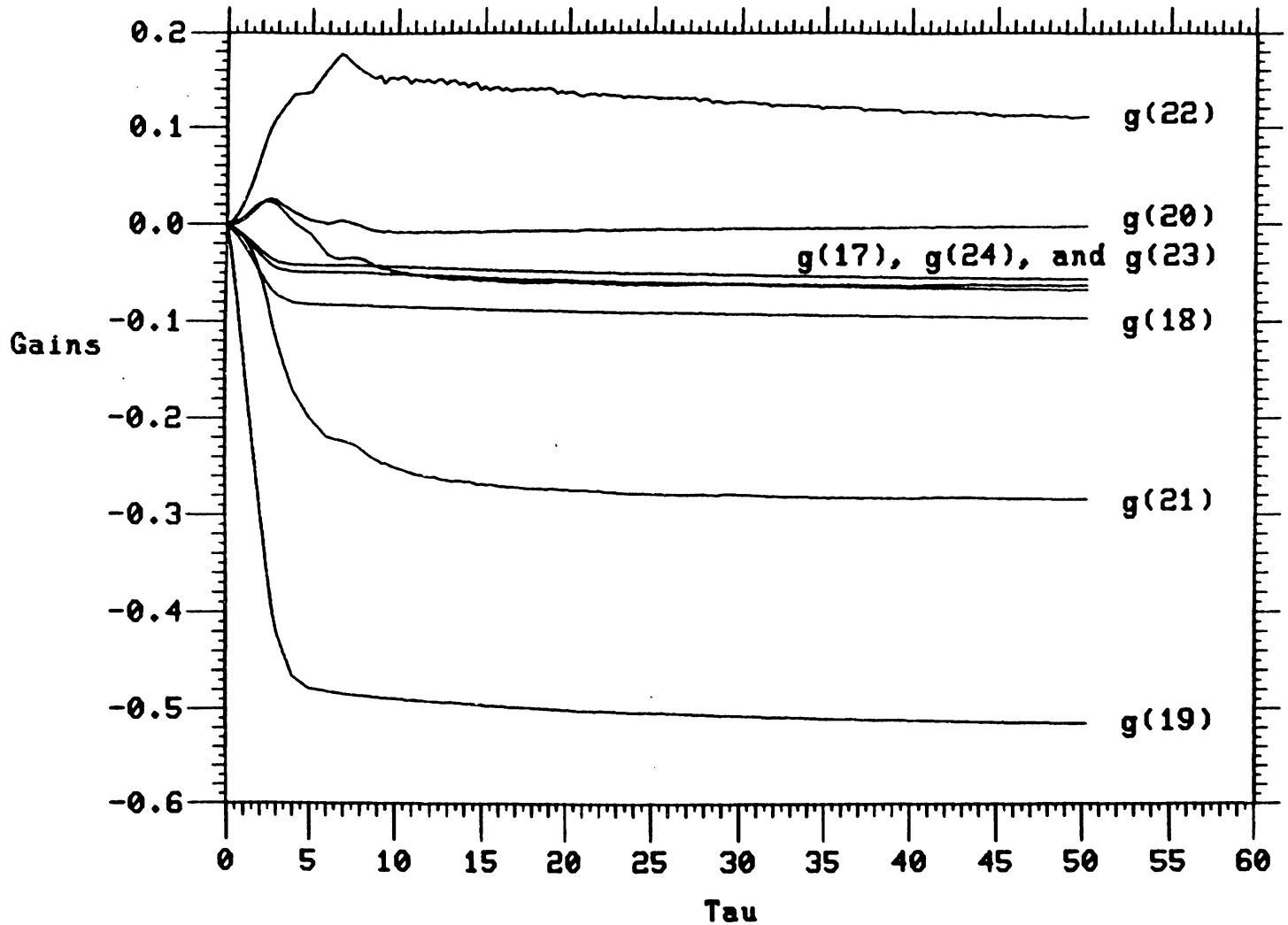


Figure A.31 Gain Plots

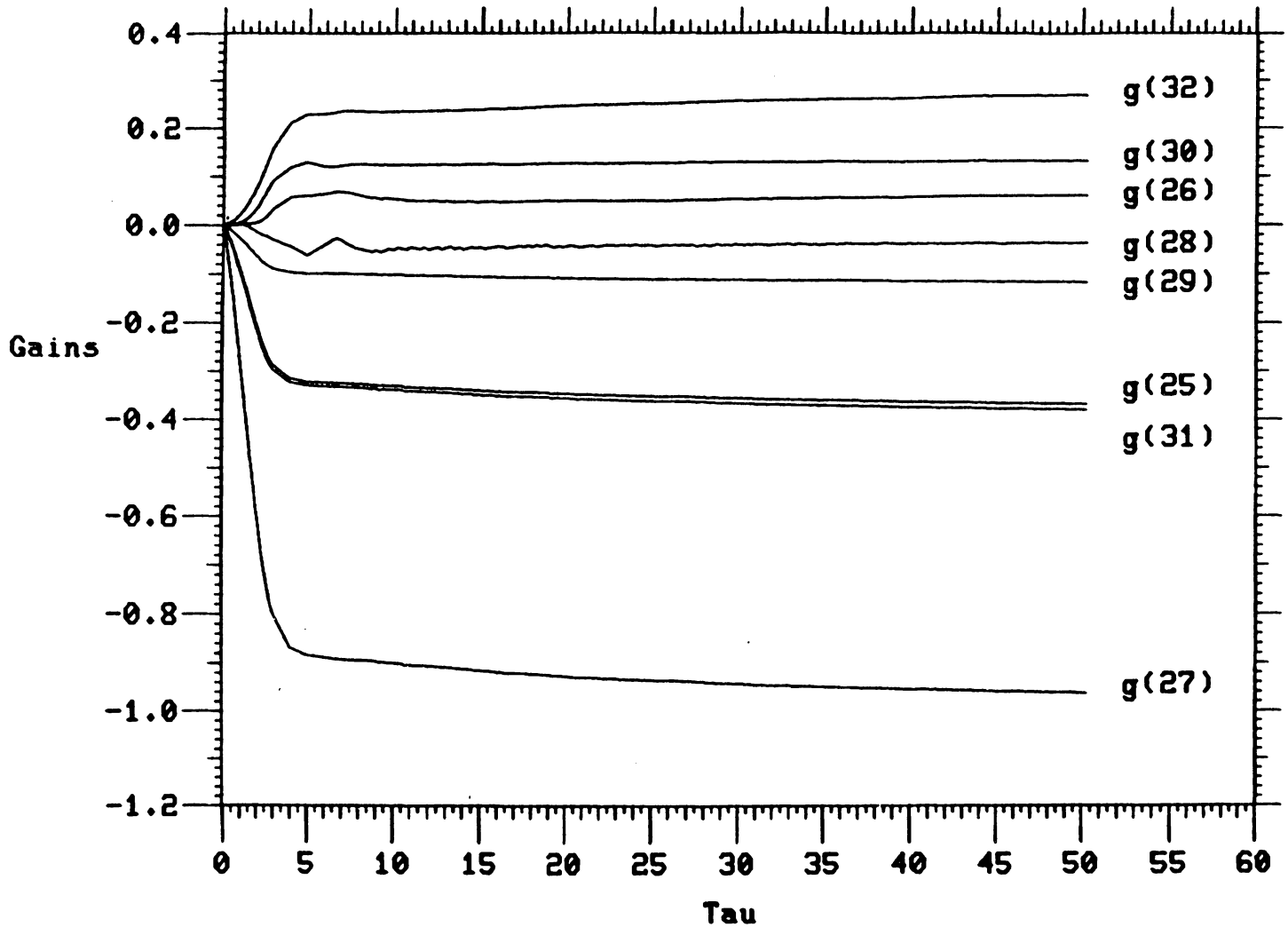


Figure A.32 Gain Plots

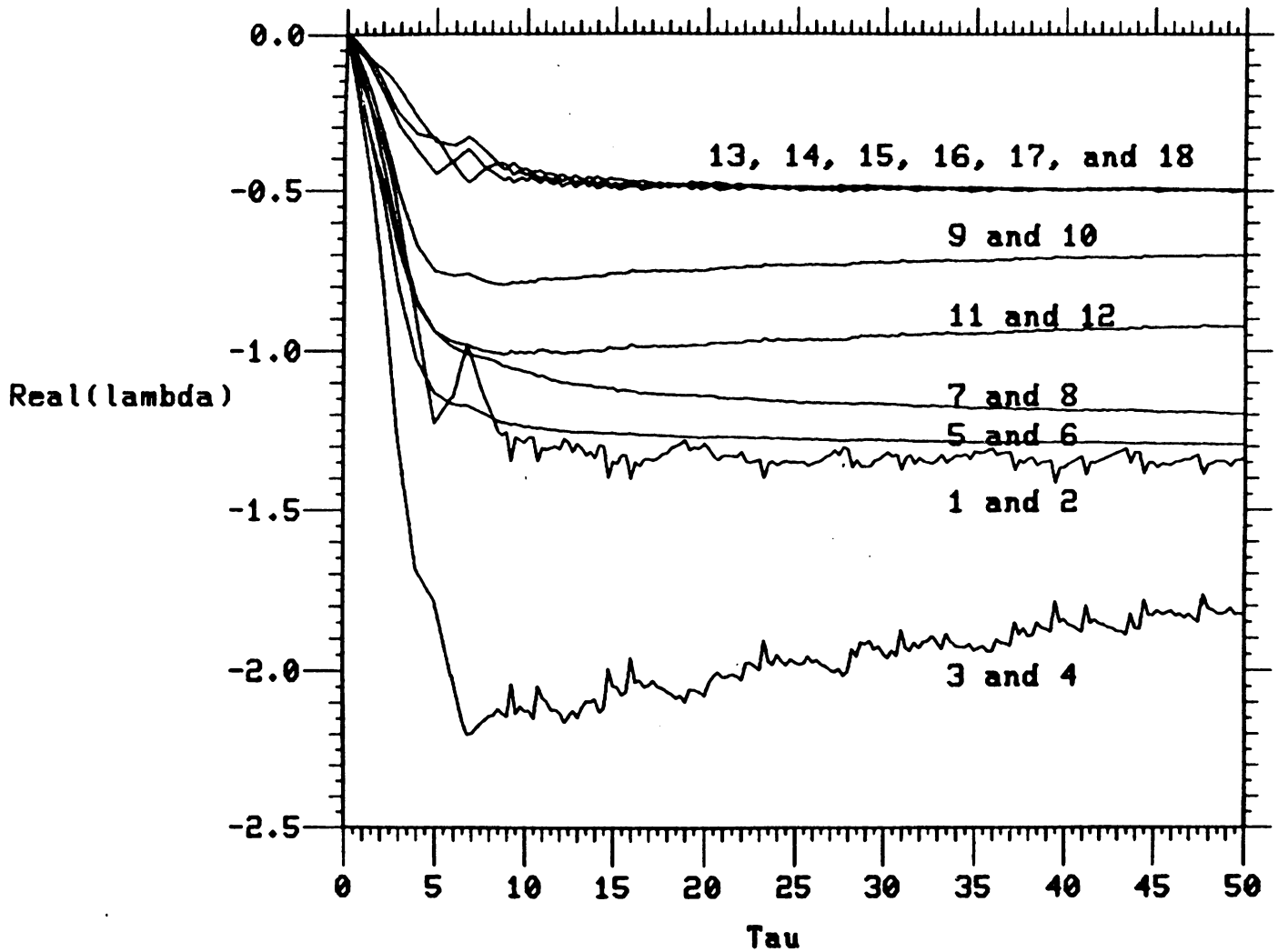


Figure A.33 Real Parts of Eigenvalues

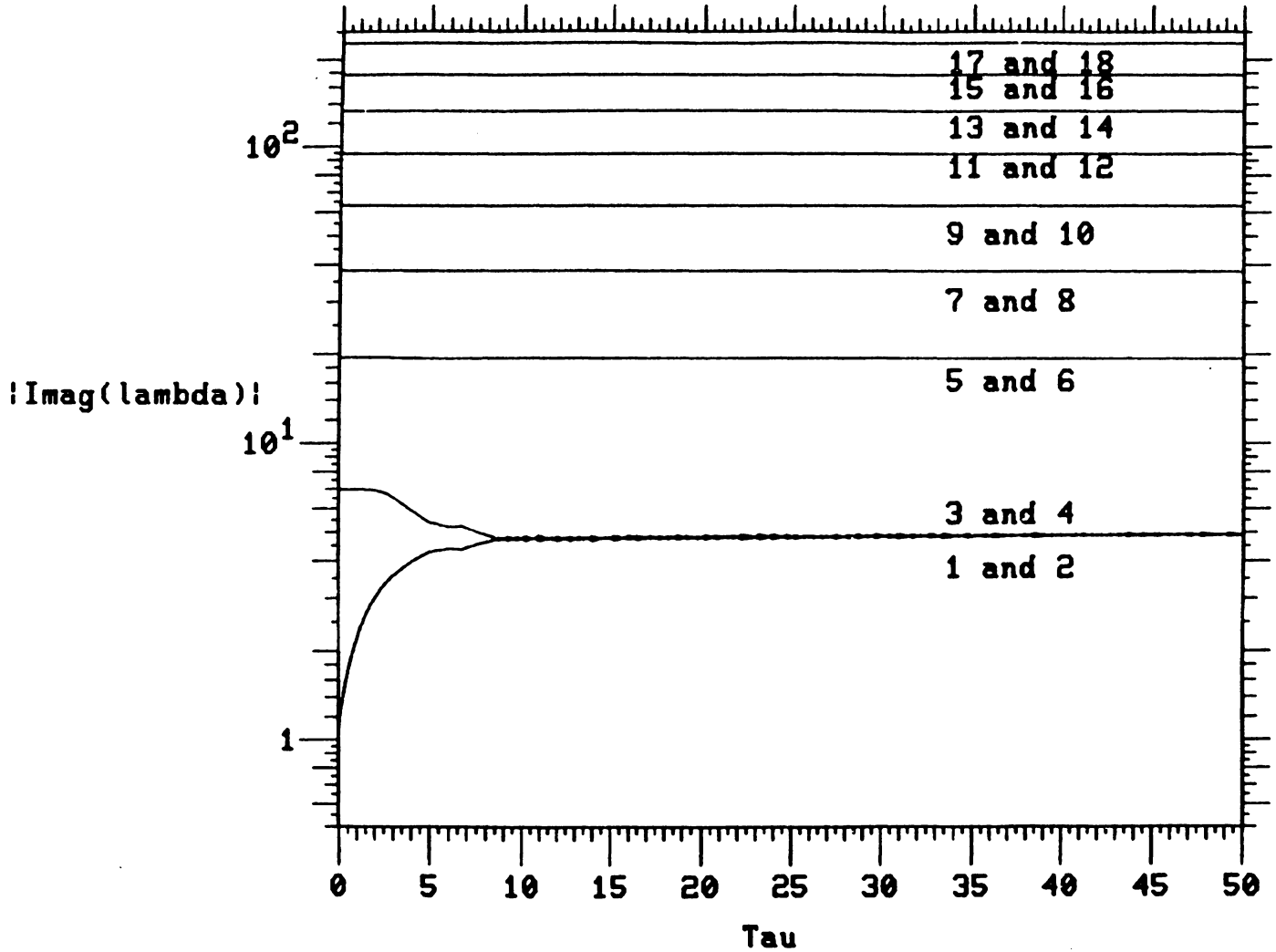


Figure A.34 Imaginary Parts of Eigenvalues

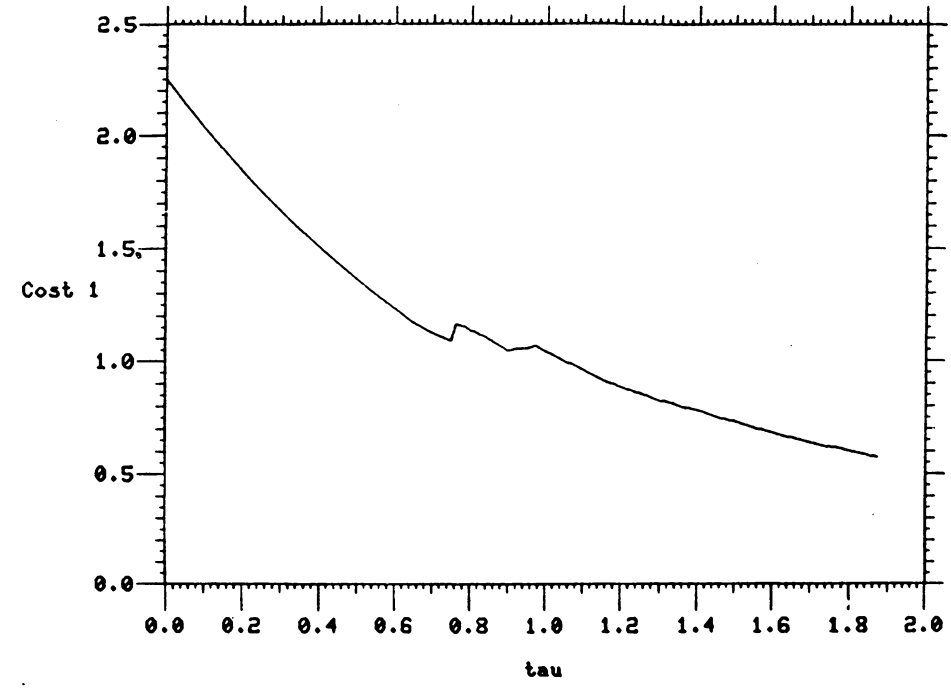
Table A.10
Gains and Parameters

Parameter	Initial Value	Final Value	Parameter	Initial Value	Final Value
tau	0	1.8			
cost1	2.25	0.57	cost2	2.23	0.75
g_1	0	-6.4	g_{17}	0	-0.93
g_2	0	-1.7	g_{18}	0	0.24
g_3	0	-0.48	g_{19}	0	-0.61
g_4	0	0.068	g_{20}	0	-0.12
g_5	0	-2.5	g_{21}	0	-0.25
g_6	0	0.089	g_{22}	0	-0.39
g_7	0	-0.30	g_{23}	0	-0.12
g_8	0	-0.016	g_{24}	0	0.92
g_9	0	0.21	g_{25}	0	-2.0
g_{10}	0	-1.1	g_{26}	0	-0.55
g_{11}	0	-0.56	g_{27}	0	-2.9
g_{12}	0	-0.54	g_{28}	0	0.62
g_{13}	0	-3.1	g_{29}	0	-0.91
g_{14}	0	0.71	g_{30}	0	-0.65
g_{15}	0	0.25	g_{31}	0	2.4
g_{16}	0	0.15	g_{32}	0	0.20

Table A.11

Eigenvalue Initial and Final States

Eigenvalue	Initial Value rad./sec.		Final Value rad./sec.	
λ_1	0	+ i 1.11	-0.36	+ i 6.9
λ_2	0	- i 1.11	-0.36	- i 6.9
λ_3	0	+ i 6.97	-3.7	+ i 12.6
λ_4	0	- i 6.97	-3.7	- i 12.6
λ_5	0	+ i 19.5	-3.3	+ i 12.8
λ_6	0	- i 19.5	-3.3	- i 12.8
λ_7	0	+ i 38.5	-3.5	+ i 36
λ_8	0	- i 38.5	-3.5	- i 36
λ_9	0	+ i 63.3	-1.4	+ i 63
λ_{10}	0	- i 63.3	-1.4	- i 63
λ_{11}	0	+ i 94.7	-0.13	+ i 94
λ_{12}	0	- i 94.7	-0.13	- i 94
λ_{13}	0	+ i 132	-0.31	+ i 132
λ_{14}	0	- i 132	-0.31	- i 132
λ_{15}	0	+ i 177	-0.16	+ i 177
λ_{16}	0	- i 177	-0.16	- i 177
λ_{17}	0	+ i 228	0.075	+ i 228
λ_{18}	0	- i 228	0.075	- i 228



