FILTER FAILURE DETECTION FOR SYSTEMS WITH LARGE SPACE STRUCTURE DYNAMICS

Craig R. Carignan

June, 1982 SSL#1-82

(Under NASA Grant #NAG1-126)

SSL

SPACE SYSTEMS LABORATORY
DEPT. OF AERONAUTICS AND ASTRONAUTICS
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MA 02139

NASA CR-169,146

FILTER FAILURE DETECTION FOR SYSTEMS WITH LARGE SPACE STRUCTURE DYNAMICS

Craig R. Carignan

June, 1982 SSL#1-82

(Under NASA Grant #NAG1-126)

N82-29344

FILTER FAILURE DETECTION FOR SYSTEMS WITH LARGE SPACE STRUCTURE DYNAMICS

by

CRAIG RAYMOND CARIGNAN

Submitted to the Department of Aeronautics and
Astronautics on May 14, 1982 in partial
fulfillment of the requirements for the
Degree of Master of Science in
Aeronautics and Astronautics

ABSTRACT

A failure detection filter is applied to the detection of actuator and sensor failures on a free-free beam. Computer simulation tests are used to verify the filter design and study the effect of unmodeled modes on filter performance.

In actuator tests, the failure signal to spillover noise ratio was found to be greatest when the filter bandwidth was 5 rad/sec beyond the input frequency. Observation spillover, however, was found to vary widely in tests run under similar conditions (same input frequency and filter poles) but with different detector gains.

In sensor tests, the maximum signal-to-noise ratio for varying filter bandwidth depended upon the initial conditions placed on the unmodeled modes; the performance was good even for initial amplitudes on the first unmodeled mode 7.5% of that on the last modeled mode.

Data-sampling tests on filters designed for continuous data processing but employed in a sampled data mode revealed that adequate filter performance could be achieved only when the sampling rate was considerably beyond the natural frequency of the last system mode. Stability problems were encountered when the filter bandwidth became too high relative to the sampling rate.

The failure simulation tests suggest high sampling rates and sensor post-filtering to deal with the problems posed by sampling phase lag and observation spillover.

Thesis Supervisor: Wallace E. Vander Velde
Professor of Aeronautics and Astronautics

ACKNOWLEDGMENTS

This thesis is not the result of one person's work, but rather the result of advice and help from many people. In view of this, I would like to express my gratitude and thanks to the following people: my thesis advisor, Professor Wally Vander Velde, for his expert help and guidance throughout this thesis, Professor Rene Miller for his advice and encouragement in this and previous work, and Barbara for typing this thesis. My acknowledgments would not be complete without thanking my friends in the Space Systems Lab and my family for their encouragement and support.

I would also like to express my appreciation to the National Aeronautics and Space Administration for sponsoring this work under NASA Grant #NAG1-126, "Reliability Issues in Active Control of Flexible Space Structures."