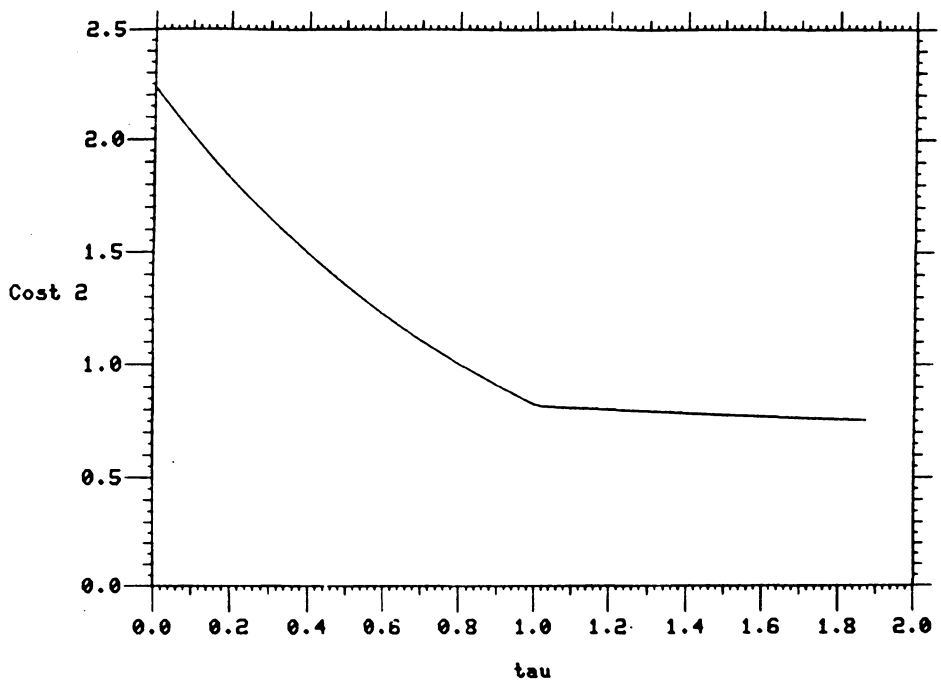


Figure A.35 Cost Trajectories

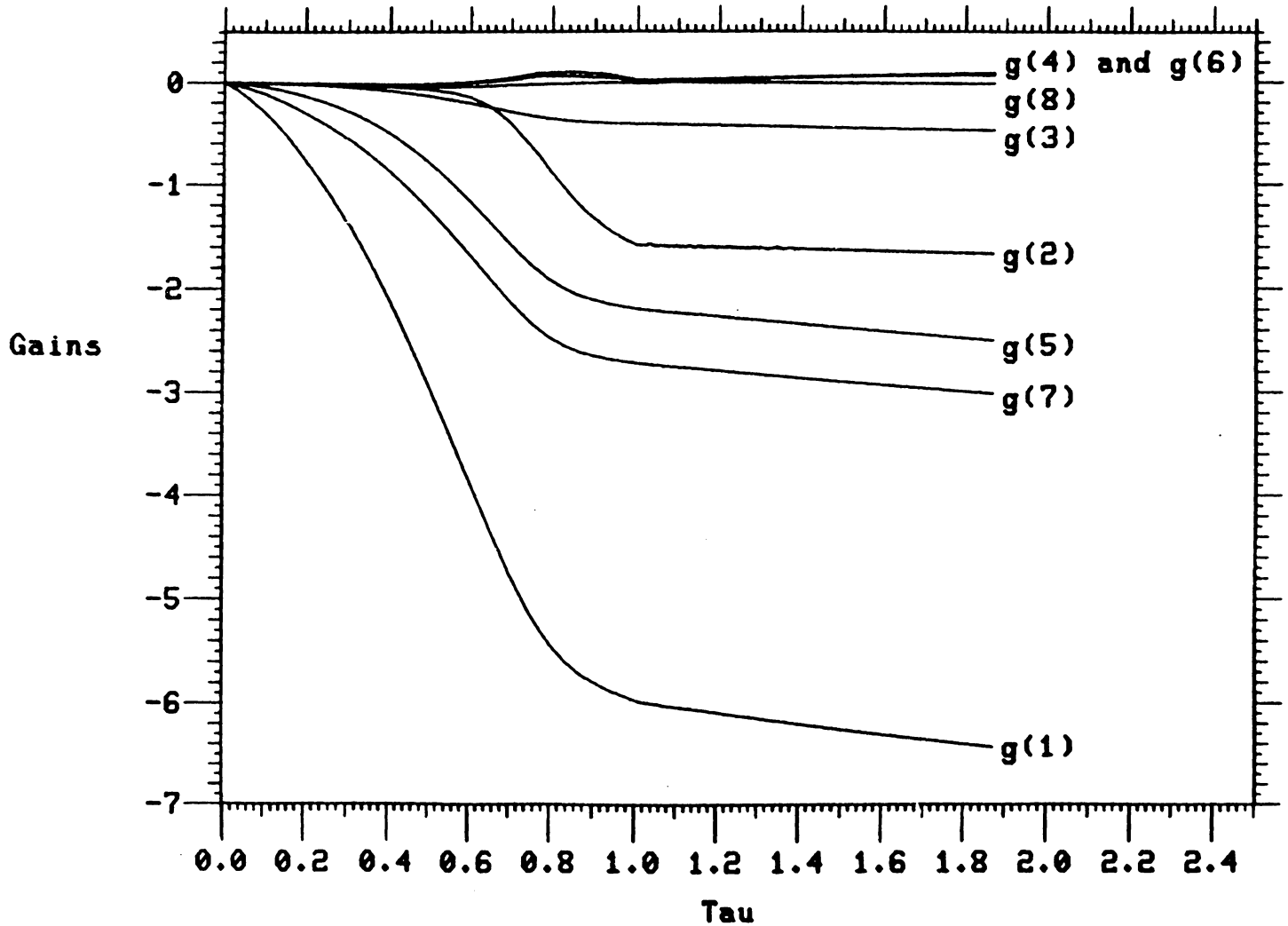


Figure A.36 Gain Plots

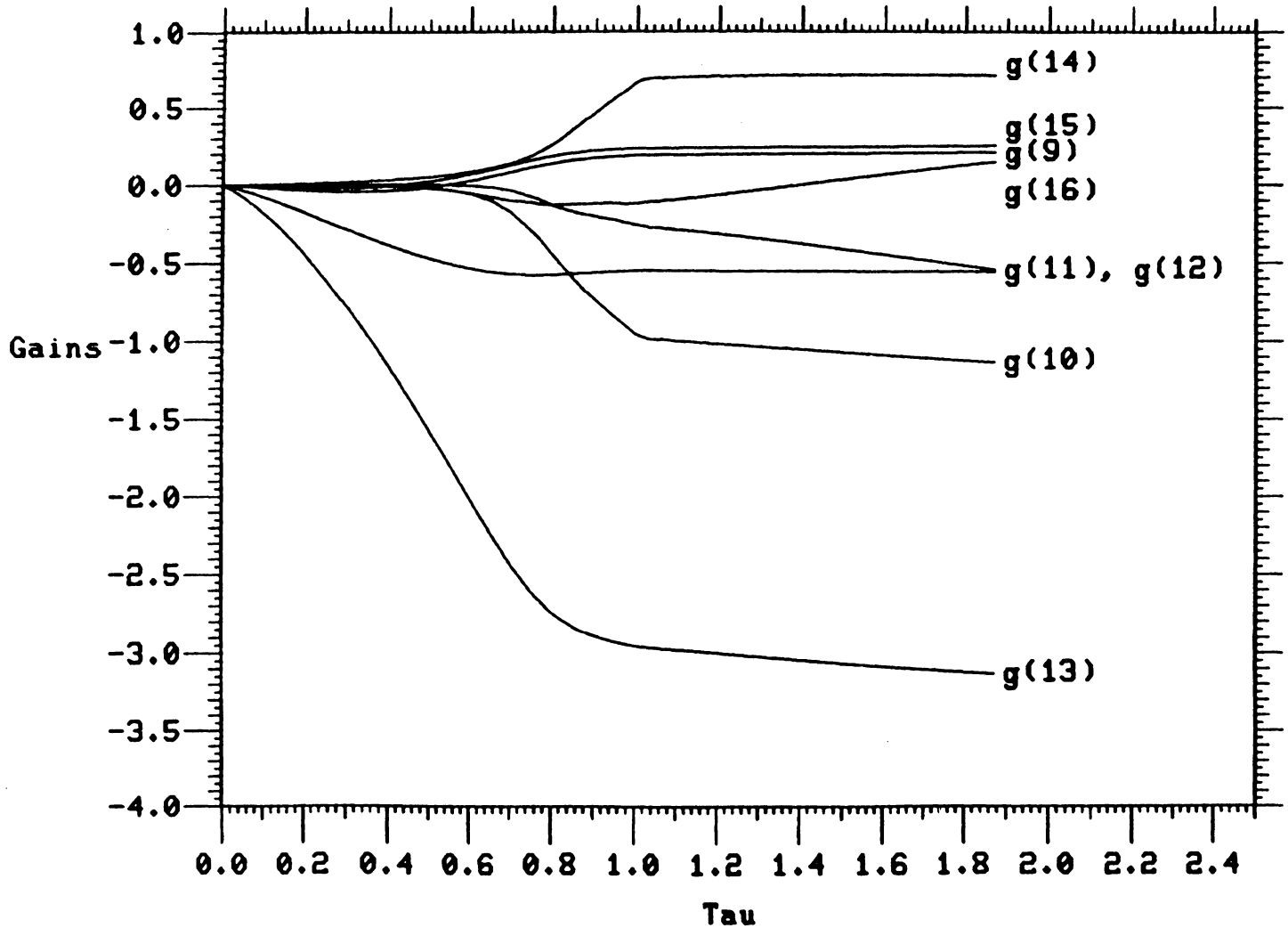


Figure A.37 Gain Plots

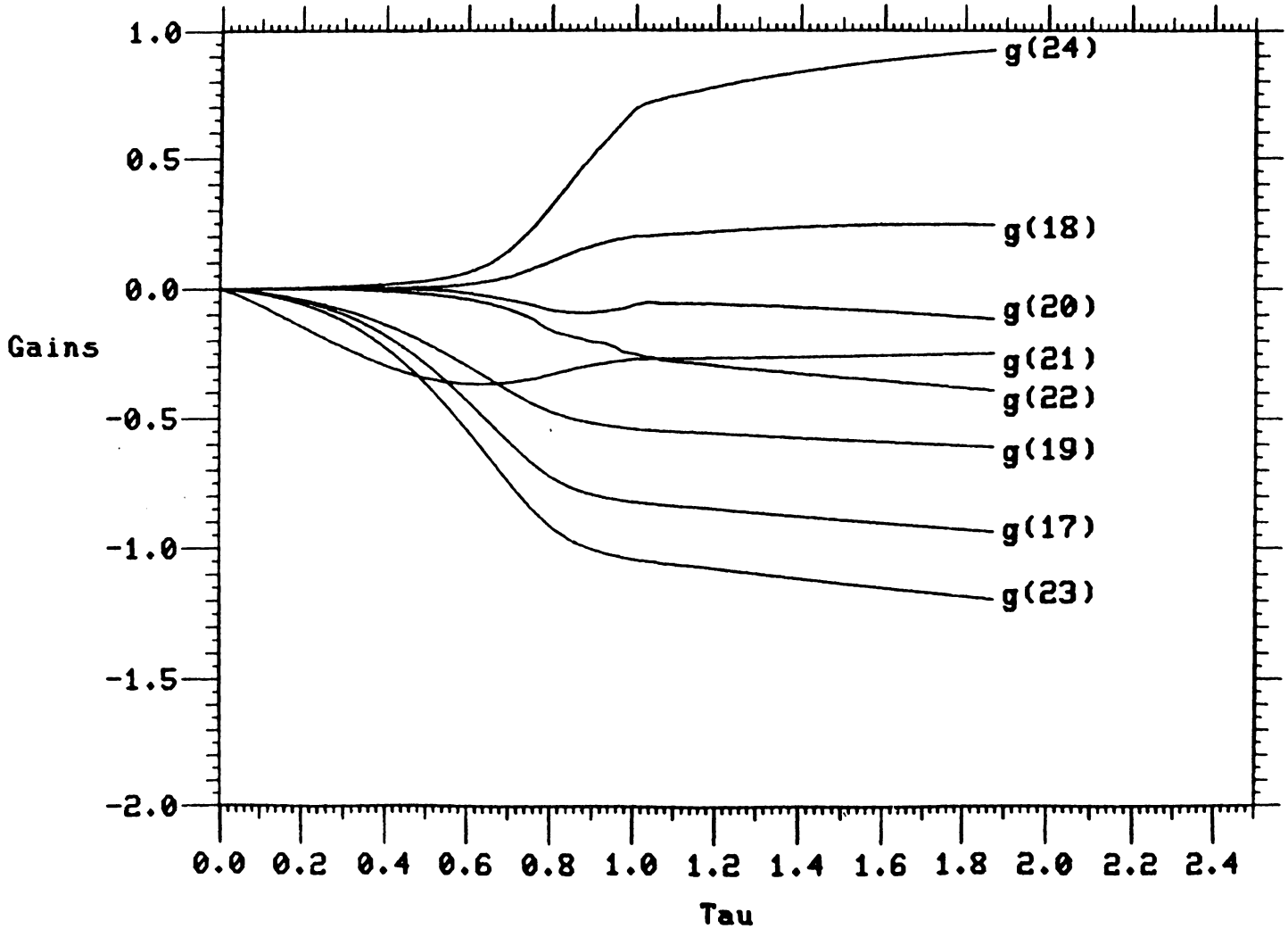


Figure A.38 Gain Plots

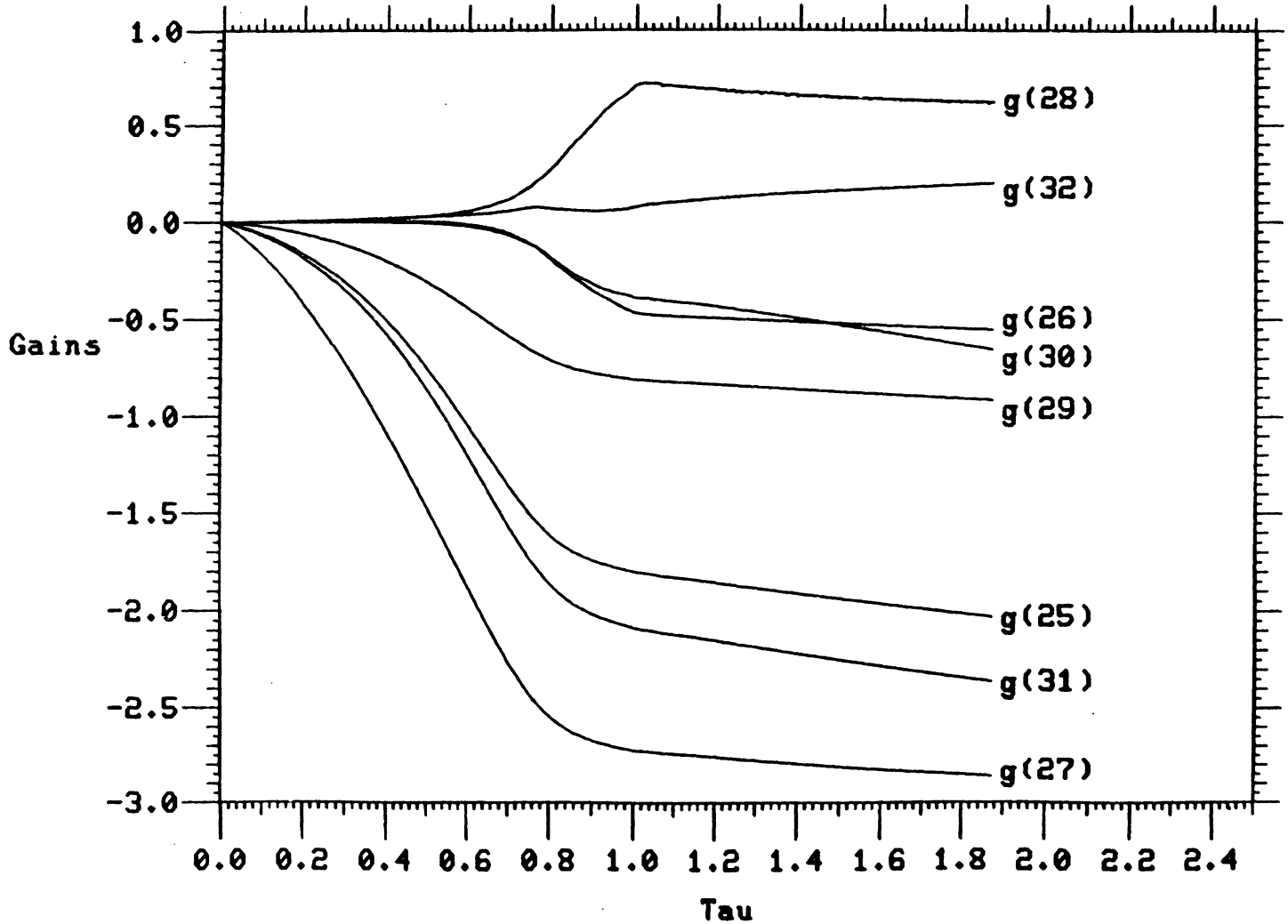


Figure A.39 Gain Plots

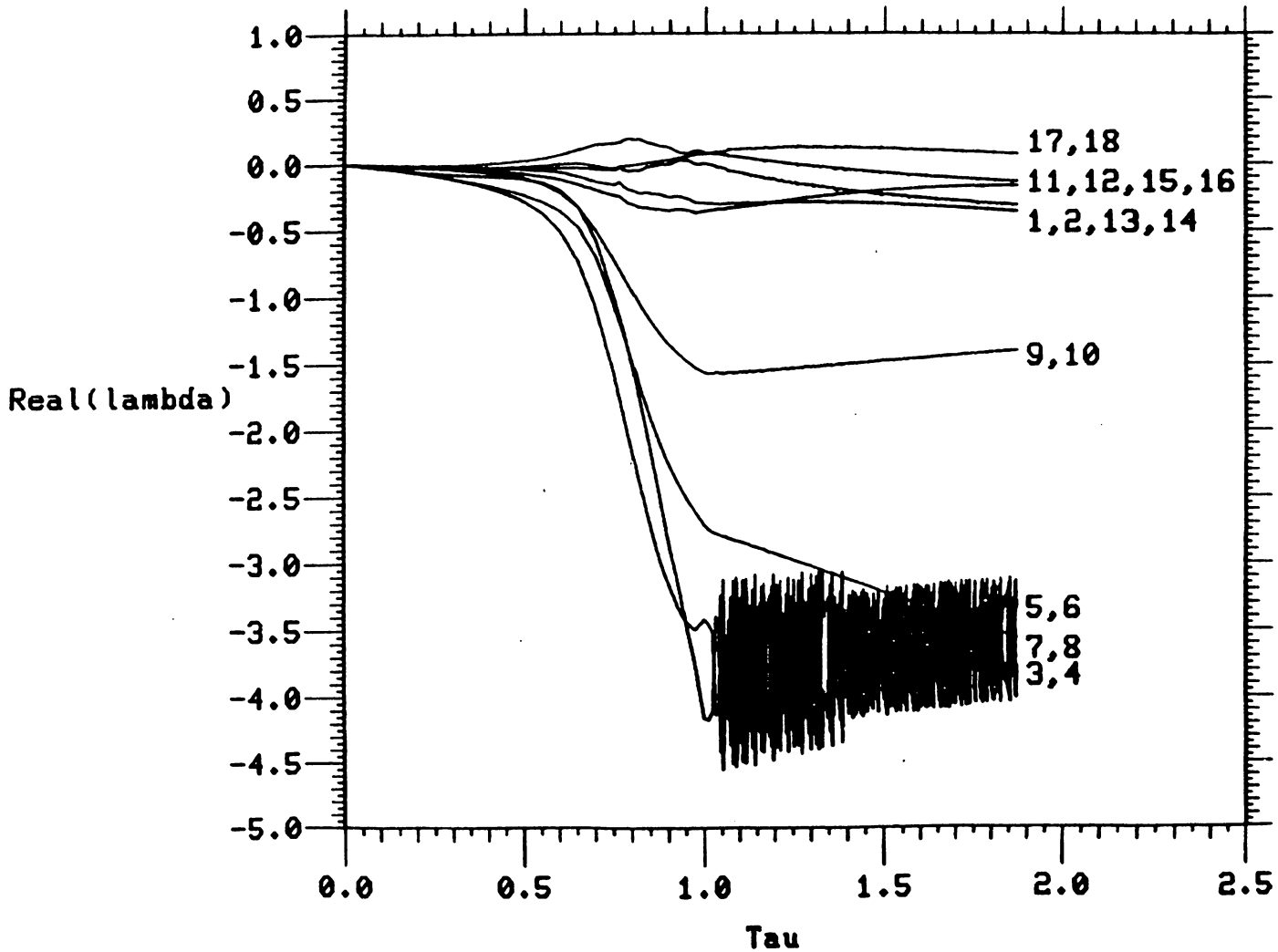


Figure A.40 Real Parts of Eigenvalues

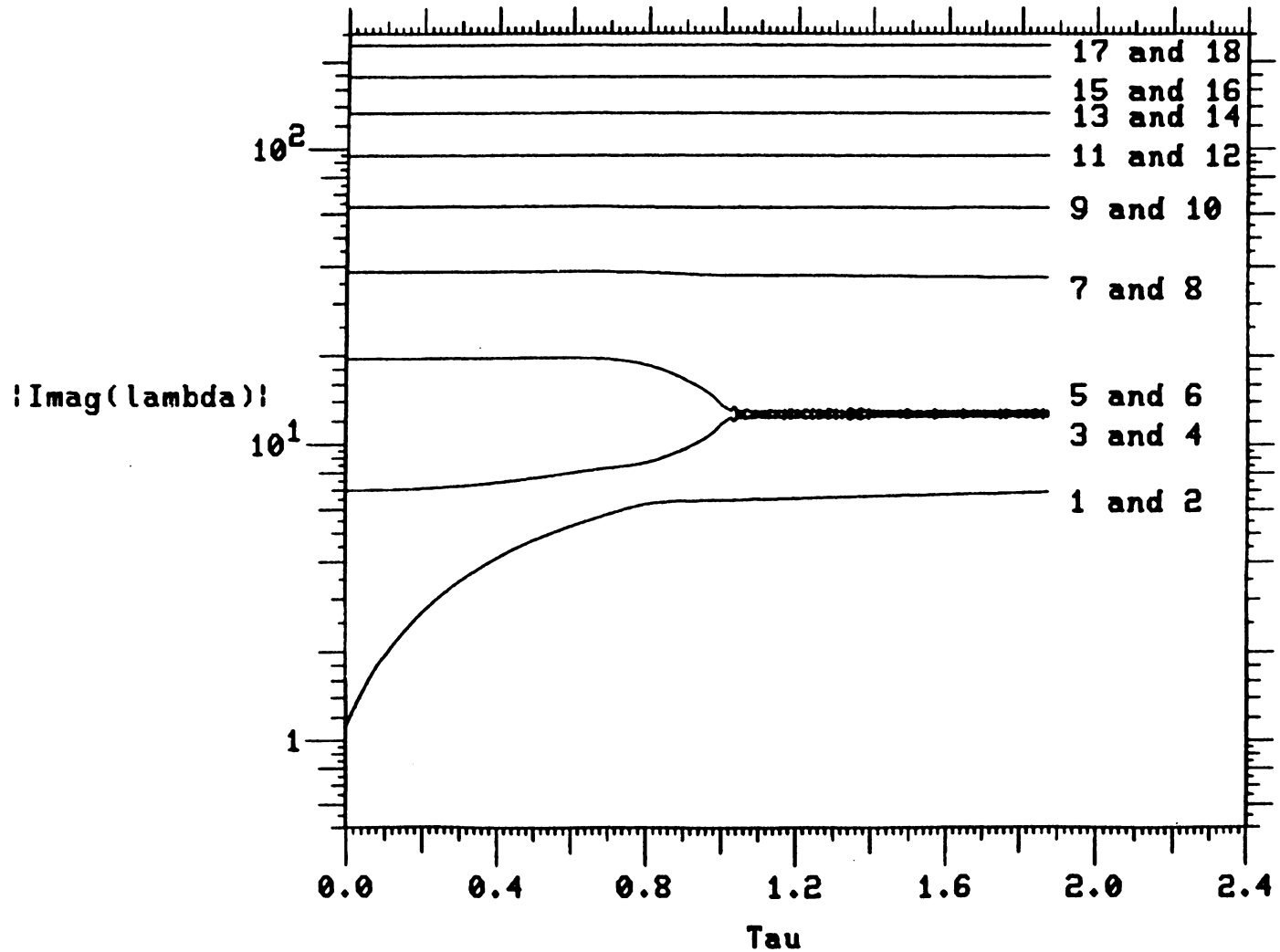


Figure A.41 Imaginary Parts of Eigenvalues

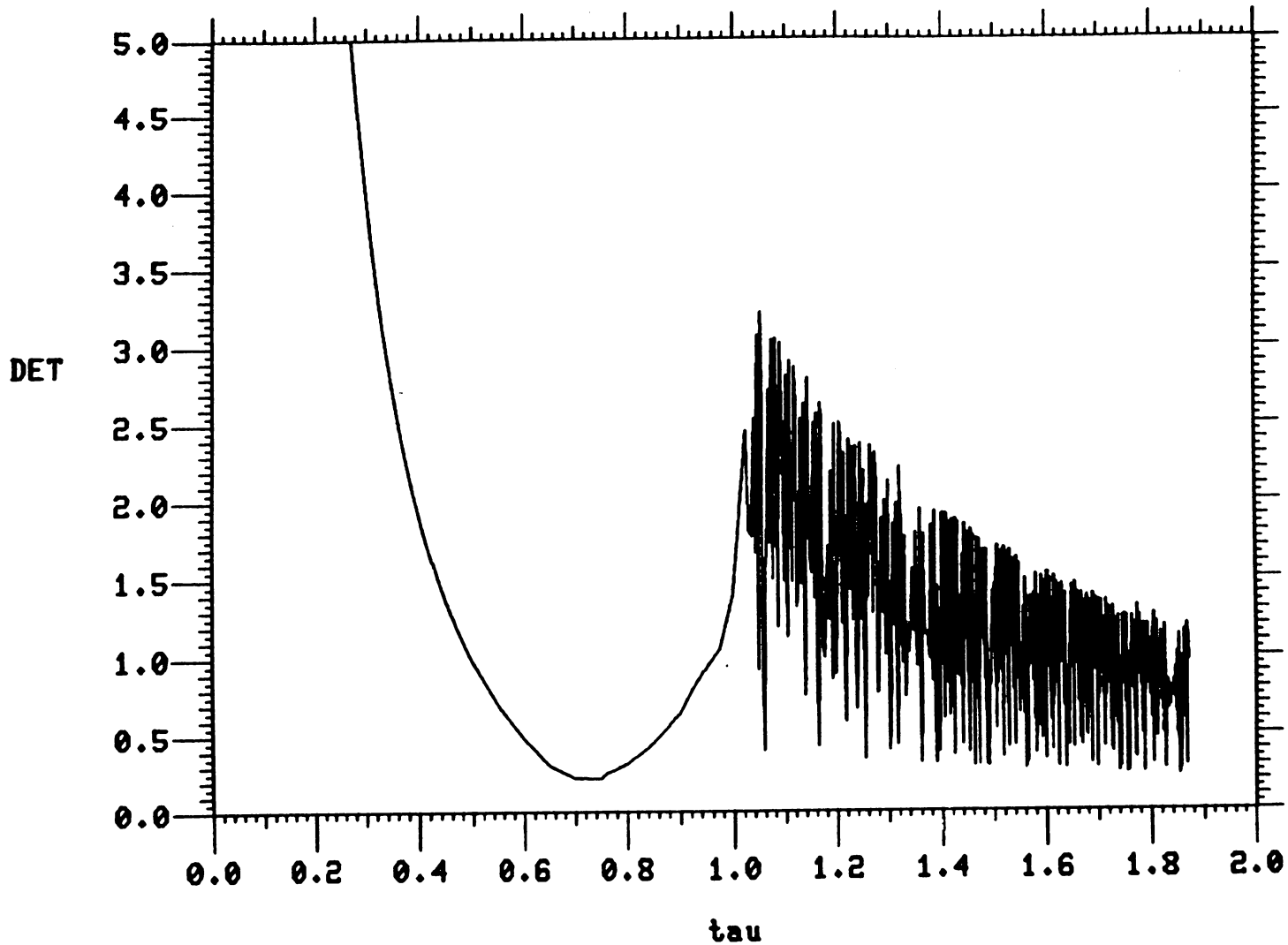


Figure A.42 The Determinant

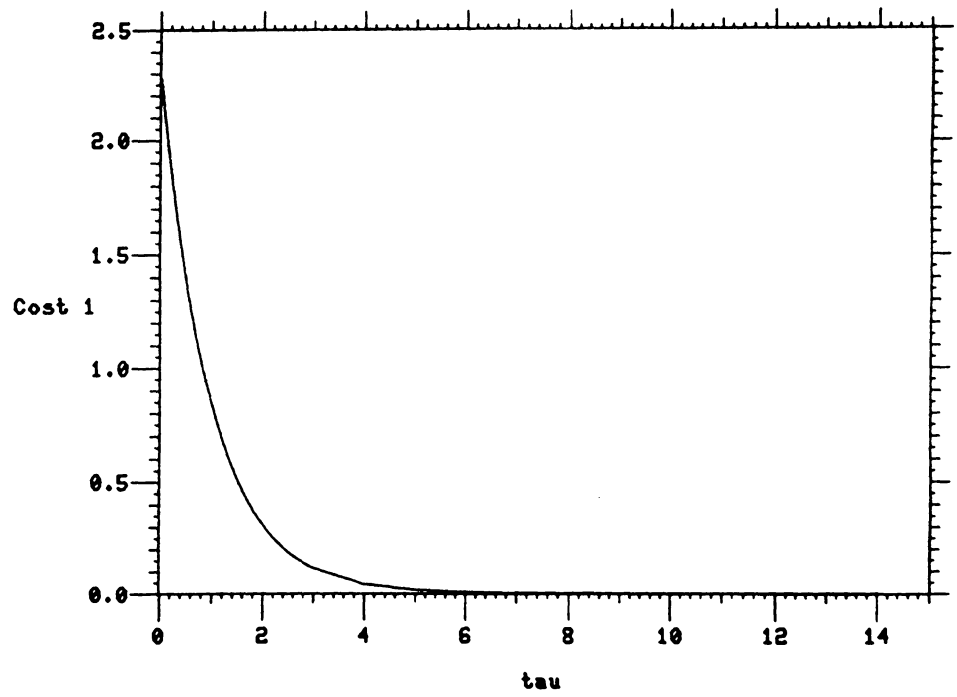
Table A.12
Gains and Parameters

Parameter	Initial Value	Final Value	Parameter	Initial Value	Final Value
tau	0	14			
cost1	2.25	1.8×10^{-6}	cost2	0.0249	3.9×10^{-6}
g_1	0	-0.60	g_{17}	0	-0.017
g_2	0	-0.28	g_{18}	0	-1.6
g_3	0	0.053	g_{19}	0	-0.16
g_4	0	-1.7	g_{20}	0	-0.27
g_5	0	-0.19	g_{21}	0	-0.15
g_6	0	-0.89	g_{22}	0	2.1
g_7	0	-0.20	g_{23}	0	-0.028
g_8	0	-0.40	g_{24}	0	0.68
g_9	0	-0.20	g_{25}	0	-0.31
g_{10}	0	-2.4	g_{26}	0	-0.82
g_{11}	0	-0.32	g_{27}	0	0.026
g_{12}	0	-2.3	g_{28}	0	-0.077
g_{13}	0	-0.26	g_{29}	0	-0.26
g_{14}	0	0.045	g_{30}	0	1.1
g_{15}	0	-0.12	g_{31}	0	-0.42
g_{16}	0	1.5	g_{32}	0	0.54

Table A.13

Eigenvalue Initial and Final States

Eigenvalue	Initial Value rad./sec.		Final Value rad./sec.	
λ_1	0	+ i 1.11	-0.57	+ i 1.25
λ_2	0	- i 1.11	-0.57	- i 1.25
λ_3	0	+ i 6.97	-1.5	+ i 8.3
λ_4	0	- i 6.97	-1.5	- i 8.3
λ_5	0	+ i 19.5	-2.7	+ i 27
λ_6	0	- i 19.5	-2.7	- i 27
λ_7	0	+ i 38.5	-0.73	+ i 38
λ_8	0	- i 38.5	-0.73	- i 38
λ_9	0	+ i 63.3	-1.4	+ i 64
λ_{10}	0	- i 63.3	-1.4	- i 64
λ_{11}	0	+ i 94.7	-3.8	+ i 95
λ_{12}	0	- i 94.7	-3.8	- i 95
λ_{13}	0	+ i 132	-0.49	+ i 132
λ_{14}	0	- i 132	-0.49	- i 132
λ_{15}	0	+ i 177	-0.49	+ i 178
λ_{16}	0	- i 177	-0.49	- i 178
λ_{17}	0	+ i 228	-3.7	+ i 228
λ_{18}	0	- i 228	-3.7	- i 228