Table of Contents

Page

1. Introduction 7

2. Detection Filter Theory 11

2.1 Detection filter structure 11

2.2 Failure models 14

2.3 Detection filter design 17

2.3.1 Fully measurable systems 18

2.3.2 Partially measurable systems 19

2.3.2.1 Detection generator 20

2.3.2.2 Detector gain 22

2.3.2.3 Detection space 24

2.3.3 Sensor detectability 26

2.3.4 Sets of events 30

2.3.4.1 Actuator set 31

2.3.4.2 Sensor set 31

2.4 Two-mode Design Example 33

3. Computational Design of Filter 38

3.1 Orthogonal Reduction 38

3.2 Input Failure Event Design 41

3.2.1 Subroutine SEPDET 42

3.2.2 Subroutine DETGEN 44

3.2.3 Subroutine DGAIN 46

3.3 Measurement failure event design 48

Page

4. Application to Flexible Beam 51

4.1 NASA LaRC Experimental Beam 51

4.2 State equations for beam model 53

4.3 Failure detection filter design for the beam 56

4.4 Computer simulation of beam 58

4.4.1 Effect of model error 61

4.4.2 Sampled-data systems 75

5. Summary and Conclusions 81

Appendix A: Failure Detection Filter program (FDFIL) 84

Appendix B: Beam Simulation Program (FDSIM) 99

References 105

List of Illustrations Page

2.1 Failure detection filter block diagram 12

2.2 Eigenvalue assignment for detection filter 12

3.1 Flowchart for ORTRED 39

3.2 Flowchart for SEPDET 43

3.3 Flowchart for DETGEN 45

3.4 Flowchart for DGAIN 47

3.5 Sensor design process schematic 49

4.1 LaRC experimental beam set-up 52

4.2 SPAR beam modal frequencies and shapes 54

4.3 Output error transformation for sensor case 60

4.4 Sampled-data system/filter 60

4.5 Actuator failure for matched models 64

4.6 Model error effect for different filter models 65

4.7 Actuator tests for $\lambda^{+}=-10$ and $\omega_{u}^{+}=5,20,50$ 68

4.8 Actuator tests for $\lambda=-15$ and $\omega_{u}=5,20,50$ 69

4.9 Actuator tests for $\lambda=-20$ and $\omega_{u}=5,20,50$ 70

4.10 Sensor tests for various initial conditions 73

4.11 Sensor tests for various filter bandwidths 74

4.12 Data sampling tests for various sampling rates 78

4.13 Data sampling tests for various filter bandwidths 79

+λ denotes filter poles (rad/sec)

ω_{u} denotes input frequencies (rad/sec)

CHAPTER I

INTRODUCTION

With the advent of the space shuttle, aerospace engineers are contemplating the assembly and deployment of some very large space structures. Some structures under consideration include antennas and reflectors 100 meters in diameter and solar power satellites as large as 20 x 10 kilometers. Unlike the spacecraft of previous decades, these large structures have little inherent rigidity due to their low mass and large size. If the natural damping is not somehow increased, periodic disturbances such as gravity gradient and solar pressure which are close to the low natural frequencies of the structure will cause large dynamic overstresses that will eventually tear the structure apart.

The solar power satellite provides a good example of the types of overwhelming issues one would typically encounter in designing a control system for a large space structure. In order to adequately damp the many vibrational modes of the satellite, hundreds of thrusters and control moment gyros may be required to supplement passive damping. The system designer will have to decide how many actuators and sensors to use and where to place them on the structure. For example, rate gyro sensors and control moment gyros could be located almost anywhere on a truss-like structure. The control engineer will then have to decide what kind of control law to implement in order to maintain satisfactory structural rigidity. Obviously, the control system cannot incorporate all the structural modes in its model, so care must be taken when controlling the disturbance-induced vibrations in the low frequency modes that the control does not spillover into the higher frequency unmodeled modes.

One factor which should not be overlooked in either the design or operation of the control system is the likelihood of some failures among actuators and sensors. For example, if the interval between maintenance visits is three years and the control system utilizes a total of 400 sensors and actuators each with an exponential distribution of time to failure with a mean time to failure of 100,000 hours, the expected number of failures in this interval is 92, and the probability that there will be no failures is 2×10^{-46}. Thus even with a very optimistic mean time to failure, it is virtually certain that failures will occur.

One of the major issues in dealing with component unreliability in control systems is how to detect a failure and identify the failed component. This thesis is concerned with one method of doing failure detection and identification (FDI).

Many approaches to FDI have been used, the simplest of which involves triplication of components: a discrepancy between the signals of two like sensors signifies a failure, and comparison with the third determines which of the two has failed. Though simple, this method rapidly becomes costly and even bulky for certain applications.

There are several approaches to FDI which require specification of failure modes ahead of time, but one which does not is generalized parity relations.[5] This method uses sensor data from several time steps to detect failures rather than data from duplicate sensors at the same time instant. This approach has the obvious advantage of requiring fewer components, but it turns out to be very susceptible to plant disturbances and sensor noise. This detection routine also performs poorly when there is model error present, whether it be in the form of modal truncation or frequency errors.