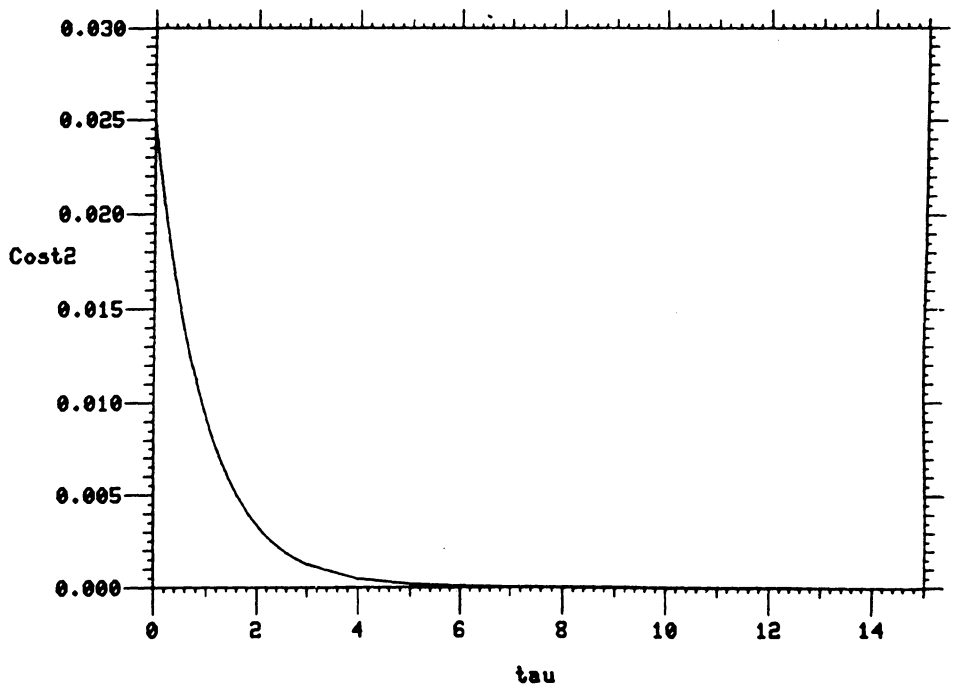


Figure A.43 Cost Trajectories

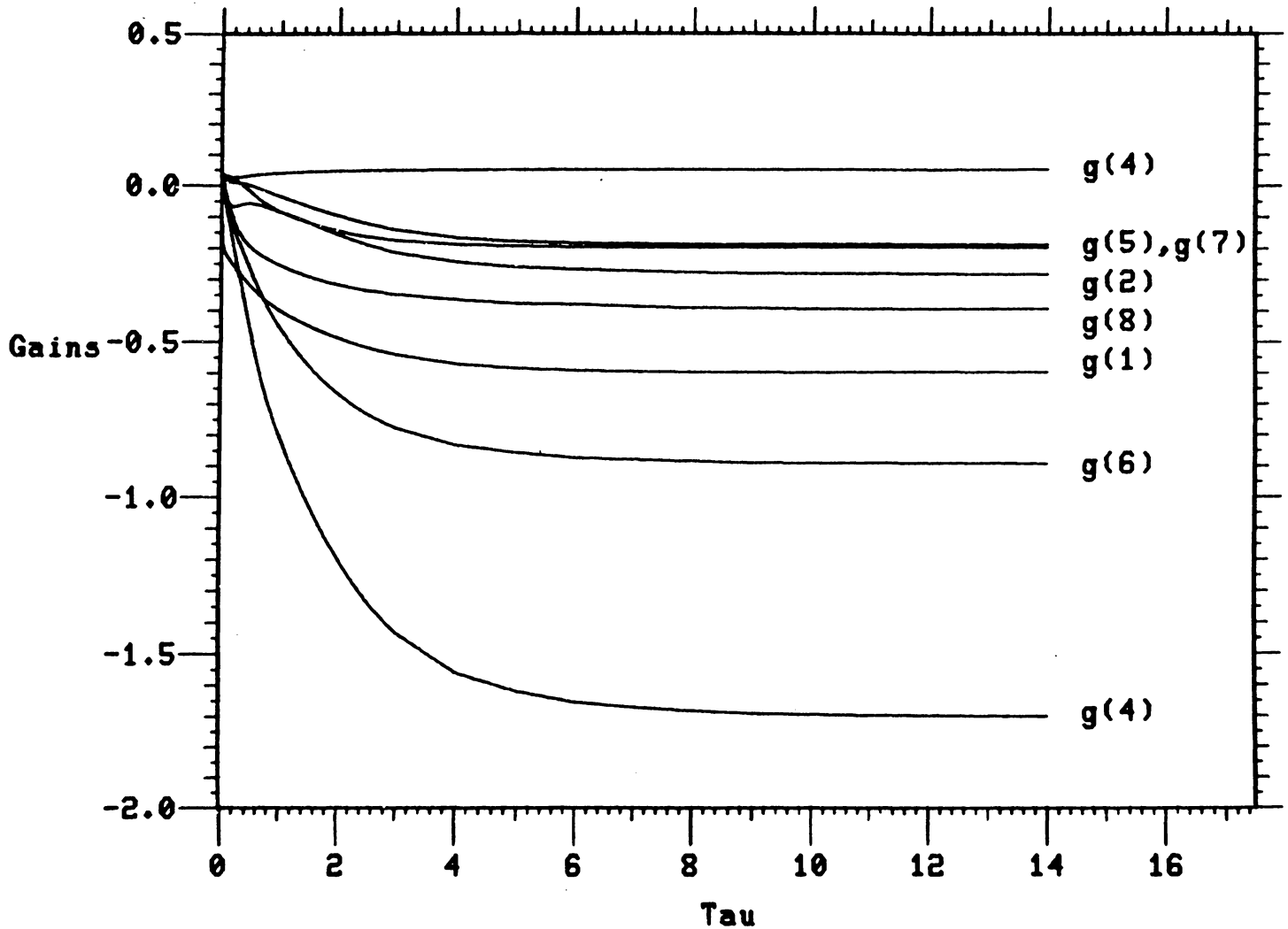


Figure A.44 Gain Plots

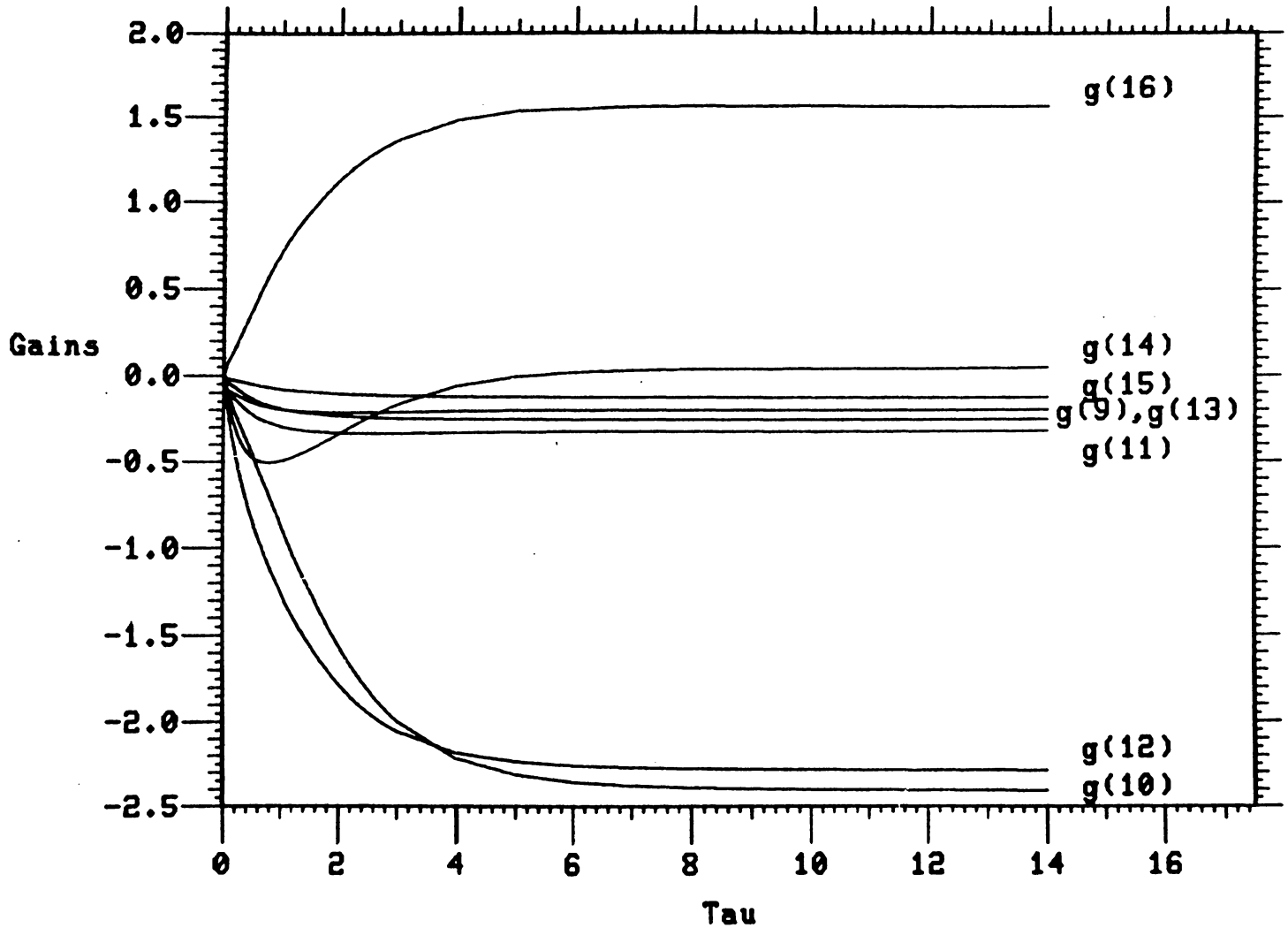


Figure A.45 Gain Plots

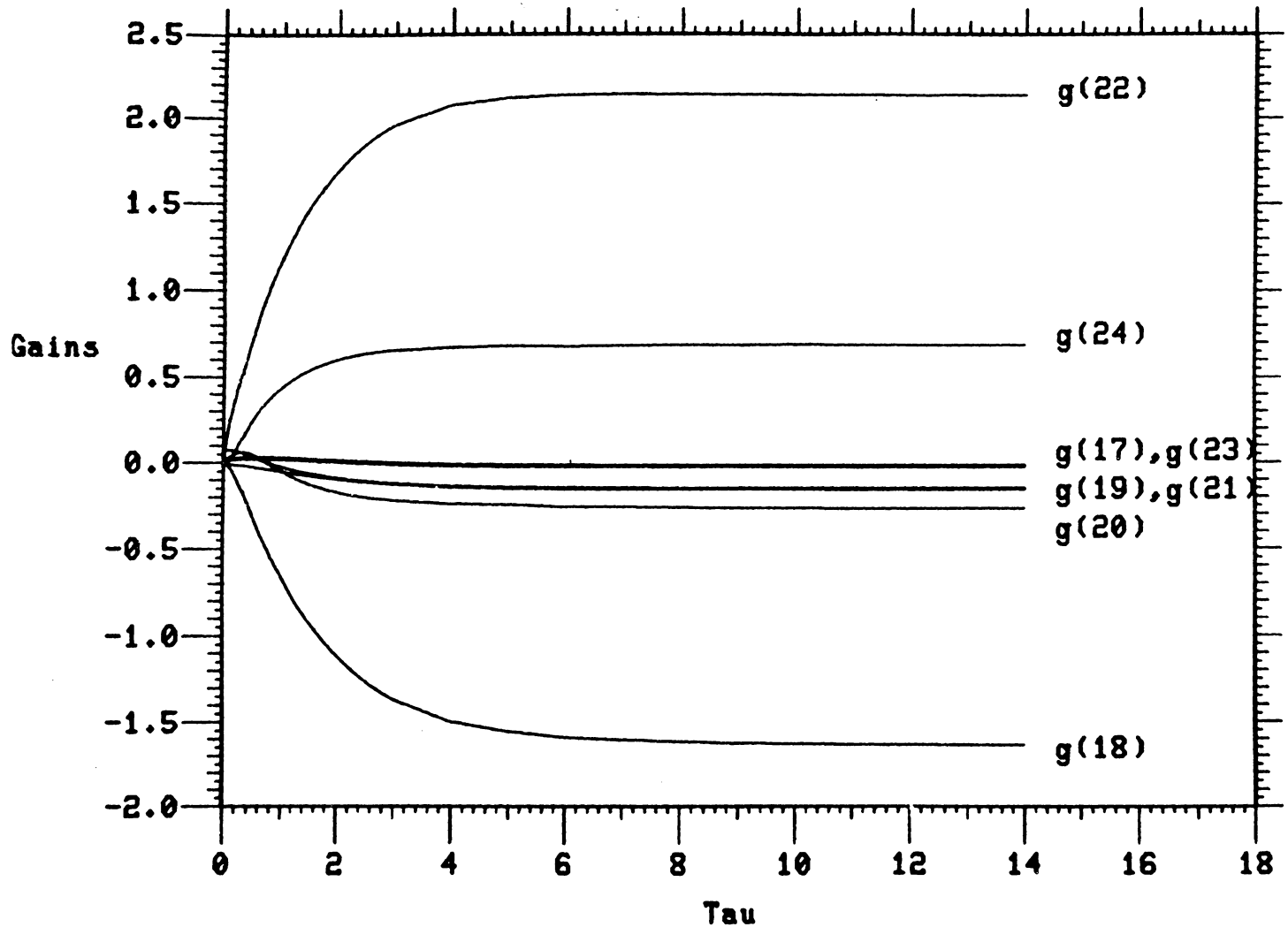


Figure A.46 Gain Plots

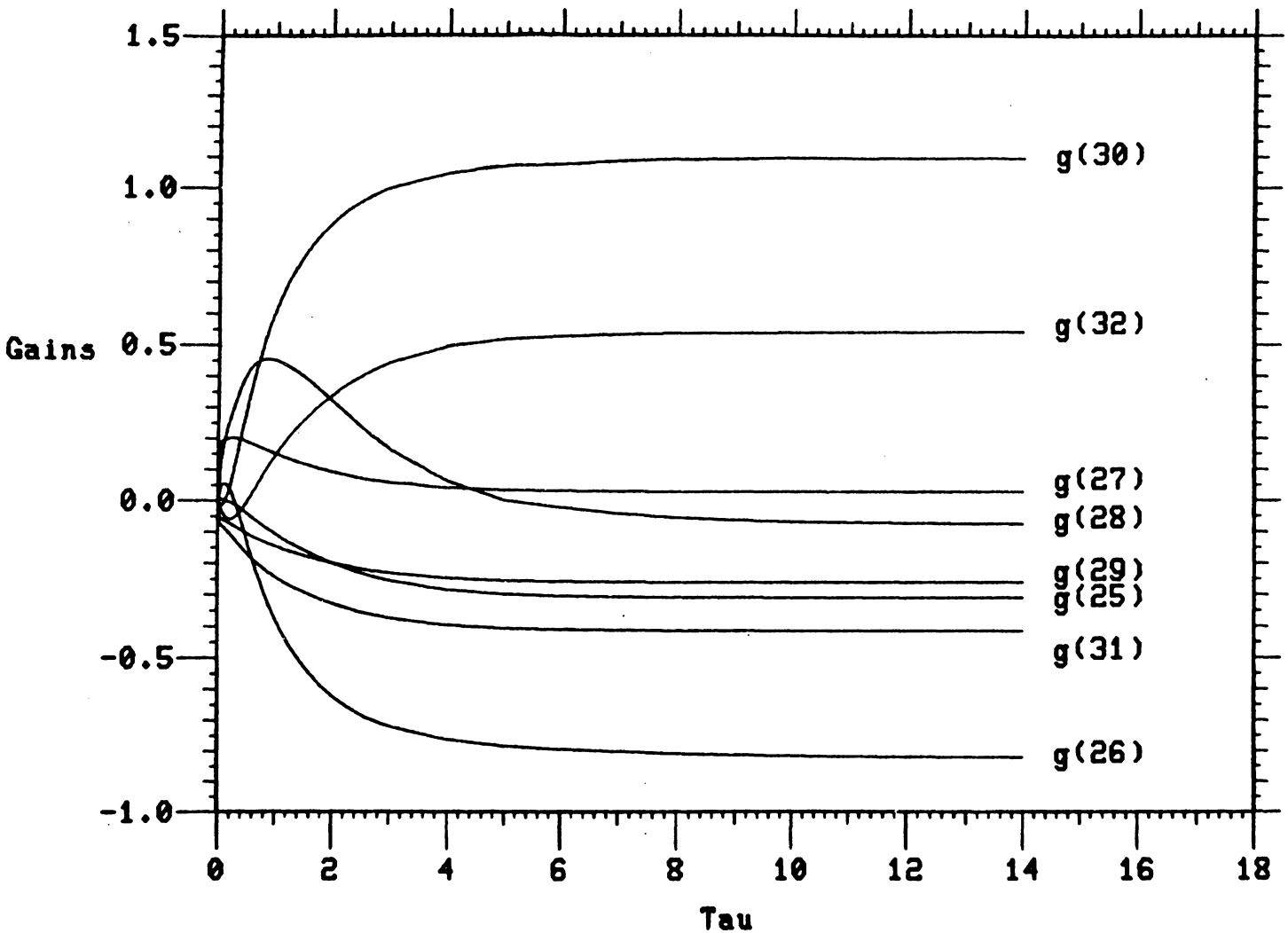


Figure A.47 Gain Plots

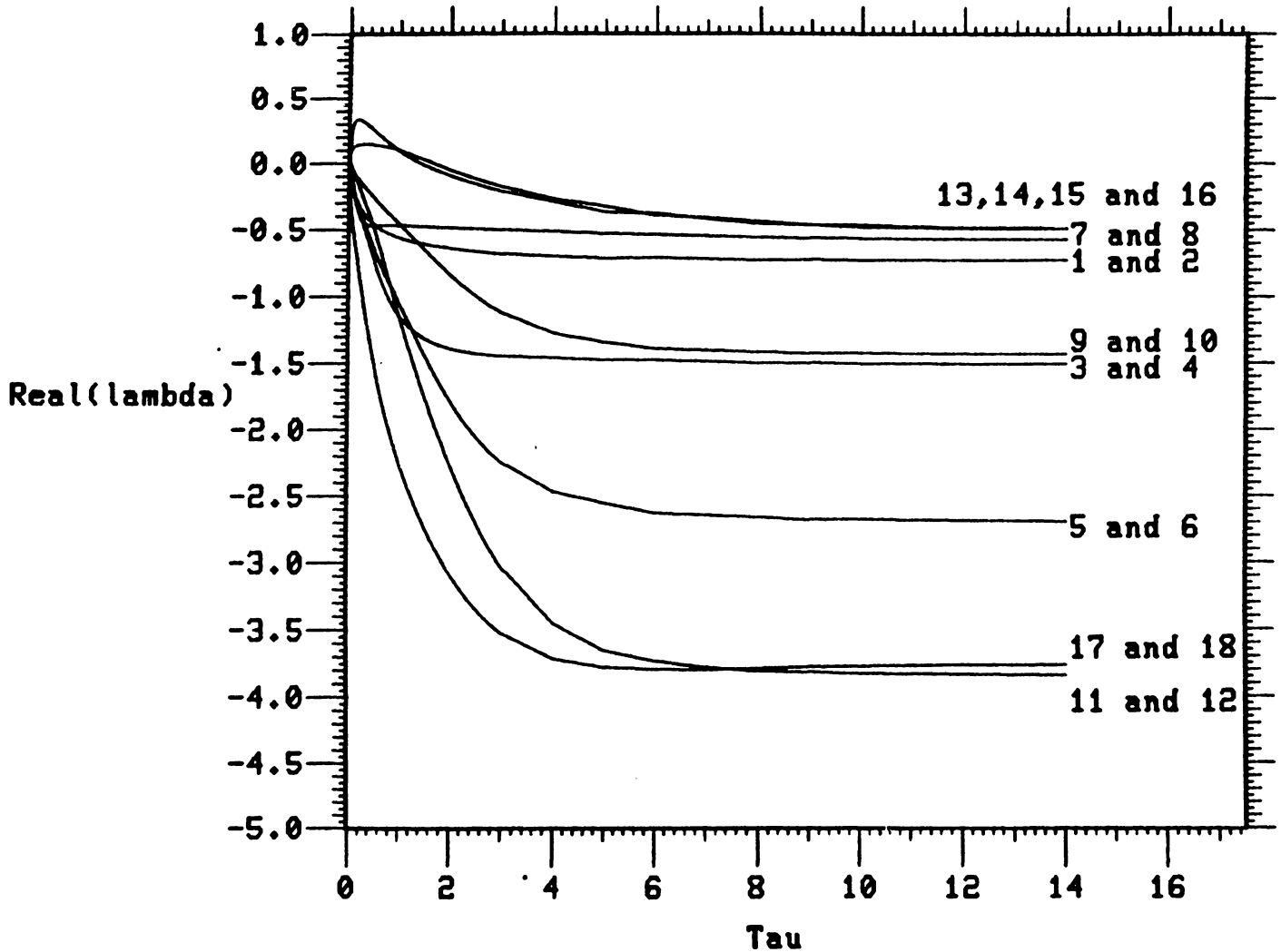


Figure A.48 Real Parts of Eigenvalues

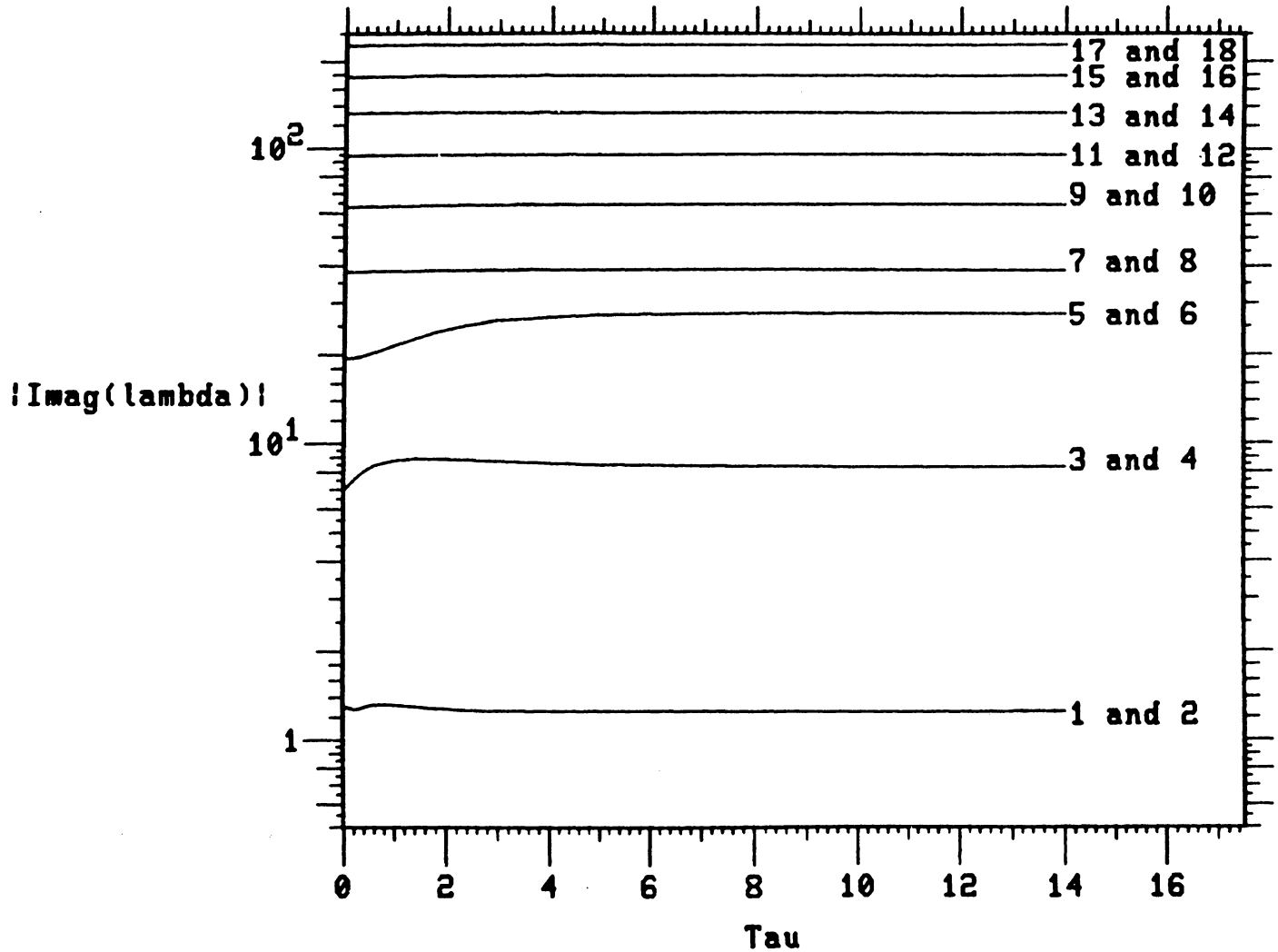


Figure A.49 Imaginary Parts of Eigenvalues

Appendix B

Optimization Routines Listings

```
C     $INSERT DIMENSIONS.INS
C
C     THIS INSERT FILE IS USED TO SPECIFY DIMENSIONS OF THE STRUCTURE
C     MODEL.
C
C     NELEM IS THE NUMBER OF ELEMENTS IN THE FINITE ELEMENT MODEL.
C     NA IS THE DIMENSION OF THE MODEL WRITTEN IN FIRST ORDER FORM.
C     NA MUST BE 4*NELEM, AND NELEM MUST BE < 21.
C
      PARAMETER ( NELEM = 12 )
      PARAMETER ( NA = 48 )
C
C     NGAIN IS THE NUMBER OF GAINS.
C
      PARAMETER ( NGAIN = 32 )
C



C     MAIN PROGRAM GRAD
C
C     THIS MAIN PROGRAM IMPLEMENTS THE GRADIENT ALGORITHM FOR THE
C     DESIGN OF A FEEDBACK CONTROL LAW FOR A FLEXIBLE BEAM.  THE MAIN
C     PROGRAM SIMPLY SETS UP THE INTEGRATION AND PASSES THE PROBLEM OFF TO
C     THE INTEGRATION ALGORITHM IN SUBROUTINE DHPCG.  SEE SUBROUTINE GRAD
C     FOR DETAILS ON THE GRADIENT ALGORITHM.
C
C*************************************************************************
C
C     FOLLOW THESE STEPS TO RECONFIGURE THE PROGRAM:
C
C           I.    CHANGE THE NUMBER OF ELEMENTS OR GAINS IN THE INSERT FILE
C                 DIMENSIONS.INS.
C
C           II.   CHANGE THE BAR STIFFNESS, MASS PER UNIT LENGTH, OR LENGTH
C                 IN SUBROUTINE STRUCT.
C
C           III.  CHANGE THE WAY THE GAINS FEEDBACK IN SUBROUTINE STRDER.
C
C           IV.   CHANGE THE PERFORMANCE MEASURE IN SUBROUTINE PERFOR.
C
C           V.    CHANGE THE PROGRAM OUTPUT IN SUBROUTINES PERFOR AND OUTP.
C
C*************************************************************************
C
C     LOCAL:
C
C           NGAIN  INTEGER PARAMETER IN THE INSERT FILE WHICH CONTAINS THE
C                  NUMBER OF GAINS.
C
C           G      REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE GAIN VALUES.
C
C           GDER   REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE DERIVATIVE
C                  OF THE GAINS WRT THE INDEPENDENT VARIABLE.  THESE ARE ALSO
C                  USED AS ERROR WEIGHTS BY DHPCG AT FIRST CALL.
C
C           PRMT   REAL*8DIMENSIONED (5) WHICH IS USED TO COMMUNICATE WITH
```

```
C                   DHPCG.
C
C                   PRMT(1)        LOWER BOUND OF THE INDEPENDENT VARIABLE.
C                   PRMT(2)        UPPER BOUND OF THE INDEPENDENT VARIABLE.
C                   PRMT(3)        INITIAL INCREMENT OF THE INDEPENDENT VARIABLE.
C                   PRMT(4)        UPPER ABSOLUTE ERROR BOUND USED TO ADJUST
C                                  STEP SIZE.
C                   PRMT(5)        A FLAG WHICH MUST BE SET TO ZERO INITIALLY.
C                                  SETTING PRMT(5) NONZERO IN OUTP WILL STOP
C                                  EXECUTION OF DHPCG.
C
C             IHLF   ON OUTPUT FROM DHPCG, SPECIFIES THE NUMBER OF BISECTIONS
C                    OF THE INITIAL INCREMENT THAT WERE NECESSARY.
C
C             AUX    REAL*8 DIMENSIONED (16,NGAIN) WORKING SPACE FOR DHPCG.
C
C             START  A FLAG TO COMMUNICATE WITH SUBROUTINE GRAD TO INDICATE
C                    GRAD'S FIRST PASS.
C
C     EXTERNAL FUNCTIONS:
C
C             GRAD   THE SUBROUTINE WHICH RETURNS GDER(*), THE GAIN DERIVATIVES,
C                    TO DHPCG.
C
C             OUTP   CALLED BY DHPCG TO FACILITATE OUTPUT.
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      EXTERNAL GRAD,OUTP
C
$INSERT  DIMENSIONS.INS
C
      LOGICAL*2 START
C
      COMMON/RSTART/ START
C
      DIMENSION PRMT(5),G(NGAIN),GDER(NGAIN),AUX(16,NGAIN)
C
C     I HAVE PUT THE INTEGRATION IN A DO LOOP TO HAVE MORE CONTROL OVER IT.
C     SET UP THE INITIAL PARAMETERS.
C
      START=.TRUE.
C
      DO 30 I=1,3
C
      IF( I .EQ. 1 ) THEN
C
       PRMT(1)=0.D0
       PRMT(2)=0.5D0
       PRMT(3)=0.01D0
       PRMT(4)=.001D0
       PRMT(5)=0.
C
      ENDIF
      IF ( I .EQ. 2 ) THEN
C
       PRMT(1)=0.5D0
       PRMT(2)=3.D0
       PRMT(3)=0.1D0
```

```
       PRMT(4)=.01D0
       PRMT(5)=0.
C
      ENDIF
      IF ( I .EQ. 3 ) THEN
C
       PRMT(1)=3.D0
C      WAS 120
       PRMT(2)=240.D0
C      WAS 2.0
       PRMT(3)=1.D0
       PRMT(4)=.01D0
       PRMT(5)=0.
C
      ENDIF
C
C     SET INITIAL CONDITIONS IN G(*) AND WEIGHTS IN GDER(*).
C
      DO 10 J=1,NGAIN
      IF( I .EQ. 1 ) G(J)=0.D0
10    GDER(J)=1.D0/NGAIN
C
C     NOW PASS OFF TO THE INTEGRATION PACKAGE.
C
      CALL DHPCG(PRMT,G,GDER,NGAIN,IHLF,GRAD,OUTP,AUX)
C
30    CONTINUE
      STOP
      END



      SUBROUTINE GRAD(TAU,G,GDER)
C
C     SUBROUTINE GRAD GENERATES THE DERIVATIVES IN GDER(*) WHICH
C     ARE NEEDED TO SOLVE THE DIFFERENTIAL EQUATIONS.  GRAD IS CALLED BY THE
C     DIFFERENTIAL EQUATION PACKAGE AND RETURNS GDER(*).  THE ALGORITHM
C     IMPLEMENTS THE EQUATIONS FOR STRUCTURAL DYNAMICS WRITTEN IN THE FORM:
C
C            -   -              -   -
C           | X'' |            |  X' |
C           | --- |    =    A  | --- |
C           |  X' |            |  X  |
C            -   -              -   -
C
C     INPUTS:
C
C            TAU    REAL*8 INDEPENDENT VARIABLE OF INTEGRATION
C
C            G      REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE CURRENT GAIN
C                   VALUES
C
C     OUTPUTS:
C
C            GDER   REAL*8 DIMENSIONED (NGAIN)  CONTAINS THE DERIVATIVES OF
C                   THE GAINS NEEDED TO IMPLEMENT THE GRADIENT ALGORITHM.
C
C     LOCAL:
C
```

```
C             NELEM  INTEGER PARAMETER IN THE INSERT FILE WHICH IS THE NUMBER
C                    OF ELEMENTS
C
C             NA     INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                    NUMBER OF DEGREES OF FREEDOM.  ( MUST BE 4*NELEM  )
C
C             NGAIN  INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                    NUMBER OF GAINS.
C
C             NLIMIT INTEGER USED TO TELL SUBROUTINE SECDER HOW MANY SECOND
C                    DERIVATIVES ARE NEEDED.  THIS IS DONE TO SAVE CPU TIME.
C
C             START  LOGICAL*2 LOCATED IN COMMON WHICH IS USED TO INDICATE THE
C                    FIRST PASS OR RESTART OF THE ALGORITHM.  THE STRUCTURE IS
C                    REBUILT WHEN START=.TRUE..
C
C             A      REAL*8 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                    COEFFICIENT MATRIX FOR THE STRUCTURE WITH CURRENT GAINS
C                    INCLUDED.
C
C
C             A2     REAL*8 DIMENSIONED (NA,NA) WHICH IS ANOTHER COPY OF A.  THE
C                    EIGENVALUE ROUTINE EIGEN OVERWRITES THE COPY OF A.
C
C             ATRANS REAL*8 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                    TRANSPOSE OF A.
C
C             APERM  REAL*8 DIMENSIONED (NA,NA) IN COMMON WHICH CONTAINS THE
C                    COEFFICIENT MATRIX FOR ZERO GAINS.
C
C             TEMPM  REAL*8 DIMENSIONED (NA,NA)   WHICH IS USED TO
C                    PASS INFORMATION FROM SUBROUTINE STRUCT TO STRDER.  THE
C                    INTERMEDIATE CALCULATION TEMPM IMPROVES PROGRAM EFFICIENCY.
C
C             ADER   REAL*8 DIMENSIONED (NA,NA,NGAIN) IN COMMON CONTAINS THE
C                    PARTIAL OF THE A MATRIX WRT THE GAINS.
C
C             PDER   REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE PARTIAL
C                    OF THE PERFORMANCE MEASURE WRT THE GAINS.
C
C             WK     REAL*8 DIMENSIONED (6960)   IS USED AS WORKING
C                    SPACE FOR A LIBRARY CALL AND RESTRICTS NA<81.
C
C             ER     REAL*8 DIMENSIONED (NA)   IS THE REAL PART OF THE
C                    EIGENVALUES.
C
C             EI     REAL*8 DIMENSIONED (NA)   IS THE IMAGINARY PART
C                    OF THE EIGENVALUES.
C
C             V      REAL*8 DIMENSIONED (NA,NA)   IS THE EIGENVECTORS OF
C                    A(*,*) RETURNED FROM SUBROUTINE EIGEN AND STACKED IN SOME
C                    UNCLEAR WAY.  SUBROUTINE CONJID SORTS THEM.
C
C             CEIG   COMPLEX*16 DIMENSIONED (NA)   WHICH CONTAINS THE
C                    COMPLEX EIGENVALUES.
C
C             CVECTL COMPLEX*16 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                    COMPLEX LEFT EIGENVECTORS STORED BY COLUMN.
C
```

```
C             CVECTR COMPLEX*16 DIMENSIONED (NA,NA)  WHICH CONTAINS THE
C                    COMPLEX RIGHT EIGENVECTORS STORED BY COLUMN.
C
C             CEIGD  COMPLEX*16 DIMENSIONED (NA,NGAIN) WHICH CONTAINS THE
C                    COMPLEX DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C
C             CEIGD2 COMPLEX*16 DIMENSIONED (NA,NGAIN,NGAIN) WHICH CONTAINS THE
C                    COMPLEX SECOND DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C
C             IDENT  INTEGER*4 DIMENSIONED (NA) CONTAINS INFO USEFUL TO SORT
C                    THE EIGENVALUES AND EIGENVECTORS.
C
C                    IDENT(I)=0 => LAMBDA(I)=0
C                    IDENT(I)=1 => LAMBDA(I) COMPLEX, FIRST OF TWO.
C                    IDENT(I)=2 => LAMBDA(I) COMPLEX, SECOND OF TWO.
C
C                    IDENT(I)=0 => EIGENVECTOR IN COLUMN I OF V.
C                    IDENT(I)=1 => REAL PART OF EIGENVECTOR IN COLUMN I OF V,
C                                  IMAG PART IN COLUMN I+1 OF V.
C                    IDENT(I)=2 => REAL PART OF EIGENVECTOR IN COLUMN I-1 OF V,
C                                  NEGATIVE OF IMAG PART IN COLUMN I OF V.
C
C             ISV    INTEGER*4 USED TO PASS INFO TO A LIBRARY ROUTINE.
C
C             ISL    INTEGER*4 USED TO PASS INFO TO A LIBRARY ROUTINE.
C
C             IERR   INTEGER*4 USED TO GET ERROR MESSAGES FROM A LIBRARY
C                    ROUTINE.
C
C     BECAUSE THE COMPUTER THIS CODE WAS DEVELOPED ON HAS A DYNAMIC STACK SIZE
C     RESTRICTION OF 64K, LARGE ARRAYS HAVE BEEN PLACED IN COMMON.  SEVERAL
C     OTHER VARIABLES ARE SAVED BETWEEN CALLS TO GRAD THROUGH USE OF COMMON.
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
$INSERT DIMENSIONS.INS
C
      DIMENSION A(NA,NA), A2(NA,NA), ATRANS(NA,NA), TEMPM(NA,NA)
C
      DIMENSION ADER(NA,NA,NGAIN)
C
      DIMENSION APERM(NA,NA)
C
      DIMENSION G(NGAIN),GDER(NGAIN),PDER(NGAIN)
C
      DIMENSION WK( 6960 )
C
      DIMENSION ER(NA), EI(NA)
C
      DIMENSION V( NA, NA )
C
      COMMON /RMAIN/ WK,APERM,A,ER,EI,V,TEMPM,NLIMIT
C
      COMMON /RMAIN2/ A2
C
      COMMON /RMAIN3/ ADER
C
      COMPLEX*16 CEIG(NA),CVECTL(NA,NA),CVECTR(NA,NA)
```

```
C
      COMMON /CMAIN/ CEIG,CVECTL,CVECTR,CEIGD
C
      COMMON /CMAIN2/ CEIGD2
C
      COMPLEX*16 CEIGD(NA,NGAIN),CEIGD2(NA,NGAIN,NGAIN)
C
      DIMENSION IDENT(NA)
C
      LOGICAL*2 START
C
      COMMON/RSTART/ START
C
C*************************************************************************
C
C     THE SUBROUTINE CAN BE ORGANIZED INTO FIVE FUNCTIONAL PARTS:
C
C           I.   INITIALIZE IF START=.TRUE.
C
C           II.  INCREMENT APERM(*) BY THE GAINS TO GET A CURRENT STRUCTURE
C                MATRIX.
C
C           III. CALCULATE THE EIGENVALUES AND DERIVATIVES OF THE EIGENVALUES
C                AND EIGENVECTORS.
C
C           IV.  CALL SUBROUTINE PERFOR TO FIND THE DERIVATIVES OF THE
C                PERFORMANCE MEASURE WRT THE GAINS.
C
C           V.   APPLY THE GRADIENT ALGORITHM TO CALCULATE THE GAIN
C                DERIVATIVES GDER.
C
C*************************************************************************
C
C     I.   INITIALIZE IF START=.TRUE.
C
C          START=.TRUE. INDICATES A FIRST PASS THROUGH THIS SUBROUTINE.
C
       IF( START ) THEN
C
C       SUBROUTINE STRUCT WILL USE A FINITE ELEMENT MODEL TO FIND A AND ADER.
C
        CALL STRUCT(APERM, TEMPM, ALEN, NA, NELEM)
C
C       CALCULATE ADER(*,*,*).
C
        CALL STRDER( ADER, NGAIN, TEMPM, NA, ALEN )
C
C       APERM AND ADER HAVE BEEN INITIALIZED.  SET THE INITIALIZATION FLAG
C       TO FALSE.
C
        START=.FALSE.
C
C       LOOK AT A MAXIMUM OF HALF THE EIGENVALUES.
C
C       NLIMIT=NA/2
        NLIMIT=0
C
       ENDIF
C
```

```
C     II.  INCREMENT APERM(*) BY THE GAINS TO GET A CURRENT STRUCTURE
C          MATRIX.
C
C          NOW INCREMENT A BY THE AMOUNTS CAUSED BY GAIN FEEDBACK.
C          USE THE PERMANENT VERSION OF A FROM SUBROUTINE STRUCT.
C
           DO 5 I=1,NA
           DO 5 J=1,NA
            A(I,J)=APERM(I,J)
            DO 5 K=1,NGAIN
5          A(I,J)=A(I,J)+G(K)*ADER(I,J,K)
C
C     III. CALCULATE THE EIGENVALUES AND DERIVATIVES OF THE EIGENVALUES AND
C          EIGENVECTORS.
C
C          THE TRANSPOSE OF A IS NEEDED FOR THE LEFT EIGENVALUE PROBLEM.
C          THE A2 MATRIX SAVES A.  A IS OVERWRITTEN IN SUBROUTINE EIGEN.
C
              DO 15 I=1,NA
              DO 15 J=I,NA
C
                  ATRANS(I,J)=A(J,I)
                  ATRANS(J,I)=A(I,J)
C
                  A2(I,J)=A(I,J)
                  A2(J,I)=A(J,I)
C
15            CONTINUE
C
C          NOW SET THE VARIABLES NEEDED FOR THE EIGEN CALL.
C
              ISV=NA
              ILV=0
C
C          DO THE RIGHT EIGENVALUE PROBLEM.
C          REMEMBER THAT THE ROUTINE EIGEN DESTROYS A.
C
            CALL EIGEN(NA,NA,A,ER,EI,ISV,ILV,V,WK,IERR)
C
C          NOW CALL THE SUBROUTINE WHICH SORTS FOR COMPLEX PAIRS AND RETURNS
C          THE IDENTIFICATION CODE IDENT(*).  THIS ALSO PLACES THE EIGENVALUES
C          AND VECTORS IN CONVENIENT COMPLEX VARIABLES.
C
            CALL CONJID( ER, EI, V, CEIG, CVECTR, IDENT, NA )
C
C          NOW DO THE LEFT EIGENVALUE PROBLEM.
C
            CALL EIGEN(NA,NA,ATRANS,ER,EI,ISV,ILV,V,WK,IERR)
C
            CALL CONJID( ER, EI, V, CEIG, CVECTL, IDENT, NA )
C
C          NOW DO THE EIGENVALUE DERIVATIVE PROBLEM.
C
            CALL EIGDER(CEIG,CVECTL,CVECTR,IDENT,ADER,CEIGD,NA,NGAIN)
C
C          NOW CALCULATE THE SECOND DERIVATIVES.
C
            CALL SECDER( NA, NGAIN, NLIMIT, ADER, CVECTL, CVECTR,
     1                   IDENT, A2, CEIGD, CEIG, CEIGD2          )
```

```
C
C     IV.  CALL SUBROUTINE PERFOR TO FIND THE DERIVATIVES OF THE
C          PERFORMANCE MEASURE WRT THE GAINS.
C
           CALL PERFOR( G, CEIG, CEIGD, CVECTR, PDER, NA, NGAIN, TAU )
C
C     V.   APPLY THE GRADIENT ALGORITHM TO CALCULATE THE GAIN
C          DERIVATIVES GDER.
C
           DO 25 I=1,NGAIN
25         GDER(I) = -PDER(I)
C
C     GDER NOW CONTAINS THE NEEDED DERIVATIVES.
C
C     THAT'S ALL FOLKS......
C
      RETURN
      END


      SUBROUTINE PERFOR( G, CEIG, CEIGD, CVECTR, PDER, NA, NGAIN, TAU )
C
C     SUBROUTINE PERFOR GENERATES THE DERIVATIVES OF THE PERFORMANCE MEASURE
C     WRT THE GAINS.  THIS IS A USER PROVIDED SUBROUTINE WHICH IS CHANGED
C     TO REFLECT DIFFERENT PERFORMANCE MEASURES.  THIS ROUTINE SHOULD ALSO
C     BE USED FOR THE GENERATION OF OUTPUT.
C
C     INPUTS:
C
C            G       REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE CURRENT
C                    GAINS.
C
C            CEIG    COMPLEX*16 DIMENSIONED (NA) WHICH CONTAINS THE
C                    COMPLEX EIGENVALUES.
C
C            CEIGD   COMPLEX*16 DIMENSIONED (NA,NGAIN) WHICH CONTAINS
C                    THE COMPLEX DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C
C            CVECTR COMPLEX*16 DIMENSIONED (NA,NA) WHICH CONTAINS THE
C                    COMPLEX RIGHT EIGENVECTORS STORED BY COLUMN.
C
C            NA      INTEGER PARAMETER WHICH INDICATES THE NUMBER OF DEGREES OF
C                    FREEDOM. ( IN FIRST ORDER FORM )
C
C            NGAIN   INTEGER PARAMETER WHICH INDICATES THE NUMBER OF GAINS.
C
C            TAU     REAL*8 INDEPENDENT VARIABLE OF INTEGRATION.
C
C     OUTPUTS:
C
C            PDER    REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE PARTIAL
C                    DERIVATIVES OF THE PERFORMANCE MEASURE WRT THE GAINS.
C
C     LOCAL:
C
C            TOTAL   REAL*8 WHICH IS THE TOTAL FOR THE PERFORMANCE MEASURE
C                    AT THIS SET OF GAINS.  THIS IS USED PRIMARILY FOR
C                    OUTPUT.
```

```
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMPLEX*16 CEIG(NA), CEIGD(NA,NGAIN), CVECTR(NA,NA)
      REAL*8     PDER(NGAIN), G(NGAIN)
C
      COMMON /PER/ TOTAL
C
C     THIS RUN WILL ONLY MAKE USE OF GAIN 1 AND 2.  GAIN 1 IS A UNIFORM FORCE
C     ON THE LAST ELEMENT FEEDING BACK ON END POSITION.  GAIN 2 IS A UNIFORM
C     FORCE ON THE LAST ELEMENT FEEDING BACK ON END POSITION RATE.  THE
C     PDER(3) AND PDER(4) VALUES WILL BE RETURNED AS ZERO.
C
C     NOW CALCULATE THE TOTAL PERFORMACE MEASURE A THIS STEP.  I AM LOOKING
C     AT ONLY THE FIRST 18 EIGENVALUES. (FIRST NINE PAIRS)  THESE EIGENVALUES
C     WILL BE PUSHED TO THE LEFT OF REAL PART=-0.5D0.
C
C     NOTE:  I HAVE ADDED COMPUTATIONS TO INCLUDE
C            ALL GAINS.
C
            TOTAL=0.D0
C
            DO 20 I=1,18
            IF( DREAL(CEIG(I)) .GT. -0.5D0)
     1        TOTAL=TOTAL+( DREAL(CEIG(I))+0.5D0 )**3
20          IF( DABS( DIMAG(CEIG(I)) ) .LT. 5.D0)
     1        TOTAL=TOTAL+2.D-3*(5.D0-DABS( DIMAG(CEIG(I)) ))**3
C
C     NOW I NEED TO LOAD MY PDER.
C     FIRST INITIALIZE:
C
            DO 30 I=1,NGAIN
30          PDER(I)=0.0D0
C
C     NOW BUILD PER(1) AND PDER(2) FROM THE PERFORMANCE MEASURE DERIVATIVES:
C
            DO 40 I=1,18
C
            IF( DREAL(CEIG(I)) .GT. -0.5D0) THEN
C
                  DO 50 J=1,NGAIN
50                PDER(J)=PDER(J)+3.D0*( DREAL(CEIG(I))+0.5D0 )**2
     1                  * DREAL(CEIGD(I,J))
C
            ENDIF
C
            IF( DABS( DIMAG(CEIG(I)) ) .LT. 5.D0) THEN

                  DO 55 J=1,NGAIN
C
                  IF( DIMAG(CEIG(I)) .GT.0.D0 )
     1            PDER(J)=PDER(J)+3.D0*2.D-3
     2                  * (5.D0-DABS( DIMAG(CEIG(I)) ))**2
     3                  * ( - DIMAG(CEIGD(I,J) ) )
C
                  IF( DIMAG(CEIG(I)) .LT.0.D0 )
     1            PDER(J)=PDER(J)+3.D0*2.D-3
     2                  * (5.D0-DABS( DIMAG(CEIG(I)) ))**2
```

```
     3                      * (  DIMAG(CEIGD(I,J) ) )
C
55                 CONTINUE
C
           ENDIF
C
40         CONTINUE
C
C     NOW PRINT OUT SOME INTERESTING VALUES.
C
C           WRITE(1,60)TAU,TOTAL
60         FORMAT(' PERFOR ENTERED WITH TAU=',D16.7,
     1             ' AND TOTAL=',D16.7)
C           WRITE(1,70)(PDER(I), I=1,NGAIN)
70         FORMAT(' IN PERFOR, THE GAIN DERIVATIVES ARE:'/4D20.7)
C
C     THAT'S ALL FOLKS...
C
      RETURN
      END


      SUBROUTINE OUTP(TAU,G,GDER,IHLF,NDIM,PRMT)
C
C     SUBROUTINE OUTP IS USED FOR OUTPUT, AS REQUESTED BY THE INTEGRATION
C     ROUTINE.
C
C     INPUTS:
C
C           TAU    REAL*8 INDEPENDENT VARIABLE OF INTEGRATION.
C
C           G      REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE GAINS.
C
C           GDER   REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE DERIVATIVE
C
C           NA     INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                  NUMBER OF DEGREES OF FREEDOM.  ( MUST BE 4*NELEM  )
C
C           NGAIN  INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                  NUMBER OF GAINS.
C
C           CEIG   COMPLEX*16 DIMENSIONED (NA)   WHICH CONTAINS THE
C                  COMPLEX EIGENVALUES.
C
C           CVECTL COMPLEX*16 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                  COMPLEX LEFT EIGENVECTORS STORED BY COLUMN.
C
C           CVECTR COMPLEX*16 DIMENSIONED (NA,NA)  WHICH CONTAINS THE
C                  COMPLEX RIGHT EIGENVECTORS STORED BY COLUMN.
C
C           CEIGD  COMPLEX*16 DIMENSIONED (NA,NGAIN) WHICH CONTAINS THE
C                  COMPLEX DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C                  OF THE GAINS WRT TAU.
C
C           CEIGD2 COMPLEX*16 DIMENSIONED (NA,NGAIN,NGAIN) WHICH CONTAINS THE
C                  COMPLEX SECOND DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C
C           TOTAL  REAL*8 WHICH IS THE TOTAL FOR THE PERFORMANCE MEASURE
```

```
C                   AT THIS SET OF GAINS.
C
C             PRMT  REAL*8DIMENSIONED (5) WHICH IS USED TO COMMUNICATE WITH
C                   DHPCG.
C
C                   PRMT(1)       LOWER BOUND OF THE INDEPENDENT VARIABLE.
C                   PRMT(2)       UPPER BOUND OF THE INDEPENDENT VARIABLE.
C                   PRMT(3)       INITIAL INCREMENT OF THE INDEPENDENT VARIABLE.
C                   PRMT(4)       UPPER ABSOLUTE ERROR BOUND USED TO ADJUST
C                                 STEP SIZE.
C                   PRMT(5)       A FLAG WHICH MUST BE SET TO ZERO INITIALLY.
C                                 SETTING PRMT(5) NONZERO IN OUTP WILL STOP
C                                 EXECUTION OF DHPCG.
C
C             IHLF  ON OUTPUT FROM DHPCG, SPECIFIES THE NUMBER OF BISECTIONS
C                   OF THE INITIAL INCREMENT THAT WERE NECESSARY.
C
C     OUTPUTS:      NONE
C
C     LOCAL:
C
C             NCOUNT INDICATES THE NUMBER OF TIMES OUTP HAS BEEN CALLED.
C                   NCOUNT ALLOWS ORDERLY PROGRAM STOPPING.
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
$INSERT DIMENSIONS.INS
C
      DIMENSION G(NDIM), GDER(NDIM), PRMT(5)
C
      COMPLEX*16 CEIG(NA),CVECTL(NA,NA),CVECTR(NA,NA)
      COMPLEX*16 CEIGD2(NA,NGAIN,NGAIN)
C
      COMMON /CMAIN/ CEIG,CVECTL,CVECTR,CEIGD
C
      COMMON /CMAIN2/ CEIGD2
C
      COMMON /PER/ TOTAL
C
      COMMON /OUT/ NCOUNT
C
C
      COMPLEX*16 CEIGD(NA,NGAIN)
C
C     NOTE:  I HAVE ASSUMED THAT NCOUNT IS ORIGINALLY SET TO ZERO.
C     THIS MAY NEED MODIFICATION IF THE PROGRAM IS PORTED.
C
C     WRITE(1,*)CEIG,CVECTR
      NCOUNT = NCOUNT + 1
C
      WRITE(1,*)TAU,IHLF,NCOUNT,(G(I),I=1,NDIM)
10    FORMAT(/'TAU=',D16.7,'  IHLF=',I16,'  NCOUNT=',I4
     1       /'GAINS:',4(/2D20.7))
C
C     NOW PRINT OUT SOME INTERESTING VALUES.
C
            WRITE(1,*)TAU,TOTAL
60          FORMAT(//' AT TAU=',D16.7,' TOTAL=',D16.7/)
C
```

```
C
            DO 70 I=1,NA/2,2
70          WRITE(1,*)CEIG(I)
C
C     NOW HALT EXECUTION IF NCOUNT IS LARGE.
C
      IF( NCOUNT .EQ. 1000 ) STOP
C
      RETURN
      END
```

Appendix C

Continuation Routines Listings

```
C     $INSERT DIMENSIONS.INS
C
C     THIS INSERT FILE IS USED TO SPECIFY DIMENSIONS OF THE STRUCTURE
C     MODEL.
C
C     NELEM IS THE NUMBER OF ELEMENTS IN THE FINITE ELEMENT MODEL.
C     NA IS THE DIMENSION OF THE MODEL WRITTEN IN FIRST ORDER FORM.
C     NA MUST BE 4*NELEM, AND NELEM MUST BE < 21.
C
      PARAMETER ( NELEM = 12 )
      PARAMETER ( NA = 48 )
C
C     NGAIN IS THE NUMBER OF GAINS.
C
      PARAMETER ( NGAIN = 32 )
C
C     NPER IS THE NUMBER OF PERFORMANCE MEASURES.
C
      PARAMETER ( NPER = 2 )
C



C     MAIN PROGRAM CONT
C
C     THIS MAIN PROGRAM IMPLEMENTS THE CONTINUATION ALGORITHM FOR THE
C     DESIGN OF A FEEDBACK CONTROL LAW FOR A FLEXIBLE BEAM.  THE MAIN
C     PROGRAM SIMPLY SETS UP THE INTEGRATION AND PASSES THE PROBLEM OFF TO
C     THE INTEGRATION ALGORITHM IN SUBROUTINE DHPCG.  SEE SUBROUTINE CONT
C     FOR DETAILS ON THE CONTINUATION ALGORITHM.
C
C*************************************************************************
C
C     FOLLOW THESE STEPS TO RECONFIGURE THE PROGRAM:
C
C           I.    CHANGE THE NUMBER OF ELEMENTS OR GAINS IN THE INSERT FILE
C                 DIMENSIONS.INS.
C
C           II.   CHANGE THE BAR STIFFNESS, MASS PER UNIT LENGTH, OR LENGTH
C                 IN SUBROUTINE STRUCT.
C
C           III.  CHANGE THE WAY THE GAINS FEEDBACK IN SUBROUTINE STRDER.
C
C           IV.   CHANGE THE PERFORMANCE MEASURE IN SUBROUTINE PERFOR.
C
C           V.    CHANGE THE PROGRAM OUTPUT IN SUBROUTINES PERFOR AND OUTP.
C
C*************************************************************************
C
C     LOCAL:
C
C           NGAIN  INTEGER PARAMETER IN THE INSERT FILE WHICH CONTAINS THE
C                  NUMBER OF GAINS.
C
C           G      REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE GAIN VALUES.
C
C           GDER   REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE DERIVATIVE
C                  OF THE GAINS WRT THE INDEPENDENT VARIABLE.  THESE ARE ALSO
C                  USED AS ERROR WEIGHTS BY DHPCG AT FIRST CALL.
```

```
C
C            PRMT   REAL*8DIMENSIONED (5) WHICH IS USED TO COMMUNICATE WITH
C                   DHPCG.
C
C                   PRMT(1)        LOWER BOUND OF THE INDEPENDENT VARIABLE.
C                   PRMT(2)        UPPER BOUND OF THE INDEPENDENT VARIABLE.
C                   PRMT(3)        INITIAL INCREMENT OF THE INDEPENDENT VARIABLE.
C                   PRMT(4)        UPPER ABSOLUTE ERROR BOUND USED TO ADJUST
C                                  STEP SIZE.
C                   PRMT(5)        A FLAG WHICH MUST BE SET TO ZERO INITIALLY.
C                                  SETTING PRMT(5) NONZERO IN OUTP WILL STOP
C                                  EXECUTION OF DHPCG.
C
C            IHLF   ON OUTPUT FROM DHPCG, SPECIFIES THE NUMBER OF BISECTIONS
C                   OF THE INITIAL INCREMENT THAT WERE NECESSARY.
C
C            AUX    REAL*8 DIMENSIONED (16,NGAIN) WORKING SPACE FOR DHPCG.
C
C            START  A FLAG TO COMMUNICATE WITH SUBROUTINE CONT TO INDICATE
C                   CONT'S FIRST PASS.
C
C     EXTERNAL FUNCTIONS:
C
C            CONT   THE SUBROUTINE WHICH RETURNS GDER(*), THE GAIN DERIVATIVES,
C                   TO DHPCG.
C
C            OUTP   CALLED BY DHPCG TO FACILITATE OUTPUT.
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      EXTERNAL CONT,OUTP
C
$INSERT  DIMENSIONS.INS
C
      LOGICAL*2 START
C
      COMMON/RSTART/ START
C
      DIMENSION PRMT(5),G(NGAIN),GDER(NGAIN),AUX(16,NGAIN)
C
C     I HAVE PUT THE INTEGRATION IN A DO LOOP TO HAVE MORE CONTROL OVER IT.
C     SET UP THE INITIAL PARAMETERS.
C
      START=.TRUE.
C
      DO 30 I=1,3
C
      IF( I .EQ. 1 ) THEN
C
       PRMT(1)=0.D0
       PRMT(2)=0.5D0
       PRMT(3)=0.01D0
       PRMT(4)=.001D0
       PRMT(5)=0.
C
      ENDIF
      IF ( I .EQ. 2 ) THEN
C
       PRMT(1)=0.5D0
```

```
       PRMT(2)=3.D0
       PRMT(3)=0.1D0
       PRMT(4)=.01D0
       PRMT(5)=0.
C
      ENDIF
      IF ( I .EQ. 3 ) THEN
C
       PRMT(1)=3.D0
       PRMT(2)=120.D0
       PRMT(3)=2.D0
       PRMT(4)=.01D0
       PRMT(5)=0.
C
      ENDIF
C
C     SET INITIAL CONDITIONS IN G(*) AND WEIGHTS IN GDER(*).
C
      DO 10 J=1,NGAIN
      IF( I .EQ. 1 ) G(J)=0.D0
10    GDER(J)=1.D0/NGAIN
C
C     NOW PASS OFF TO THE INTEGRATION PACKAGE.
C
      CALL DHPCG(PRMT,G,GDER,NGAIN,IHLF,CONT,OUTP,AUX)
C
30    CONTINUE
      STOP
      END



      SUBROUTINE CONT(TAU,G,GDER)
C
C     SUBROUTINE CONT GENERATES THE DERIVATIVES IN GDER(*) WHICH
C     ARE NEEDED TO SOLVE THE DIFFERENTIAL EQUATIONS.  GRAD IS CALLED BY THE
C     DIFFERENTIAL EQUATION PACKAGE AND RETURNS GDER(*).  THE ALGORITHM
C     IMPLEMENTS THE EQUATIONS FOR STRUCTURAL DYNAMICS WRITTEN IN THE FORM:
C
C            -   -              -   -
C           | X'' |            |  X' |
C           | --- |    =   A  | --- |
C           |  X' |            |  X  |
C            -   -              -   -
C
C     INPUTS:
C
C            TAU    REAL*8 INDEPENDENT VARIABLE OF INTEGRATION
C
C            G      REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE CURRENT GAIN
C                   VALUES
C
C     OUTPUTS:
C
C            GDER   REAL*8 DIMENSIONED (NGAIN)  CONTAINS THE DERIVATIVES OF
C                   THE GAINS NEEDED TO IMPLEMENT THE GRADIENT ALGORITHM.
C
C     LOCAL:
C
```

```
C            NELEM  INTEGER PARAMETER IN THE INSERT FILE WHICH IS THE NUMBER
C                   OF ELEMENTS
C
C            NA     INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                   NUMBER OF DEGREES OF FREEDOM.  ( MUST BE 4*NELEM  )
C
C            NGAIN  INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                   NUMBER OF GAINS.
C
C            NPER   INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                   NUMBER OF PERFORMANCE CONSTRAINTS.
C
C            NLIMIT INTEGER USED TO TELL SUBROUTINE SECDER HOW MANY SECOND
C                   DERIVATIVES ARE NEEDED.  THIS IS DONE TO SAVE CPU TIME.
C
C            START  LOGICAL*2 LOCATED IN COMMON WHICH IS USED TO INDICATE THE
C                   FIRST PASS OR RESTART OF THE ALGORITHM.  THE STRUCTURE IS
C                   REBUILT WHEN START=.TRUE..
C
C            A      REAL*8 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                   COEFFICIENT MATRIX FOR THE STRUCTURE WITH CURRENT GAINS
C                   INCLUDED.
C
C
C            A2     REAL*8 DIMENSIONED (NA,NA) WHICH IS ANOTHER COPY OF A.  THE
C                   EIGENVALUE ROUTINE EIGEN OVERWRITES THE COPY OF A.
C
C            ATRANS REAL*8 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                   TRANSPOSE OF A.
C
C            APERM  REAL*8 DIMENSIONED (NA,NA) IN COMMON WHICH CONTAINS THE
C                   COEFFICIENT MATRIX FOR ZERO GAINS.
C
C            TEMPM  REAL*8 DIMENSIONED (NA,NA)   WHICH IS USED TO
C                   PASS INFORMATION FROM SUBROUTINE STRUCT TO STRDER.  THE
C                   INTERMEDIATE CALCULATION TEMPM IMPROVES PROGRAM EFFICIENCY.
C
C            ADER   REAL*8 DIMENSIONED (NA,NA,NGAIN) IN COMMON CONTAINS THE
C                   PARTIAL OF THE A MATRIX WRT THE GAINS.
C
C            PER    REAL*8 DIMENSIONED (NPER) WHICH CONTAINS THE CURRENT VALUE
C                   OF THE PERFORMANCE CONSTRAINTS.  PER IS STORED IN
C                   COMMON /PERF/ SO THAT OUTPUT CAN PRINT IT.
C
C            PDER   REAL*8 DIMENSIONED (NPER,NGAIN) WHICH CONTAINS THE PARTIAL
C                   OF THE PERFORMANCE MEASURES WRT THE GAINS.
C
C            AMATIN REAL*8 DIMENSIONED (NPER,NPER) USED IN THE APPLICATION
C                   OF THE CONTINUATION FORMULA.  THIS IS THE MATRIX TO
C                   INVERT.
C
C            WORK   REAL*8 DIMENSIONED (80) PROVIDES WORKING SPACE FOR A
C                   LINPACK CALL.
C
C            IPVT   INT VECTOR DIMENSIONED 80 USED BY LINPACK.
C
C            INFO   INTEGER USED BY THE LINPACK ROUTINES.
C
C            DET    REAL*8 USED BY THE LINPACK ROUTINES.  IT CONTAINS THE
```

```
C                DETERMINANT AND IS PASSED THROUGH COMMON TO SUBROUTINE
C                OUTP.
C
C           PTEMP  REAL*8 DIMENSIONED (NPER) USED TO STORE AN INTERMEDIATE
C                  CALCULATION.
C
C           WK     REAL*8 DIMENSIONED (6960)   IS USED AS WORKING
C                  SPACE FOR A LIBRARY CALL AND RESTRICTS NA<81.
C
C           ER     REAL*8 DIMENSIONED (NA)   IS THE REAL PART OF THE
C                  EIGENVALUES.
C
C           EI     REAL*8 DIMENSIONED (NA)   IS THE IMAGINARY PART
C                  OF THE EIGENVALUES.
C
C           V      REAL*8 DIMENSIONED (NA,NA)   IS THE EIGENVECTORS OF
C                  A(*,*) RETURNED FROM SUBROUTINE EIGEN AND STACKED IN SOME
C                  UNCLEAR WAY.  SUBROUTINE CONJID SORTS THEM.
C
C           CEIG   COMPLEX*16 DIMENSIONED (NA)   WHICH CONTAINS THE
C                  COMPLEX EIGENVALUES.
C
C           CVECTL COMPLEX*16 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                  COMPLEX LEFT EIGENVECTORS STORED BY COLUMN.
C
C           CVECTR COMPLEX*16 DIMENSIONED (NA,NA)  WHICH CONTAINS THE
C                  COMPLEX RIGHT EIGENVECTORS STORED BY COLUMN.
C
C           CEIGD  COMPLEX*16 DIMENSIONED (NA,NGAIN) WHICH CONTAINS THE
C                  COMPLEX DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C
C           CEIGD2 COMPLEX*16 DIMENSIONED (NA,NGAIN,NGAIN) WHICH CONTAINS THE
C                  CQMPLEX SECOND DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C
C           IDENT  INTEGER*4 DIMENSIONED (NA) CONTAINS INFO USEFUL TO SORT
C                  THE EIGENVALUES AND EIGENVECTORS.
C
C                  IDENT(I)=0 => LAMBDA(I)=0
C                  IDENT(I)=1 => LAMBDA(I) COMPLEX, FIRST OF TWO.
C                  IDENT(I)=2 => LAMBDA(I) COMPLEX, SECOND OF TWO.
C
C                  IDENT(I)=0 => EIGENVECTOR IN COLUMN I OF V.
C                  IDENT(I)=1 => REAL PART OF EIGENVECTOR IN COLUMN I OF V,
C                                IMAG PART IN COLUMN I+1 OF V.
C                  IDENT(I)=2 => REAL PART OF EIGENVECTOR IN COLUMN I-1 OF V,
C                                NEGATIVE OF IMAG PART IN COLUMN I OF V.
C
C           ISV    INTEGER*4 USED TO PASS INFO TO A LIBRARY ROUTINE.
C
C           ISL    INTEGER*4 USED TO PASS INFO TO A LIBRARY ROUTINE.
C
C           IERR   INTEGER*4 USED TO GET ERROR MESSAGES FROM A LIBRARY
C                  ROUTINE.
C
C     BECAUSE THE COMPUTER THIS CODE WAS DEVELOPED ON HAS A DYNAMIC STACK SIZE
C     RESTRICTION OF 64K, LARGE ARRAYS HAVE BEEN PLACED IN COMMON.  SEVERAL
C     OTHER VARIABLES ARE SAVED BETWEEN CALLS TO GRAD THROUGH USE OF COMMON.
C
C
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
C
$INSERT DIMENSIONS.INS
C
      DIMENSION A(NA,NA), A2(NA,NA), ATRANS(NA,NA), TEMPM(NA,NA)
C
      DIMENSION ADER(NA,NA,NGAIN)
C
      DIMENSION APERM(NA,NA),AMATIN(NPER,NPER)
C
      DIMENSION G(NGAIN),GDER(NGAIN),PER(NPER),PDER(NPER,NGAIN)
C
      DIMENSION WK(6960),WORK(80),PTEMP(NPER)
C
      INTEGER IPVT(80)
C
      DIMENSION ER(NA), EI(NA)
C
      DIMENSION V( NA, NA )
C
      COMMON /RMAIN/ WK,APERM,A,ER,EI,V,TEMPM,NLIMIT
C
      COMMON /RMAIN2/ A2,ATRANS
C
      COMMON /RMAIN3/ ADER
C
      COMMON /RMAIN4/ PDER,WORK,IPVT,INVMAT,PTEMP
C
      COMPLEX*16 CEIG(NA),CVECTL(NA,NA),CVECTR(NA,NA)
C
      COMMON /CMAIN/ CEIG,CVECTL,CVECTR,CEIGD
C
      COMMON /CMAIN2/ CEIGD2
C
      COMPLEX*16 CEIGD(NA,NGAIN),CEIGD2(NA,NGAIN,NGAIN)
C
      DIMENSION IDENT(NA)
C
      LOGICAL*2 START
C
      COMMON/RSTART/ START
C
      COMMON/PERF/ PER, DET
C
C*************************************************************************
C
C     THE SUBROUTINE CAN BE ORGANIZED INTO FIVE FUNCTIONAL PARTS:
C
C            I.   INITIALIZE IF START=.TRUE.
C
C            II.  INCREMENT APERM(*) BY THE GAINS TO GET A CURRENT STRUCTURE
C                 MATRIX.
C
C            III. CALCULATE THE EIGENVALUES AND DERIVATIVES OF THE EIGENVALUES
C                 AND EIGENVECTORS.
C
C            IV.  CALL SUBROUTINE PGRAD TO FIND THE DERIVATIVES OF THE
C                 PERFORMANCE MEASURE WRT THE GAINS.
C
```

```
C            V.   APPLY THE GRADIENT ALGORITHM TO CALCULATE THE GAIN
C                 DERIVATIVES GDER.
C
C*************************************************************************
C
C     I.   INITIALIZE IF START=.TRUE.
C
C            START=.TRUE. INDICATES A FIRST PASS THROUGH THIS SUBROUTINE.
C
        IF( START ) THEN
C
C        SUBROUTINE STRUCT WILL USE A FINITE ELEMENT MODEL TO FIND A AND ADER.
C
         CALL STRUCT(APERM, TEMPM, ALEN, NA, NELEM)
C
C        CALCULATE ADER(*,*,*).
C
         CALL STRDER( ADER, NGAIN, TEMPM, NA, ALEN )
C
C        APERM AND ADER HAVE BEEN INITIALIZED.  SET THE INITIALIZATION FLAG
C        TO FALSE.
C
         START=.FALSE.
C
C        LOOK AT A MAXIMUM OF HALF THE EIGENVALUES.
C
C        NLIMIT=NA/2
         NLIMIT=0
C
        ENDIF
C
C     II.  INCREMENT APERM(*) BY THE GAINS TO GET A CURRENT STRUCTURE
C          MATRIX.
C
C          NOW INCREMENT A BY THE AMOUNTS CAUSED BY GAIN FEEDBACK.
C          USE THE PERMANENT VERSION OF A FROM SUBROUTINE STRUCT.
C
           DO 5 I=1,NA
           DO 5 J=1,NA
            A(I,J)=APERM(I,J)
            DO 5 K=1,NGAIN
5          A(I,J)=A(I,J)+G(K)*ADER(I,J,K)
C
C     III. CALCULATE THE EIGENVALUES AND DERIVATIVES OF THE EIGENVALUES AND
C          EIGENVECTORS.
C
C          THE TRANSPOSE OF A IS NEEDED FOR THE LEFT EIGENVALUE PROBLEM.
C          THE A2 MATRIX SAVES A.  A IS OVERWRITTEN IN SUBROUTINE EIGEN.
C
              DO 15 I=1,NA
              DO 15 J=I,NA
C
                  ATRANS(I,J)=A(J,I)
                  ATRANS(J,I)=A(I,J)
C
                  A2(I,J)=A(I,J)
                  A2(J,I)=A(J,I)
C
15            CONTINUE
```

```
C
C          NOW SET THE VARIABLES NEEDED FOR THE EIGEN CALL.
C
              ISV=NA
              ILV=0
C
C          DO THE RIGHT EIGENVALUE PROBLEM.
C          REMEMBER THAT THE ROUTINE EIGEN DESTROYS A.
C
               CALL EIGEN(NA,NA,A,ER,EI,ISV,ILV,V,WK,IERR)
C
C          NOW CALL THE SUBROUTINE WHICH SORTS FOR COMPLEX PAIRS AND RETURNS
C          THE IDENTIFICATION CODE IDENT(*).  THIS ALSO PLACES THE EIGENVALUES
C          AND VECTORS IN CONVENIENT COMPLEX VARIABLES.
C
               CALL CONJID( ER, EI, V, CEIG, CVECTR, IDENT, NA )
C
C          NOW DO THE LEFT EIGENVALUE PROBLEM.
C
               CALL EIGEN(NA,NA,ATRANS,ER,EI,ISV,ILV,V,WK,IERR)
C
               CALL CONJID( ER, EI, V, CEIG, CVECTL, IDENT, NA )
C
C          NOW DO THE EIGENVALUE DERIVATIVE PROBLEM.
C
               CALL EIGDER(CEIG,CVECTL,CVECTR,IDENT,ADER,CEIGD,NA,NGAIN)
C
C          NOW CALCULATE THE SECOND DERIVATIVES.
C
               CALL SECDER( NA, NGAIN, NLIMIT, ADER, CVECTL, CVECTR,
     1                      IDENT, A2, CEIGD, CEIG, CEIGD2          )
C
C      IV.  CALL SUBROUTINE PGRAD TO FIND THE DERIVATIVES OF THE
C           PERFORMANCE MEASURE WRT THE GAINS.
C
               CALL PGRAD( G, CEIG, CEIGD, CVECTR, PDER, PER,
     1                     NA, NGAIN, NPER, TAU )
C
C      V.   APPLY THE CONTINUATION ALGORITHM TO CALCULATE THE GAIN
C           DERIVATIVES GDER.  THE WEIGHTING MATRIX HAS BEEN ASSUMED TO BE
C           IDENTITY.
C
C          THE CODE IMPLEMENTS:
C
C                          T          T
C           DG/DT=  - PDER  INV( PDER  PDER ) PER
C
C          FIRST FORM THE SMALL MATRIX TO INVERT.
C
               DO 25 IROW=1,NPER
               DO 25 ICOL=1,NPER
C
                  AMATIN(IROW,ICOL)=0.D0
C
                  DO 25 IINNER=1,NGAIN
C
25                AMATIN(IROW,ICOL) = AMATIN(IROW,ICOL) +
     1                  PDER( IROW , IINNER ) * PDER( ICOL, IINNER )
C
```

```
C          WE NOW INVERT TEMPM USING TWO SUBROUTINES FROM THE LINPACK PACKAGE.
C
              CALL DGEFA(AMATIN,NPER,NPER,IPVT,INFO)
              CALL DGEDI(AMATIN,NPER,NPER,IPVT,DET,WORK,11)
C
C          AMATIN SHOULD NOW CONTAIN THE INVERSE OF AMATIN.
C          POST MULTIPLY BY PER AND STORE THE RESULT IN A TEMPORARY
C          VARIABLE.
C
            DO 30 IROW=1,NPER
C
               PTEMP(IROW)=0.D0
C
               DO 30 ICOL=1,NPER
30             PTEMP(IROW)=PTEMP(IROW) + AMATIN(IROW,ICOL)*PER(ICOL)
C
C          NOW PRE-MULTIPLY BY TRANSPOSE OF PDER.  THE NEGATIVE
C          SIGN IS INCLUDED IN THE OPERATION.
C
            DO 35 IROW=1,NGAIN
C
               GDER(IROW)=0.D0
C
               DO 35 ICOL=1,NPER
35             GDER(IROW)=GDER(IROW)-PDER(ICOL,IROW)*PTEMP(ICOL)
C
C     GDER NOW CONTAINS THE NEEDED DERIVATIVES.
C
C     THAT'S ALL FOLKS......
C
      RETURN
      END



      SUBROUTINE PGRAD( G, CEIG, CEIGD, CVECTR, PDER, PER,
     1                  NA, NGAIN, NPER, TAU )
C
C     SUBROUTINE PGRAD GENERATES THE DERIVATIVES OF THE PERFORMANCE MEASURE
C     WRT THE GAINS.  THIS IS A USER PROVIDED SUBROUTINE WHICH IS CHANGED
C     TO REFLECT DIFFERENT PERFORMANCE MEASURES.  THIS ROUTINE SHOULD ALSO
C     BE USED FOR THE GENERATION OF OUTPUT.
C
C     INPUTS:
C
C            G      REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE CURRENT
C                   GAINS.
C
C            CEIG   COMPLEX*16 DIMENSIONED (NA) WHICH CONTAINS THE
C                   COMPLEX EIGENVALUES.
C
C            CEIGD  COMPLEX*16 DIMENSIONED (NA,NGAIN) WHICH CONTAINS
C                   THE COMPLEX DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C
C            CVECTR COMPLEX*16 DIMENSIONED (NA,NA) WHICH CONTAINS THE
C                   COMPLEX RIGHT EIGENVECTORS STORED BY COLUMN.
C
C            NA     INTEGER PARAMETER WHICH INDICATES THE NUMBER OF DEGREES OF
C                   FREEDOM.  ( IN FIRST ORDER FORM )
```

```
C
C            NGAIN  INTEGER PARAMETER WHICH INDICATES THE NUMBER OF GAINS.
C
C            NPER   INTEGER PARAMETER WHICH INDICATES THE NUMBER OF PERFORMANCE
C                   MEASURES IN THE SUBROUTINE.
C
C            TAU    REAL*8 INDEPENDENT VARIABLE OF INTEGRATION.
C
C     OUTPUTS:
C
C            PDER   REAL*8 DIMENSIONED (NPER,NGAIN) WHICH CONTAINS THE PARTIAL
C                   DERIVATIVES OF THE PERFORMANCE MEASURES WRT THE GAINS.
C
C            PER    REAL*8 DIMENSIONED (NPER) WHICH CONTAINS THE CURRENT
C                   VALUE OF THE PERFORMANCE MEASURES.
C
C     LOCAL:
C
         IMPLICIT REAL*8 (A-H,O-Z)
C
         COMPLEX*16 CEIG(NA), CEIGD(NA,NGAIN), CVECTR(NA,NA)
         REAL*8     PDER(NPER,NGAIN), G(NGAIN),PER(NPER)
C
C     NOW CALCULATE THE TOTAL PERFORMACE MEASURE A THIS STEP.  I AM LOOKING
C     AT ONLY THE FIRST 18 EIGENVALUES. (FIRST NINE PAIRS)  THESE EIGENVALUES
C     WILL BE PUSHED TO THE LEFT OF REAL PART=-0.5D0.
C
C     NOTE:  I HAVE ADDED COMPUTATIONS TO INCLUDE
C            ALL GAINS.
C
            PER(1)=0.D0
            PER(2)=0.D0
C
            DO 20 I=1,18
             IF( DREAL(CEIG(I)) .GT. -0.5D0)
     1        PER(1)=PER(1)+( DREAL(CEIG(I))+0.5D0 )**3
20          IF( DABS( DIMAG(CEIG(I)) ) .LT. 5.D0)
     1        PER(2)=PER(2)+1.D-3/8.D0*(5.D0-DABS( DIMAG(CEIG(I)) ))**3
C
C     NOW I NEED TO LOAD MY PDER.
C     FIRST INITIALIZE:
C
            DO 30 I=1,NGAIN
            PDER(1,I)=0.D0
30          PDER(2,I)=0.0D0
C
C     NOW BUILD PER(1) AND PDER(2) FROM THE PERFORMANCE MEASURE DERIVATIVES:
C
            DO 40 I=1,18
C
            IF( DREAL(CEIG(I)) .GT. -0.5D0) THEN
C
                  DO 50 J=1,NGAIN
50                PDER(1,J)=PDER(1,J)+3.D0*( DREAL(CEIG(I))+0.5D0 )**2
     1                    * DREAL(CEIGD(I,J))
C
            ENDIF
C
            IF( DABS( DIMAG(CEIG(I)) ) .LT. 5.D0) THEN
```

```
                    DO 55 J=1,NGAIN
C
                    IF( DIMAG(CEIG(I)) .GT.0.D0 )
     1              PDER(2,J)=PDER(2,J)+3.D0*1.D-3/8.D0
     2                      * (5.D0-DABS( DIMAG(CEIG(I)) ))**2
     3                      * ( - DIMAG(CEIGD(I,J) ) )
C
                    IF( DIMAG(CEIG(I)) .LT.0.D0 )
     1              PDER(2,J)=PDER(2,J)+3.D0*1.D-3/8.D0
     2                      * (5.D0-DABS( DIMAG(CEIG(I)) ))**2
     3                      * (  DIMAG(CEIGD(I,J) ) )
C
55                  CONTINUE
C
            ENDIF
C
40          CONTINUE
C
C     NOW PRINT OUT SOME INTERESTING VALUES.
C
C            WRITE(1,60)TAU,PER
60          FORMAT(' PERFOR ENTERED WITH TAU=',D16.7,
     1             ' AND TOTALS=',2D16.7)
C            WRITE(1,70)PDER
70          FORMAT(' IN PERFOR, THE GAIN DERIVATIVES ARE:'/4D20.7)
C
C     THAT'S ALL FOLKS...
C
      RETURN
      END


      SUBROUTINE OUTP(TAU,G,GDER,IHLF,NDIM,PRMT)
C
C     SUBROUTINE OUTP IS USED FOR OUTPUT, AS REQUESTED BY THE INTEGRATION
C     ROUTINE.
C
C     INPUTS:
C
C            TAU    REAL*8 INDEPENDENT VARIABLE OF INTEGRATION.
C
C            G      REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE GAINS.
C
C            GDER   REAL*8 DIMENSIONED (NGAIN) WHICH CONTAINS THE DERIVATIVE
C
C            NA     INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                   NUMBER OF DEGREES OF FREEDOM.  ( MUST BE 4*NELEM  )
C
C            NGAIN  INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                   NUMBER OF GAINS.
C
C            NPER   INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                   NUMBER OF PERFORMANCE MEASURES.
C
C            CEIG   COMPLEX*16 DIMENSIONED (NA)   WHICH CONTAINS THE
C                   COMPLEX EIGENVALUES.
C
```

```
C             CVECTL COMPLEX*16 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                    COMPLEX LEFT EIGENVECTORS STORED BY COLUMN.
C
C             CVECTR COMPLEX*16 DIMENSIONED (NA,NA)  WHICH CONTAINS THE
C                    COMPLEX RIGHT EIGENVECTORS STORED BY COLUMN.
C
C             CEIGD  COMPLEX*16 DIMENSIONED (NA,NGAIN) WHICH CONTAINS THE
C                    COMPLEX DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C                    OF THE GAINS WRT TAU.
C
C             CEIGD2 COMPLEX*16 DIMENSIONED (NA,NGAIN,NGAIN) WHICH CONTAINS THE
C                    COMPLEX SECOND DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C
C             PER    REAL*8 DIMENSIONED (NPER) WHICH IS THE TOTALS
C                    FOR THE PERFORMANCE MEASURES
C                    AT THIS SET OF GAINS.
C
C             DET    REAL*8 WHICH CONTAINS THE LAST DETERMINANT IN CONT.
C
C             PRMT   REAL*8DIMENSIONED (5) WHICH IS USED TO COMMUNICATE WITH
C                    DHPCG.
C
C                    PRMT(1)       LOWER BOUND OF THE INDEPENDENT VARIABLE.
C                    PRMT(2)       UPPER BOUND OF THE INDEPENDENT VARIABLE.
C                    PRMT(3)       INITIAL INCREMENT OF THE INDEPENDENT VARIABLE.
C                    PRMT(4)       UPPER ABSOLUTE ERROR BOUND USED TO ADJUST
C                                  STEP SIZE.
C                    PRMT(5)       A FLAG WHICH MUST BE SET TO ZERO INITIALLY.
C                                  SETTING PRMT(5) NONZERO IN OUTP WILL STOP
C                                  EXECUTION OF DHPCG.
C
C             IHLF   ON OUTPUT FROM DHPCG, SPECIFIES THE NUMBER OF BISECTIONS
C                    OF THE INITIAL INCREMENT THAT WERE NECESSARY.
C
C     OUTPUTS:       NONE
C
C     LOCAL:
C
C             NCOUNT INDICATES THE NUMBER OF TIMES OUTP HAS BEEN CALLED.
C                    NCOUNT ALLOWS ORDERLY PROGRAM STOPPING.
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
$INSERT DIMENSIONS.INS
C
      DIMENSION G(NDIM), GDER(NDIM), PRMT(5), PER(NPER)
C
      COMPLEX*16 CEIG(NA),CVECTL(NA,NA),CVECTR(NA,NA)
      COMPLEX*16 CEIGD2(NA,NGAIN,NGAIN)
C
      COMMON /CMAIN/ CEIG,CVECTL,CVECTR,CEIGD
C
      COMMON /CMAIN2/ CEIGD2
C
      COMMON /PERF/ PER, DET
C
      COMMON /OUT/ NCOUNT
C
C
```

```
      COMPLEX*16 CEIGD(NA,NGAIN)
C
C     NOTE:  I HAVE ASSUMED THAT NCOUNT IS ORIGINALLY SET TO ZERO.
C     THIS MAY NEED MODIFICATION IF THE PROGRAM IS PORTED.
C
C     WRITE(1,*)CEIG,CVECTR
      NCOUNT = NCOUNT + 1
C
      WRITE(1,10)TAU,IHLF,NCOUNT,(G(I),I=1,NDIM)
10    FORMAT(/'TAU=',D16.7,'  IHLF=',I16,'  NCOUNT=',I4
     1       /'GAINS:',4(/2D20.7))
C
C     NOW PRINT OUT SOME INTERESTING VALUES.
C
            WRITE(1,60)TAU,PER,DET
60          FORMAT(//' AT TAU=',D16.7,' TOTALS=',2D16.7/
     1                '    DET=   ',D16.7/)
C
C
            DO 70 I=1,NA/2,2
70          WRITE(1,*)CEIG(I)
C
C     NOW HALT EXECUTION IF NCOUNT IS LARGE.
C
      IF( NCOUNT .EQ. 500 ) STOP
C
      RETURN
      END
```