A closed-loop method, the failure detection filter, can simultaneously monitor many different types of components, including sensors, actuators, and dynamic elements of the system. As with any other observer, the detection filter incorporates a linear-dynamic model of the system to estimate the true states of the system. Since the model receives the same control inputs as the true system, the outputs of the system and filter will normally match resulting in an output error of zero. However, when a component fails, the output error will no longer be zero, signifying that a failure has occurred. The failed component can then be identified by the fixed line or plane to which the output error is restricted by the detection filter.

The failure detection filter was first proposed by

Beard (1) in 1971 for deterministic systems. The theory was later expanded by Jones (2) to stochastic and sampled-data systems. Though not strictly valid for sampled-data systems, the detection filter will behave satisfactorily for sufficiently high sampling rates. Besides the application by Jones to a lateral mode autopilot, the failure detection filter was applied by VanderVelde (8) and Gerard (9) to the computer control of a guideway vehicle, and by Meserole (3) to fault-tolerant control of a turbofan engine. In neither of these previous applications was the filter designed to detect a sensor failure when the sensor output was not measuring a state directly. This is also the first time model error has been introduced into the filter.

The next chapter summarizes the main concepts of failure detection theory along with an analytic design procedure for the filter. Chapter 3 proposes a computational design procedure for the deterministic filter based mainly upon algorithms suggested in Appendix A of Beard (1). In Chapter 4, simulation results of actuator and sensor failures in deterministic systems are presented along with some results on data-sampling. Finally, some conclusions are outlined in Chapter 5.

CHAPTER II

FAILURE DETECTION FILTER THEORY

Detection filter theory is based upon vector-space concepts involving the state estimation errors generated by the filter following component failures. The major feature of the failure detection filter is that the output error is small while the system is functioning normally, and following the failure of a system component that error is significantly larger and appears only in a single direction or plane—that direction or plane indicating which component has failed. Thus the filter provides the basis for both detection of component failures and isolation of the faulty component. It is not necessary to specify in advance the possible modes of component failures.

In this chapter, the structure of the failure detection filter is first presented along with failure models for both actuator and sensor malfunctions and plant dynamics changes. The concept of failure "detectability" will then be introduced followed by the filter design theory for both fully measurable systems (rank C=n) and partially measurable systems (rank C<n). Finally, a two-mode beam example is used to illustrate the analytic design procedure.

2.1 Detection Filter Structure

The structure of the failure detection filter depicted in Fig. 2.1 is the same as that of any other state estimation