Appendix D

General Routines Listings

```
      SUBROUTINE STRUCT( A, TEMPM, ALEN, NA, NELEM )
C
C     SUBROUTINE STRUCT SETS UP THE MATRICES A AND ADER FOR THE
C     EIGENVALUE PROBLEM
C
C           -    -              -   -
C          |  X'' |            |  X' |
C          | ---  |    =   A   | ---  |
C          |  X'  |            |  X  |
C           -    -              -   -
C
C     THE EIGENVALUE PROBLEM
C
C               2
C              D X
C          M   ---2  +  K  X  = 0
C              D T
C
C     IS USED AS AN INTERMEDIATE FORM.
C
C     THIS SUBROUTINE MODELS, USING FINITE ELEMENTS, A BAR IN BENDING WITH
C     ONE END FIXED.
C
C           /|
C           /|
C           /|-------------------------------------------------
C           /|   |    |    |    ...                        |    |
C           /|-------------------------------------------------
C           /|   1    2    3                          NELEM-1  NELEM
C           /|
C
C     THERE ARE NELEM ELEMENTS AND NELEM JOINTS, AS SHOWN IN THE ABOVE
C     PICTURE.  THE EVEN NUMBERED DEGREES OF FREEDOM REFER TO THE JOINT
C     DISPACEMENTS.  THE ODD NUMBERED DOF REFER TO THE JOINT ANGLES.
C     POSITIVE IS UP AND COUNTER-CLOCKWISE.
C
C     REFERENCE:      ELEMENTS OF VIBRATIONAL ANALYSIS, LEONARD MEIROVITCH,
C                     1975, MCGRAW HILL.
C
C     INPUTS:
C
C            NA       DIMENSION OF A AND ADER.  THIS MUST BE SET TO
C                     4*NELEM BY A PARAMETER STATEMENT IN THE MAIN PROGRAM.
C
C            NELEM    NUMBER OF ELEMENTS IN THE FINITE ELEMENT MODEL.
C                     AS CURRENTLY CONFIGURED, THIS SUBROUTINE WILL ALLOW
C                     ONLY NELEM < 21.
C
C     OUTPUTS:
C
C            A        REAL*8 MATRIX DIMENSIONED NA BY NA WHICH CONTAINS THE
C                     MATRIX FOR THE EIGENVALUE PROBLEM.
C
C            TEMPM    REAL*8 MATRIX DIMENSIONED NA BY NA WHICH CONTAINS, ON
C                     OUTPUT, THE INVERSE OF THE TEMPM MATRIX.  THIS IS USED
C                     FOR CALCULATING THE DERIVATIVE OF A WRT GAINS IN
C                     SUBROUTINE STRDER.
C
C            ALEN     LENGTH OF AN ELEMENT.  THIS INFORMATION IS NEEDED IN
```

```
C                    SUBROUTINE STRDER.
C
C     LOCAL: (THIS IS RUNNING ON A VIRTUAL MEMORY MACHINE, SO THERE IS NO
C             NEED TO WORRY ABOUT SPACE IN THIS SUBROUTINE.)
C
C            AM       REAL*8 MATRIX DIMENSIONED 40 BY 40 WHICH IS USED TO
C                     SAVE THE M MATRIX LOCALLY.
C
C            AK       REAL*8 MATRIX DIMENSIONED 40 BY 40 WHICH IS USED TO
C                     SAVE THE K MATRIX LOCALLY.
C
C            AMASS    REAL*8 CONSTANT MASS PER UNIT LENGTH OF THE BAR.  THE BAR
C                     IN THIS SUBROUTINE IS UNIFORM. (KG/M)
C
C            EI       REAL*8 CONSTANT BENDING STIFFNESS OF THE BAR.  THE BAR
C                     IN THIS SUBROUTINE IS UNIFORM.
C
C            ALEN     REAL*8 LENGTH OF THE BAR. (METERS)
C
C            SMALLM   REAL*8 DIMENSIONED 4 BY 4 WHICH CONTAINS THE ELEMENT
C                     MASS MATRIX.
C
C            SMALLK   REAL*8 DIMENSIONED 4 BY 4 WHICH CONTAINS THE ELEMENT
C                     STIFFNESS MATRIX.
C
C            NROW     INTEGER INDEX USED TO INDICATE A ROW OF SMALLK AND SMALLM.
C
C            NCOL     INTEGER INDEX USED TO INDICATE A COLUMN OF SMALLK AND M.
C
C            TEMPK    REAL*8 DIMENSIONED 80 BY 80 USED TO STORE THE FIRST
C                     ORDER FORM OF THE EIGENVALUE PROBLEM.
C
C            DERM     REAL*8 DIMENSIONED 80 BY 80 USED TO STORE THE INTERMEDIATE
C                     MATRIX IN THE ADER CALCULATION.
C
C            IPVT     INT VECTOR DIMENSIONED 80 AND USED BY LINPACK.
C
C            WORK     REAL*8 VECTOR DIMENSIONED 80 AND USED BY LINPACK.
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C     REMEMBER TO ADJUST THE ARRAY SIZES IN THE LINPACK CALLS IF
C     THE DIMENSION OF TEMPM IS CHANGED.
C
      DIMENSION A(NA,NA),TEMPM(NA,NA),AK(40,40),AM(40,40)
      DIMENSION SMALLK(4,4),SMALLM(4,4),TEMPK(80,80)
      DIMENSION IPVT(80),WORK(80),DERM(80,80)
C
C     I AM GOING TO PUT SOME VARIABLES IN COMMON ONLY TO REDUCE THE SIZE OF
C     THE STACK.
C
      COMMON /STRU/ TEMPK,AK,AM,IPVT,WORK,DERM
C
C     DEFINE THE PARAMETERS FOR THE BAR.
C
               AMASS =  1.D0
               EI    =  0.1D0
               ALEN  =  1.D0/NELEM
C
C     USE THIS TO DEFINE SMALLK AND SMALLM.
```

```
C
C     FIRST SMALLM:
C
               SMALLM(1,1) = ( AMASS*ALEN/420.D0 ) * 156.D0
               SMALLM(1,2) = ( AMASS*ALEN/420.D0 ) * 22.D0*ALEN
               SMALLM(1,3) = ( AMASS*ALEN/420.D0 ) * 54.D0
               SMALLM(1,4) = ( AMASS*ALEN/420.D0 ) * (-13.D0*ALEN)
               SMALLM(2,1) = ( AMASS*ALEN/420.D0 ) * 22.D0*ALEN
               SMALLM(2,2) = ( AMASS*ALEN/420.D0 ) * 4.D0*ALEN**2
               SMALLM(2,3) = ( AMASS*ALEN/420.D0 ) * 13.D0*ALEN
               SMALLM(2,4) = ( AMASS*ALEN/420.D0 ) * (-3.D0*ALEN**2)
               SMALLM(3,1) = ( AMASS*ALEN/420.D0 ) * 54.D0
               SMALLM(3,2) = ( AMASS*ALEN/420.D0 ) * 13.D0*ALEN
               SMALLM(3,3) = ( AMASS*ALEN/420.D0 ) * 156.D0
               SMALLM(3,4) = ( AMASS*ALEN/420.D0 ) * (-22.D0*ALEN)
               SMALLM(4,1) = ( AMASS*ALEN/420.D0 ) * (-13.D0*ALEN)
               SMALLM(4,2) = ( AMASS*ALEN/420.D0 ) * (-3.D0*ALEN**2)
               SMALLM(4,3) = ( AMASS*ALEN/420.D0 ) * (-22.D0*ALEN)
               SMALLM(4,4) = ( AMASS*ALEN/420.D0 ) * 4.D0*ALEN**2
C
C     NOW SMALLK:
C
               SMALLK(1,1) = ( EI/ALEN**3 ) * 12.D0
               SMALLK(1,2) = ( EI/ALEN**3 ) * 6.D0*ALEN
               SMALLK(1,3) = ( EI/ALEN**3 ) * (-12.D0)
               SMALLK(1,4) = ( EI/ALEN**3 ) * 6.D0*ALEN
               SMALLK(2,1) = ( EI/ALEN**3 ) * 6.D0*ALEN
               SMALLK(2,2) = ( EI/ALEN**3 ) * 4.D0*ALEN**2
               SMALLK(2,3) = ( EI/ALEN**3 ) * (-6.D0*ALEN)
               SMALLK(2,4) = ( EI/ALEN**3 ) * 2.D0*ALEN**2
               SMALLK(3,1) = ( EI/ALEN**3 ) * (-12.D0)
               SMALLK(3,2) = ( EI/ALEN**3 ) * (-6.D0*ALEN)
               SMALLK(3,3) = ( EI/ALEN**3 ) * 12.D0
               SMALLK(3,4) = ( EI/ALEN**3 ) * (-6.D0*ALEN)
               SMALLK(4,1) = ( EI/ALEN**3 ) * 6.D0*ALEN
               SMALLK(4,2) = ( EI/ALEN**3 ) * 2.D0*ALEN**2
               SMALLK(4,3) = ( EI/ALEN**3 ) * (-6.D0*ALEN)
               SMALLK(4,4) = ( EI/ALEN**3 ) * 4.D0*ALEN**2
C
C     NOW CONTRUCT THE AM AND AK MATRICES.
C
C     FIRST ZERO THE MATICES.
C
          DO 10 I=1,NA/2
          DO 10 J=1,NA/2
C
             AM(I,J)=0.D0
             AK(I,J)=0.D0
C
10        CONTINUE
C
C     NOW ASSEMBLE THE ELEMENTS.  THIS IS DONE ELEMENT BY ELEMENT.
C
          DO 30 IELEM=1,NELEM
C
C            GO THROUGH SMALLM AND SMALLK WITH THE APPROPRIATE INDEX
C            SHIFTS.
C
             DO 20 I=1,4
             DO 20 J=1,4
```

```
C
C                PICK INDICES.
C
                 NROW=2*(IELEM-1)+I-2
                 NCOL=2*(IELEM-1)+J-2
C
C                PUT THE SMALLM AND SMALLK INTO AM AND AK.
C
                 IF( NROW .GT. 0 .AND. NCOL .GT. 0 ) THEN
C
                     AM(NROW,NCOL)=AM(NROW,NCOL)+SMALLM(I,J)
                     AK(NROW,NCOL)=AK(NROW,NCOL)+SMALLK(I,J)
C
                 ENDIF
C
20            CONTINUE
C
30          CONTINUE
C
C      NOW THAT THE ELEMENTS ARE ASSEMBLED, AM AND AK ARE WRITTEN INTO
C      THE APPROPRIATE POSITIONS IN A.
C
C         FIRST ZERO A, TEMPM AND TEMPK.
C
            DO 40 I=1,NA
            DO 40 J=1,NA
C
               A(I,J)=0.D0
               TEMPM(I,J)=0.D0
               TEMPK(I,J)=0.D0
C
40          CONTINUE
C
C         TEMPM AND TEMPA ARE USED TO CONVERT AM AND AK TO SECOND ORDER FORM.
C         AT THAT TIME, A IS CALCULATED BY
C
C                          -1
C               A  =  TEMPM    *  TEMPK
C
C         FIRST PUT AM INTO TEMPM.
C
               DO 50 I=1,NA/2
               DO 50 J=1,NA/2
50             TEMPM(I,J)=AM(I,J)
C
C         NOW PUT THE IDENTITY MATRIX INTO TEMPM.
C
               DO 60 I=NA/2+1,NA
60             TEMPM(I,I)=1.D0
C
C         NOW PUT AK INTO TEMPK.
C
               DO 70 I=1,NA/2
               DO 70 J=NA/2+1,NA
70             TEMPK(I,J)=-AK(I,J-NA/2)
C
C         NOW PUT THE IDENTITY MATRIX INTO TEMPK.
C
               DO 80 J=1,NA/2
C
```

```
                I=NA/2+J
                TEMPK(I,J)=1.D0
C
80          CONTINUE
C
C     TEMPM AND TEMPK ARE NOW BUILT.  WE NOW INVERT TEMPM USING TWO
C     SUBROUTINES FROM THE LINPACK PACKAGE.
C
          CALL DGEFA(TEMPM,NA,NA,IPVT,INFO)
          CALL DGEDI(TEMPM,NA,NA,IPVT,DET,WORK,1)
C
C     TEMPM SHOULD NOW CONTAIN THE INVERSE.  THE ONLY TASK NOW IS TO
C     MULTIPLY THIS INVERSE BY TEMPK.
C
            DO 100 I=1,NA
            DO 100 J=1,NA
C
              A(I,J)=0.D0
              DO 90 K=1,NA
90            A(I,J)=A(I,J)+TEMPM(I,K)*TEMPK(K,J)
C               .
100         CONTINUE
C
C     A IS NOW COMPLETE.  CALCULATION OF ADER WILL USE THE TEMPM MATRIX
C     WHICH CURRENTLY CONTAINS INV(TEMPM).
C
C     THAT'S ALL FOLKS.....
C
      RETURN
      END



      SUBROUTINE STRDER( ADER, NGAIN,TEMPM, NA, ALEN )
C
C     SUBROUTINE STRDER IS USED TO CALCULATE THE DERIVATIVE OF THE
C     A MATRIX WRT A GAIN NUMBER IGAIN.  IT USES THE MATRIX TEMPM
C     AND ELEMENT LENGTH ALEN FROM SUBROUTINE STRUCT.  REFER TO THE
C     DOCUMENTATION ON SUBROUTINE STRUCT TO CLEARLY DESCRIBE THE FUNCTION
C     OF TEMPM AND DERK.
C
C     IN SUBROUTINE STRUCT, THE SYSTEM
C
C              2
C            D X
C        M   --- 2  +  K   X   =   0
C            D T
C
C     IS REWRITTEN IN FIRST ORDER FORM, WITH
C
C                 -    -
C                |  X'  |
C        X   =   | ---  |
C                |  X   |
C                 -    -
C
C     AND
C
C              TEMPM  X'  =  TEMPK  X.
C
```

```
C     THEN
C             X'  =  A  X
C
C     WHEN
C
C             A  = INV ( TEMPM ) * TEMPK.
C
C     IF TEMPM IS CONSTANT, THEN
C
C             ADER  = INV( TEMPM ) * DERK.
C
C     SUBROUTINE STRUCT HAS ALREADY STORED INV(TEMPM) IN MATRIX TEMPM.
C     THIS SUBROUTINE USES THIS TO FIND ADER.
C
C     REFERENCE:   ELEMENTS OF VIBRATIONAL ANALYSIS, LEONARD MEIROVITCH,
C                  1975, KCGRAW HILL.
C
C     INPUTS:      TEMPM    REAL*8 MATRIX DIMENSIONED NA X NA WHICH
C                           CONTAINS THE INVERSE OF THE MATRIX DESCRIBED ABOVE.
C                           THIS MATRIX IS SUPPLIED BY SUBROUTINE STRUCT.
C
C                  NGAIN    THE NUMBER OF GAINS
C
C                  NA       DIMENSION OF TEMPM AND ADER.
C
C                  ALEN     LENGTH OF EACH ELEMENT.
C
C     OUTPUTS:     ADER     REAL*8 DIMENSIONED NA X NA X NGAIN WHICH CONTAINS
C                           THE PARTIAL DERIVATIVE OF A WRT THE GAIN VECTOR.
C
C     LOCAL:       DERK     REAL*8 DIMENSIONED 80 X 80 WHICH IS USED TO
C                           SAVE THE DERIVATIVE OF TEMPK.
C
C                  IFORCE   INTEGER DIMENSIONED (100) WHICH INDICATES WHICH
C                           ELEMENT IS UNIFORMLY FORCED BY THE GAIN.
C
C                  ISTATE   INTEGER DIMENSIONED (100) WHICH IS USED TO
C                           INDICATE WHICH STATE IS USED FOR A GAIN FEEDBACK.
C                           THE ISTATE MUST BE CHOSEN CAREFULLY.  NOTE THAT
C                           THE STATES ARE STACKED WITH RATES FIRST AND
C                           POSITIONS SECOND.  ALSO RECALL THAT JOINT
C                           DISPLACEMENTS HAVE ODD STATE AND JOINT ANGLES
C                           HAVE EVEN STATE.
C
C    ISTATE #1 IS THE JOINT DISPLACEMENT RATE FOR THE END OF ELEMENT 1.
C    ISTATE #2 IS THE JOINT ANGULAR      RATE FOR THE END OF ELEMENT 1.
C
C    ISTATE #3 IS THE JOINT DISPLACEMENT RATE FOR THE END OF ELEMENT 2.
C    ISTATE #4 IS THE JOINT ANGULAR      RATE FOR THE END OF ELEMENT 2.
C
C    ISTATE #5 IS THE JOINT DISPLACEMENT RATE FOR THE END OF ELEMENT 3.
C    ISTATE #6 IS THE JOINT ANGULAR      RATE FOR THE END OF ELEMENT 3.
C
C          ...
C
C    ISTATE #NA/2+1 IS THE JOINT DISPLACEMENT  FOR THE END OF ELEMENT 1.
C    ISTATE #NA/2+2 IS THE JOINT ANGULAR       FOR THE END OF ELEMENT 1.
C
C    ISTATE #NA/2+3 IS THE JOINT DISPLACEMENT  FOR THE END OF ELEMENT 2.
C    ISTATE #NA/2+4 IS THE JOINT ANGULAR       FOR THE END OF ELEMENT 2.
```

```
C
C     ISTATE #NA/2+5 IS THE JOINT DISPLACEMENT  FOR THE END OF ELEMENT 3.
C     ISTATE #NA/2+6 IS THE JOINT ANGULAR       FOR THE END OF ELEMENT 3.
C
C          ...
C
C      THE LIMITATION OF 100 ON THE DIMENSION OF ISTATE AND IFORCE RESTRICTS
C      US TO 100 OR FEWER GAINS.
C
       IMPLICIT REAL*8 ( A-H, O-Z )
C
       DIMENSION ADER(NA,NA,NGAIN), TEMPM(NA,NA), DERK(80,80)
C
       DIMENSION ISTATE(100), IFORCE(100)
C
C      BUILD THE VECTORS ISTATE AND IFORCE TO REFLECT THE GAIN FEEDBACK.
C
C            GAIN 1 FEEDS BACK END POSITION ON THE LAST ELEMENT.
C
             ISTATE(1)=NA-1
             IFORCE(1)=NA/4
C
C            GAIN 2 FEEDS BACK END POSITION RATE ON THE LAST ELEMENT.
C
             ISTATE(2)=NA/2-1
             IFORCE(2)=NA/4
C
C            GAIN3 FEEDS BACK THIRD ELEMENT END POSITION ON THE THIRD ELEMENT.
C
             ISTATE(3)=NA/2+5
             IFORCE(3)=3
C
C            GAIN4 FEEDS BACK THIRD ELEMENT END POSITION RATE ON THE THIRD
C            ELEMENT.
C
             ISTATE(4)=5
             IFORCE(4)=3
C
C            GAIN5 FEEDS BACK SIXTH ELEMENT END POSITION ON THE SIXTH
C            ELEMENT.
C
             ISTATE(5)=NA/2+11
             IFORCE(5)=6
C
C
C            GAIN6 FEEDS BACK SIXTH ELEMENT END POSITION RATE ON THE SIXTH
C            ELEMENT.
C
             ISTATE(6)=11
             IFORCE(6)=6
C
C
C            GAIN7 FEEDS BACK NINTH ELEMENT END POSITION  ON THE NINTH
C            ELEMENT.
C
             ISTATE(7)=NA/2+17
             IFORCE(7)=9
C
C
C            GAIN8 FEEDS BACK NINTH ELEMENT END POSITION RATE ON THE NINTH
```

```
C           ELEMENT.
C
            ISTATE(8)=17
            IFORCE(8)=9
C
C
C           GAIN9 FEEDS BACK 3RD ELEMENT END POSITION ON THE LAST
C           ELEMENT.
C
            ISTATE(9)=NA/2+5
            IFORCE(9)=NA/4
C
C
C           GAIN10 FEEDS BACK 3RD ELEMENT END POSITION RATE ON THE LAST
C           ELEMENT.
C
            ISTATE(10)=5
            IFORCE(10)=NA/4
C
C
C           GAIN11 FEEDS BACK 6TH ELEMENT END POSITION ON THE LAST
C           ELEMENT.
C
            ISTATE(11)=NA/2+11
            IFORCE(11)=NA/4
C
C
C           GAIN12 FEEDS BACK 6TH ELEMENT END POSITION RATE ON THE LAST
C           ELEMENT.
C
            ISTATE(12)=11
            IFORCE(12)=NA/4
C
C
C           GAIN13 FEEDS 9TH BACK ELEMENT END POSITION ON THE LAST
C           ELEMENT.
C
            ISTATE(13)=NA/2+17
            IFORCE(13)=NA/4
C
C
C           GAIN14 FEEDS BACK 9TH ELEMENT END POSITION RATE ON THE LAST
C           ELEMENT.
C
            ISTATE(14)=17
            IFORCE(14)=NA/4
C
C
C           GAIN15 FEEDS BACK LAST ELEMENT END POSITION ON THE 3RD
C           ELEMENT.
C
            ISTATE(15)=NA-1
            IFORCE(15)=3
C
C
C           GAIN16 FEEDS BACK LAST ELEMENT END POSITION RATE ON THE 3RD
C           ELEMENT.
C
            ISTATE(16)=NA/2-1
            IFORCE(16)=3
```

```
C
C
C           GAIN17 FEEDS BACK 6TH ELEMENT END POSITION ON THE 3RD
C           ELEMENT.
C
            ISTATE(17)=NA/2+11
            IFORCE(17)=3
C
C
C           GAIN18 FEEDS BACK 6TH ELEMENT END POSITION RATE ON THE 3RD
C           ELEMENT.
C
            ISTATE(18)=11
            IFORCE(18)=3
C
C
C           GAIN19 FEEDS BACK 9TH ELEMENT END POSITION ON THE 3RD
C           ELEMENT.
C
            ISTATE(19)=NA/2+17
            IFORCE(19)=3
C
C
C           GAIN20 FEEDS BACK 9TH ELEMENT END POSITION RATE ON THE 3RD
C           ELEMENT.
C
            ISTATE(20)=17
            IFORCE(20)=3
C
C
C           GAIN21 FEEDS BACK LAST ELEMENT END POSITION ON THE 6TH
C           ELEMENT.
C
            ISTATE(21)=NA-1
            IFORCE(21)=6
C
C
C           GAIN22 FEEDS BACK LAST ELEMENT END POSITION RATE ON THE 6TH
C           ELEMENT.
C
            ISTATE(22)=NA/2-1
            IFORCE(22)=6
C
C
C           GAIN23 FEEDS BACK 3RD ELEMENT END POSITION ON THE 6TH
C           ELEMENT.
C
            ISTATE(23)=NA/2+5
            IFORCE(23)=6
C
C
C           GAIN24 FEEDS BACK 3RD ELEMENT END POSITION RATE ON THE 6TH
C           ELEMENT.
C
            ISTATE(24)=5
            IFORCE(24)=6
C
C
C           GAIN25 FEEDS BACK 9TH ELEMENT END POSITION ON THE 6TH
C           ELEMENT.
```

```
C
            ISTATE(25)=NA/2+17
            IFORCE(25)=6
C
C
C           GAIN26 FEEDS BACK 9TH ELEMENT END POSITION RATE ON THE 6TH
C           ELEMENT.
C
            ISTATE(26)=17
            IFORCE(26)=6
C
C
C           GAIN27 FEEDS BACK LAST ELEMENT END POSITION ON THE 9TH
C           ELEMENT.
C
            ISTATE(27)=NA-1
            IFORCE(27)=9
C
C
C           GAIN28 FEEDS BACK LAST ELEMENT END POSITION RATE ON THE 9TH
C           ELEMENT.
C
            ISTATE(28)=NA/2-1
            IFORCE(28)=9
C
C
C           GAIN29 FEEDS BACK 3RD ELEMENT END POSITION ON THE 9TH
C           ELEMENT.
C
            ISTATE(29)=NA/2+5
            IFORCE(29)=9
C
C
C           GAIN30 FEEDS BACK 3RD ELEMENT END POSITION RATE ON THE 9TH
C           ELEMENT.
C
            ISTATE(30)=5
            IFORCE(30)=9
C
C
C           GAIN31 FEEDS BACK 6TH ELEMENT END POSITION ON THE 9TH
C           ELEMENT.
C
            ISTATE(31)=NA/2+11
            IFORCE(31)=9
C
C
C           GAIN32 FEEDS BACK 6TH ELEMENT END POSITION RATE ON THE 9TH
C           ELEMENT.
C
            ISTATE(32)=11
            IFORCE(32)=9
C
C     I JUST LOOP THROUGH THE GAINS, BUILDING ADER(*,*,IGAIN) AT EACH PASS.
C
C           ZERO DERK.
C
            DO 120 I=1,NA
            DO 120 J=1,NA
120         DERK(I,J)=0.D0
```

```
C
      DO 150 IGAIN=1,NGAIN
C
C           NOW CALULATE DERK.  THIS MATRIX IS THE MATRIX SUCH THAT
C
C                       -1
C           ADER = TEMPM    * DERK
C
C           NOW DEFINE THE FEW NON-ZERO ELEMENTS.
C
               DERK( 2*(IFORCE(IGAIN)-2)+1,ISTATE(IGAIN) )= 1.D0/2.D0
               DERK( 2*(IFORCE(IGAIN)-2)+2,ISTATE(IGAIN) )= ALEN/12.D0
               DERK( 2*(IFORCE(IGAIN)-2)+3,ISTATE(IGAIN) )= 1.D0/2.D0
               DERK( 2*(IFORCE(IGAIN)-2)+4,ISTATE(IGAIN) )=-ALEN/12.D0
C
C           NOW MULTIPLY INV(TEMPM) BY DERK.  RECALL THAT INV(TEMPM) HAS
C           BEEN STORED IN TEMPM.
C
            DO 140 I=1,NA
            DO 140 J=1,NA
C
              ADER(I,J,IGAIN)=0.D0
              DO 130 K=1,NA
130           ADER(I,J,IGAIN)=ADER(I,J,IGAIN)+TEMPM(I,K)*DERK(K,J)
C
140         CONTINUE
C
C           NOW DEFINE THE RE-ZERO THE ELEMENTS OF DERK TO PREPARE
C           FOR THE NEXT IGAIN.
C
               DERK( 2*(IFORCE(IGAIN)-2)+1,ISTATE(IGAIN) )= 0.D0
               DERK( 2*(IFORCE(IGAIN)-2)+2,ISTATE(IGAIN) )= 0.D0
               DERK( 2*(IFORCE(IGAIN)-2)+3,ISTATE(IGAIN) )= 0.D0
               DERK( 2*(IFORCE(IGAIN)-2)+4,ISTATE(IGAIN) )= 0.D0
C
150   CONTINUE
C
C     THAT'S ALL FOLKS......
C
      RETURN
      END


      SUBROUTINE CONJID( ER, EI, V, CEIG, CVECT, IDENT, NA )
C
C     THIS SUBROUTINE SORTS THE REAL AND COMPONENTS OF THE EIGENVALUES
C     TO IDENTIFY WHICH PAIRS ARE COMPLEX CONGUGATES.  THE SUBROUTINE
C     ASSUMES THE EIGENVALUES WERE COMPUTED BY THE ORACLE SUBROUTINE EIGEN.
C     THE EIGENVECTORS ARE ALSO NORMALIZED SO THAT TRANSPOSE(VECT)*VECT=1
C     AS REQUIRED BY THE SECOND DERIVATIVE OF THE EIGENVALUE ROUTINE.
C
C     INPUTS:
C
C            NA        INT (>2) INDICATING THE DIMENSION OF THE OTHER VARIABLES.
C
C            ER        REAL*8 VECTOR DIMENSIONED NA WHICH CONTAINS THE REAL
C                      COMPONENTS OF THE EIGENVALUES.
C
C            EI        REAL*8 VECTOR DIMENSIONED NA WHICH CONTAINS THE IMAG.
```

```
C                       COMPONENTS OF THE EIGENVALUES.
C
C             V         REAL*8 ARRAY DIM NA BY NA WHICH CONTAINS THE EIGEN-
C                       VECTORS IN THE FORMAT OUTPUT BY THE ORACLS SUBROUTINE
C                       EIGEN.
C
C     OUTPUTS:
C
C             CEIG      COMPLEX*16 VECTOR DIMENSIONED NA WHICH CONTAINS THE
C                       EIGENVALUES.
C
C             CVECT     COMPLEX*16 ARRAY DIM NA BY NA WHICH CONTAINS THE
C                       EIGENVECTORS BY COLUMN.
C
C             IDENT     INTEGER VECTOR DIMENSIONED NA WHICH CONTAINS THE CODE
C                       FOR LOCATING THE EIGENVECTORS IN THE EIGENVECTOR MATRIX
C                       V.
C
C             VECMAG    COMPLEX*16 USED TO NORMALIZE THE EIGENVECTORS.
C
C                     IDENT(I)=0 => LAMBDA(I)=0
C                     IDENT(I)=1 => LAMBDA(I) COMPLEX, FIRST OF TWO.
C                     IDENT(I)=2 => LAMBDA(I) COMPLEX, SECOND OF TWO.
C
C                     IDENT(I)=0 => EIGENVECTOR IN COLUMN I OF V.
C                     IDENT(I)=1 => REAL PART OF EIGENVECTOR IN COLUMN I OF V,
C                                   IMAG PART IN COLUMN I+1 OF V.
C                     IDENT(I)=2 => REAL PART OF EIGENVECTOR IN COLUMN I-1 OF V,
C                                   NEGATIVE OF IMAG PART IN COLUMN I OF V.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION IDENT(NA), ER(NA), EI(NA),V(NA,NA)
      COMPLEX*16 CEIG(NA),CVECT(NA,NA),VECMAG
C
C     I WILL BE USED AS A COUNTER TO RUN THROUGH THE VECTORS.
C
C     I DO THE FIRST ONE OUT OF THE LOOP SINCE I KNOW IT IS NOT A SECOND
C     IN A PAIR.
C
      IF( EI(1) .EQ. 0.D0 ) THEN
        IDENT(1)=0
      ELSE
        IDENT(1)=1
      ENDIF
C
C
      DO 10 I=2,NA
C
C
         IF( EI(I) .EQ. 0.D0 ) THEN
           IDENT(I)=0
         ELSE
           IDENT(I)=1
           IF( IDENT(I-1) .EQ. 1 ) IDENT(I)=2
         ENDIF
C
C
 10   CONTINUE
C
C     NOW IT IS VERY EASY TO THE EIGENVALUES AND VARIABLES INTO CEIG AND
```

```
C     CVECT.
C
      DO 100 I=1,NA
C
         CEIG(I)=DCMPLX(ER(I),EI(I))
C
C        CHECK FOR PAIRS AND BEHAVE APPROPRIATELY.
C
         IF( IDENT(I) .EQ. 0 ) THEN
C
C             EIGENVALUE REAL.
C
              DO 20 J=1,NA
20            CVECT(J,I)=DCMPLX(V(J,I),0.D0)
C
         END IF
C
         IF( IDENT(I) .EQ. 1 ) THEN
C
C             EIGENVALUE FIRST IN A COMPLEX CONJUGATE PAIR.
C
              DO 30 J=1,NA
30            CVECT(J,I)=DCMPLX(V(J,I),V(J,I+1))
C
         END IF
C
         IF( IDENT(I) .EQ. 2 ) THEN
C
C             EIGENVALUE SECOND IN A COMPLEX PAIR.
C
              DO 40 J=1,NA
40            CVECT(J,I)=DCMPLX(V(J,I-1),-V(J,I))
C
         END IF
C
100   CONTINUE
C
C     NOW NORMALIZE THE EIGENVECTORS.
C
C     GO THROUGH EACH EIGENVECTOR.
C
      DO 500 IVECT=1,NA
C
         VECMAG = DCMPLX( 0.D0, 0.D0 )
C
         DO 200 JROW=1,NA
200      VECMAG= VECMAG + CVECT( JROW, IVECT )**2
C
         VECMAG = VECMAG **(1.D0/2.D0)
C
         DO 400 JROW=1,NA
400      CVECT(JROW,IVECT) = CVECT(JROW,IVECT) / VECMAG
C
500   CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE EIGDER(CEIG,CVECTL,CVECTR,IDENT,ADER,CEIGD,NA,NGAIN)
C
C     SUBROUTINE EIGDER RETURNS THE DERIVATIVES OF THE EIGENVALUES
C     WITH RESPECT TO SOME PARAMETER OF THE MATRIX A.
C
C     INPUTS:
C
C           CEIG    COMPLEX*16 VECTOR OF DIMENSION NA WHICH CONTAINS THE
C                   EIGENVALUES.
C
C           CVECTL  COMPLEX*16 MATRIX OF DIMENSION NA BY NA WHICH CONTAINS THE
C                   LEFT EIGENVECTORS STORED BY COLUMNS.  THESE ARE ORDERED
C                   AS SPECIFIED BY IDENT.
C
C           CVECTR  COMPLEX*16 MATRIX OF DIMENSION NA BY NA WHICH CONTAINS THE
C                   RIGHT EIGENVECTORS STORED BY COLUMNS.  THESE ARE ORDERED
C                   AS SPECIFIED BY IDENT.
C
C           IDENT   INT VECTOR OF DIMENSION NA WHICH CONTAINS THE CODES TO
C                   IDENTIFY THE REDUNDANT EIGENVECTORS AND VALUES.
C
C           ADER    REAL*8 MATRIX OF DIM NA BY NA BY NGAIN WHICH CONTAINS THE
C                   PARTIAL DERIVATIVE OF THE A MATRIX WITH RESPECT TO THE
C                   DIFFERENTIATING PARAMETER.
C
C           NA      INTEGER VARIABLE INDICATION THE DIMENSIONS AND
C                   EIGENVALUE PROBLEM SIZE.  THIS SUBROUTINE LIMITED TO
C                   N<101.
C
C           NGAIN   INTEGER VARIABLE WHICH IS THE NUMBER OF GAINS AND A
C                   DIMENSION OF MATRIX CEIGD.
C
C     OUTPUTS:
C
C           CEIGD   COMPLEX*16 VECTOR OF DIMENSION NA X NGAIN WHICH IS THE
C                   PARTIAL DERIVATIVE OF THE EIGENVALUES WITH RESPECT TO THE
C                   GAIN VECTOR.
C
C     LOCAL VARIABLES:
C
C           DENOM   COMPEX*16 VARIABLE WHICH IS THE DENOMINATOR OF THE
C                   DERIVATIVES.
C
C           ANUM    COMPEX*16 VARIABLE WHICH IS THE NUMERATOR OF THE
C                   DERIVATIVES.
C
C           TNUM     COMPLEX*16 VARIABLE USE AS TEMPORARY STORAGE.
C
      COMPLEX*16 CEIGD(NA,NGAIN),CEIG(NA),CVECTL(NA,NA),CVECTR(NA,NA),
     1           DENOM,ANUM,TNUM
C
      REAL*8 ADER(NA,NA,NGAIN)
      INTEGER IDENT(NA)
C
C     THIS SUBROUTINE CALCULATES THE DERIVATIVE USING THE FORMULA
C
C
C                                    T  D A
C                                L(I) [ ---------- ] R(I)
C          D  LAMDBA(I)                 D PARAMETER
```

```
C          ------------=   ---------------------------------
C          D PARAMETER                  T
C                                     L(I)  R(I)
C
C     WE NOW DO A BIG LOOP FOR EACH EIGENVALUE.
C
      DO 100 IEIG=1,NA
C
C       A LOT OF CALCULATIONS CAN BE SKIPPED IF THE EIGENVALUE IS THE
C       SECOND MEMBER OF A PAIR OF COMPLEX CONJUGATES.
C
        IF( IDENT(IEIG) .EQ. 2 ) THEN
C
            DO 5 IGAIN=1,NGAIN
5           CEIGD(IEIG,IGAIN)=DCONJG(CEIGD(IEIG-1,IGAIN))
C
        ELSE
C
C           FIRST SET THE NEEDED VARIABLES TO ZERO.
C
            DENOM=DCMPLX(0.D0,0.D0)
C
C           NOW CALCULATE THE DENOMINATOR ASSOCIATED WITH THIS EIGENVALUE.
C
            DO 10 I=1,NA
10          DENOM=DENOM+CVECTL(I,IEIG)*CVECTR(I,IEIG)
C
C           NOW LOOP THROUGH THE GAINS TO GET THE SENSITIVITY OF IEIG TO
C           EACH GAIN.
C
            DO 50 IGAIN=1,NGAIN
C
C              FIRST SET THE NEEDED VARIABLES TO ZERO.
C
               ANUM=DCMPLX(0.D0,0.D0)
C
C              THERE IS AN OUTER AND AN INNER LOOP FOR THE NUMERATOR.  ONLY AN
C              OUTER LOOP IS USED WITH THE DENOMINATOR.
C
               DO 30 I=1,NA
C
                  TNUM=DCMPLX(0.D0,0.D0)
C
                  DO 25 J=1,NA
25                TNUM=TNUM+DCMPLX(ADER(I,J,IGAIN),0.D0)*CVECTR(J,IEIG)
C
                  ANUM=ANUM+CVECTL(I,IEIG)*TNUM
C
30             CONTINUE
C
               CEIGD(IEIG,IGAIN)=ANUM/DENOM
C
50          CONTINUE
C
        END IF
C
100         CONTINUE
C
            RETURN
            END
```

```
      SUBROUTINE SECDER( NA, NGAIN, NLIMIT, ADER, CVECTL, CVECTR,
     1                   IDENT, A, CEIGD, CEIG, CEIGD2           )
C
C     SUBROUTINE SECDER IS USED TO CALCULATE THE SECOND DERIVATIVE OF THE
C     EIGENVALUES WITH RESPECT TO THE GAINS.
C
C                                 2
C                          PARTIAL  ( EIGENVALUE I )
C      CEIGD2(I,J,K)   =   ------------------------------------
C                          PARTIAL( GAIN J )  PARTIAL( GAIN K )
C
C     DERIVATIVES ARE ONLY CALCULATED FOR THE FIRST NLIMIT EIGENVALUES TO
C     SAVE COMPUTATION TIME, SINCE PROBABLY ONLY THE FIRST HALF OF THE
C     EIGENVALUES WILL BE USED.
C
C     THIS SUBROUTINE ALSO REQUIRES THE NORMALIZATION OF THE EIGENVECTORS
C     PERFORMED IN SUBROUTINE CONJID.
C
C                           T
C      ( RIGHT EIGENVECTOR ) ( RIGHT EIGENVECTOR )  =  1
C
C                           T
C      ( LEFT EIGENVECTOR  ) ( LEFT EIGENVECTOR  )  =  1
C
C     A NUMBER OF VARIABLES ARE PASSED DOWN THROUGH THE CALL.
C
C         NLIMIT      SPECIFIES THE NUMBER OF EIGENVALUES WHICH NEED SECOND
C                     DERIVATIVES.
C
C         ADER        REAL*8 DIMENSIONED (NA,NA,NGAIN) IN COMMON CONTAINS THE
C                     PARTIAL OF THE A MATRIX WRT THE GAINS.
C
C
C         CVECTL      COMPLEX*16 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                     COMPLEX LEFT EIGENVECTORS STORED BY COLUMN.
C
C         CVECTR      COMPLEX*16 DIMENSIONED (NA,NA)  WHICH CONTAINS THE
C                     COMPLEX RIGHT EIGENVECTORS STORED BY COLUMN.
C
C         IDENT       INTEGER*4 DIMENSIONED (NA) CONTAINS INFO USEFUL TO SORT
C                     THE EIGENVALUES AND EIGENVECTORS.
C
C                       IDENT(I)=0 => LAMBDA(I)=0
C                       IDENT(I)=1 => LAMBDA(I) COMPLEX, FIRST OF TWO.
C                       IDENT(I)=2 => LAMBDA(I) COMPLEX, SECOND OF TWO.
C
C                       IDENT(I)=0 => EIGENVECTOR IN COLUMN I OF V.
C                       IDENT(I)=1 => REAL PART OF EIGENVECTOR IN COLUMN I OF V,
C                                     IMAG PART IN COLUMN I+1 OF V.
C                       IDENT(I)=2 => REAL PART OF EIGENVECTOR IN COLUMN I-1 OF
V,
C
C         A           REAL*8 DIMENSIONED (NA,NA)   WHICH CONTAINS THE
C                     COEFFICIENT MATRIX FOR THE STRUCTURE WITH CURRENT GAINS
C                     INCLUDED.
C
C         CEIGD       COMPLEX*16 DIMENSIONED (NA,NGAIN) WHICH CONTAINS THE
```

```
C                   COMPLEX DERIVATIVES OF THE EIGENVALUES WRT THE GAINS.
C
C       CEIG        COMPLEX*16 DIMENSIONED (NA)   WHICH CONTAINS THE
C                   COMPLEX EIGENVALUES.
C
C       NA          INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                   NUMBER OF DEGREES OF FREEDOM.  ( MUST BE 4*NELEM  )
C
C       NGAIN       INTEGER PARAMETER IN THE INSERT FILE WHICH INDICATES THE
C                   NUMBER OF GAINS.
C
C    THE ROUTINE OUTPUT IS STORED IN CEIGD2.
C
C       CEIGD2      COMPLEX*16 DIMENSIONED (NEIG,NGAIN,NGAIN) CONTAINS THE
C                   SECOND DERIVATIVES OF THE FIRST NLIMIT EIGENVALUES WITH
C                   RESPECT TO THE GAINS.
C
C    A NUMBER OF COMPLEX VARIABLES ARE USED AS INTERMEDIATE CALCULATIONS.
C
C       C1          TRANSPOSE( L ) * R
C
C       C2          TRANSPOSE( DERIV( L WRT GK ) ) * R
C
C       C3          TRANSPOSE( L ) * DERIV( R WRT GK )
C
C       C4          TRANSPOSE( L ) * ADER(J) * R
C
C       C5          TRANSPOSE( DERIV( L WRT GK ) ) * ADER(J) * R
C
C       C6          TRANSPOSE( L ) * ADER(J) * DERIV( R WRT GK )
C
C       C7          TRANSPOSE( L ) * CTEMP3(*)
C
C       C8          TRANSPOSE( L ) * CTEMP5(*)
C
C       C9          TRANSPOSE( R ) * CTEMP4(*)
C
C       C10         TRANSPOSE( R ) * CTEMP6(*)
C
C       CTEMP1(*)   ADER(J) * R
C
C       CTEMP2(*)   ADER(J) * DERIV( R WRT GK )
C
C                   (NOTE BELOW THAT CTEMP1 AND CTEMP2 HAVE MAXIMUM
C                   DIMENSIONS OF 100, WHICH LIMITS THE MAXIMUM DIMENSION
C                   OF NA TO <= 100.)
C
C       CTEMP3(*)   TRANS( L ) * R
C
C       CTEMP5(*)   ( ADER(*,*,NEIG) - DERIV( NEIG WRT GAIN K ) * I ) * R
C
C       CTEMP6(*)   (TRANS( ADER(*,*,NEIG) - DERIV( NEIG WRT GAIN K ) * I)) * L
C
C       CLDER(*)    THE COMPLEX DERIVATIVE DERIV( L WRT GK )
C
C       CRDER(*)    THE COMPLEX DERIVATIVE DERIV( R WRT GK )
C
C       ACOF(*)        THE COEFFICIENTS OF THE DERIV( R WRT GK )
C
C       BCOF(*)        THE COEFFICIENTS OF THE DERIV( L WRT GK )
```

```
C
C          NEIG        THE EIGENVLAUE NUMBER FOR THE CURRENT DERIVATIVE
C
C          JGAIN       THE GAIN NUMBER FOR THE CURRENT DERIVATIVE
C
C          KGAIN       THE GAIN NUMBER FOR THE CURRENT DERIVATIVE
C
       IMPLICIT REAL*8 (A-H,O-Z)
C
       REAL*8 ADER(NA,NA,NGAIN), A(NA,NA)
C
       DIMENSION IDENT(NA)
C
       COMPLEX*16 CVECTL(NA,NA),CVECTR(NA,NA),CEIG(NA),CEIGD(NA,NGAIN)
C
       COMPLEX*16 C1,C2,C3,C4,C5,C6,C8,C10
C
       COMPLEX*16 CTEMP1(100),CTEMP2(100)
C
       COMPLEX*16 CTEMP3(100), CTEMP5(100), CTEMP6(100)
C
       COMPLEX*16 CEIGD2(NA,NGAIN,NGAIN),CRDER(100),CLDER(100)
C
       COMPLEX*16 ACOF(100),BCOF(100)
C
C      THE CALCULATION OF THE SECOND DERIVATIVE REQUIRES THE CALCULATION OF
C      SOME INTERMEDIATE VARIABLES.  THESE HAVE BEEN LUMPED INTO THE SAME LOOPS
C      TO SAVE CPU TIME.
C
C      DO CALCULATIONS FOR THE FIRST NLIMIT EIGENVALUES.
C
C      I CAN CALCULATE CTEMP3 NOW TO SAVE COMPUTATIONS.
C
       DO 10 I=1,NA
C
          CTEMP3(I)=DCMPLX(0.D0,0.D0)
C
          DO 10 J=1,NA
C
          CTEMP3(I)=CTEMP3(I)+CVECTR(J,I)*CVECTL(J,I)
C
10     CONTINUE
C
       DO 1000 NEIG=1,NLIMIT
C
C           NOW CHECK FOR COMPLEX CONJUGATES.
C
       IF( IDENT( NEIG ) .NE. 2 ) THEN
C
C           FIRST I WILL CALCULATE THE INTERMEDIATE C1.
C
            C1=DCMPLX(0.0D0,0.0D0)
C
            DO 20 I=1,NA
20          C1=C1+CVECTL(I,NEIG)*CVECTR(I,NEIG)
C
C       NOW I WILL DO THE KGAIN LOOP.  I FIRST NEED THE EIGENVECTOR DERIVATIVES.
C
        DO 900 KGAIN=1,NGAIN
C
```

```
C       FIRST I NEED TO CALCULATE THE DERIVATIVES OF THE EIGENVECTORS WITH
C       RESPECT TO THE KGAIN.  THESE VALUES ARE STORED IN CLDER AND CRDER.
C
C
             DO 30 I=1,NA
C
               CTEMP5(I) = - CEIGD(NEIG,KGAIN) * CVECTR(I,NEIG)
               CTEMP6(I) = - CEIGD(NEIG,KGAIN) * CVECTL(I,NEIG)
C
               DO 30 J=1,NA
C
                 CTEMP5(I) = CTEMP5(I) + ADER(I,J,KGAIN)*CVECTR(J,NEIG)
                 CTEMP6(I) = CTEMP6(I) + ADER(J,I,KGAIN)*CVECTL(J,NEIG)
C
30           CONTINUE
C
C            NOW I CAN FINALLY CALCULATE THE COEFFICIENTS.
C
             DO 35 I=1,NA
C
               IF( I .NE. NEIG ) THEN
C
                 C8  = DCMPLX( 0.D0, 0.D0 )
                 C10 = DCMPLX( 0.D0, 0.D0 )
C
C            THE INTERMEDIATES CTEMP3 AND CTEMP4 CAN ALSO BE CALCULATED
C            HERE TO SAVE COMPUTATIONS.
C
                 DO 33 J=1,NA
C
                   C8  = C8  + CTEMP5(J)*CVECTL(J,I)
                   C10 = C10 + CTEMP6(J)*CVECTR(J,I)
C
33               CONTINUE
C
                 ACOF(I) = -C8/(CTEMP3(I)*(CEIG(I)-CEIG(NEIG)))
                 BCOF(I) = -C10/(CTEMP3(I)*(CEIG(I)-CEIG(NEIG)))
C
               END IF
35           CONTINUE
C
C              NOW I CAN HANDLE I = NEIG.
C
               ACOF(NEIG) = DCMPLX(0.D0,0.D0)
               BCOF(NEIG) = DCMPLX(0.D0,0.D0)
C
               DO 38 I=1,NA
C
                 IF( I .NE. NEIG ) THEN
C
                   C11 = DCMPLX(0.D0,0.D0)
                   C12 = DCMPLX(0.D0,0.D0)
C
                   DO 37 J=1,NA
                   C11 = C11 + CVECTR(J,I)*CVECTR(J,NEIG)
37                 C12 = C12 + CVECTL(J,I)*CVECTL(J,NEIG)
C
                   ACOF(NEIG) = ACOF(NEIG) - ACOF(I) * C11
                   BCOF(NEIG) = BCOF(NEIG) - BCOF(I) * C12
C
```

```
                END IF
38            CONTINUE
C
C             FINALLY I CAN USE THESE COEFFICIENTS TO FIND THE DERIVATIVES OF THE
C             EIGENVECTORS.
C

              DO 39 I=1,NA
C
                CLDER(I) = DCMPLX(0.D0,0.D0)
                CRDER(I) = DCMPLX(0.D0,0.D0)
C
                DO 39 J=1,NA
C
                CRDER(I) = CRDER(I) + ACOF(J) * CVECTR(I,J)
                CLDER(I) = CLDER(I) + BCOF(J) * CVECTL(I,J)
C
39            CONTINUE
C
C
C           I CAN NOW CALCULATE C2 AND C3 SINCE THEY DONT DEPEND ON JGAIN.
C
            C2=DCMPLX(0.D0,0.D0)
            C3=DCMPLX(0.D0,0.D0)
C
            DO 40 I=1,NA
            C2=C2+CLDER(I)*CVECTR(I,NEIG)
40          C3=C3+CVECTL(I,NEIG)*CRDER(I)
C
C       NOW I WILL DO THE JGAIN LOOP.
C
        DO 900 JGAIN=1,KGAIN
C
C           I HAVE USED THE SYMMETRY OF THE DERIVATIVE TO REDUCE THE RANGE
C           OF JGAIN.
C
C           NOW I AM FREE TO CALCULATE C4, C5 AND C6.  CTEMP1 AND CTEMP2
C           ARE USED AS INTERMEDIATES.
C
            DO 60 I=1,NA
C
               CTEMP1(I)=DCMPLX(0.D0,0.D0)
               CTEMP2(I)=DCMPLX(0.D0,0.D0)
C
               DO 60 J=1,NA
C
                CTEMP1(I)=CTEMP1(I)+ADER(I,J,JGAIN)*CVECTR(J,NEIG)
                CTEMP2(I)=CTEMP2(I)+ADER(I,J,JGAIN)*CRDER(J)
C
60          CONTINUE
C
            C4=DCMPLX(0.D0,0.D0)
            C5=DCMPLX(0.D0,0.D0)
            C6=DCMPLX(0.D0,0.D0)
C
            DO 80 I=1,NA
C
               C4=C4+CVECTL(I,NEIG)*CTEMP1(I)
               C5=C5+CLDER(I)*CTEMP1(I)
               C6=C6+CVECTL(I,NEIG)*CTEMP2(I)
```

```
C
80          CONTINUE
C
C       NOW THE DERIVATIVE CAN FINALLY BE EVALUATED.
C
        CEIGD2(NEIG,KGAIN,JGAIN)= -C4*(C2+C3)/C1**2+(C5+C6)/C1
C
C       THEN, BY SYMMETRY,
C
        CEIGD2(NEIG,JGAIN,KGAIN)= CEIGD2(NEIG,KGAIN,JGAIN)
C
C
900     CONTINUE
C
      ELSE
C
C     I EXECUTE THIS IF IDENT(NEIG)=2 AND ALL I NEED IS TO USE THE
C     COMPLEX CONJUGATE.
C
          DO 950 I=1,NGAIN
          DO 950 J=1,NGAIN
950       CEIGD2(NEIG,I,J)=DCONJG(CEIGD2(NEIG-1,I,J))
C
      END IF
C
1000  CONTINUE
      RETURN
      END
```

The vita has been removed from the scanned document

DESIGN OF ROBUST FEEDBACK CONTROL LAWS FOR HIGH-DIMENSIONED SYSTEMS

by

James P. Dunyak

Committee Chairman: Scott L. Hendricks

Engineering Mechanics

(ABSTRACT)

The design of feedback control laws for discrete models of flexible structures is addressed. Two strategies are proposed. First, a parameter optimization method is used, in which the behavior of the controller is described by a performance measure and numerically optimized. A second method, based on continuation maps, allows several performance criterion to be met. These performance measures are quite general in nature; they may be functions of eigenvalues, eigenvectors, sensitivities, and other mission specific quantities. A model of a cantilevered flexible beam is developed and used to demonstrate capabilities and problems with the design methods.