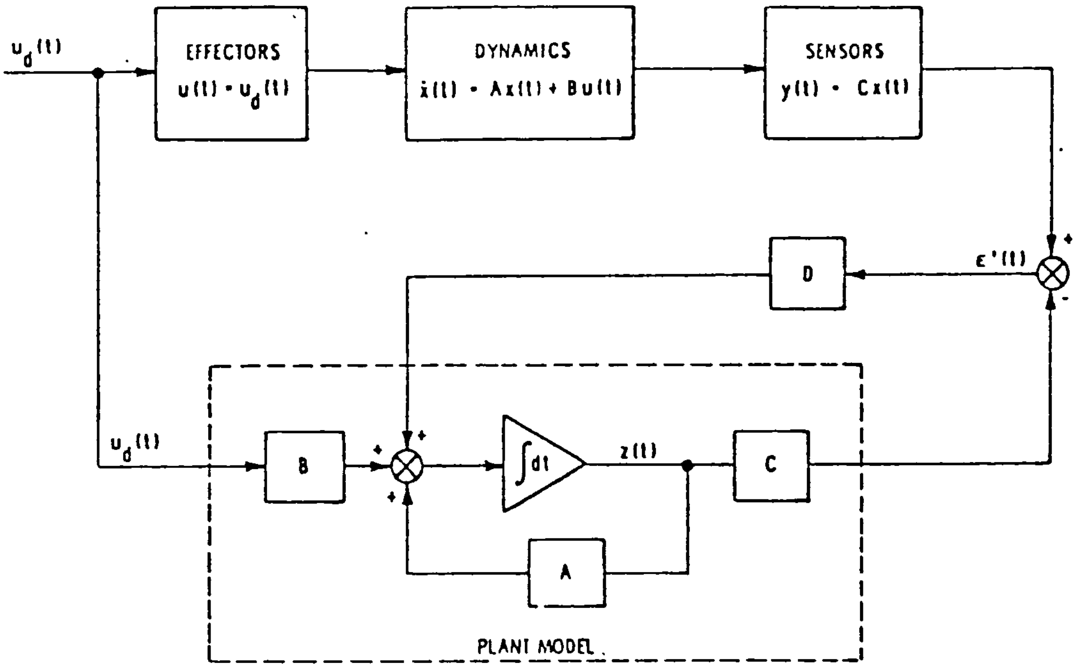


Fig. 2.1: Failure detection filter block diagram.
(taken from reference[1])

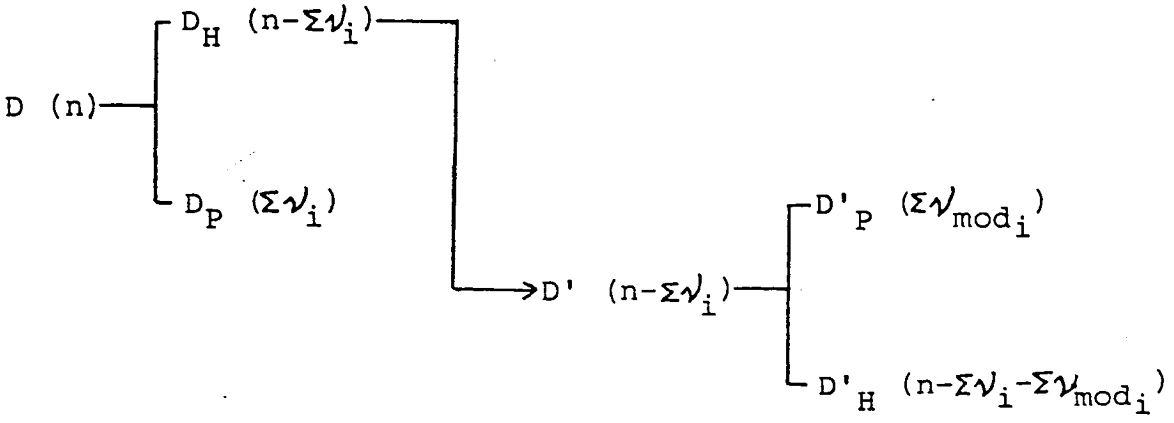


Fig. 2.2: Eigenvalue assignment for detection filter
(number in parentheses indicates the number of
eigenvalues of [A-DC] assigned with that gain).

filter monitoring a linear time invariant system. The difference is in the feedback gain D which is designed so that the output error will be held to a fixed direction or plane depending upon what component has failed. If D were the Kalman gain, the filter would minimize the mean square error between the states $\underline{x}(t)$ and the filter estimates $\hat{\underline{x}}(t)$ when noise is introduced into the system. For the case of an observer, $\hat{\underline{x}}(t)$ approaches $\underline{x}(t)$ asymptotically since no noise is present.

The system being monitored by the detection filter must be linear, time-invariant, and observable. The consequences of unobservability will be pointed out as they are encountered. In this work, mainly continuous inputs and measurements are considered.

The system is represented by the linear equations:

$$\begin{aligned} \dot{\underline{x}}(t) &= A\underline{x}(t) + B\underline{u}(t) \\ \underline{y}(t) &= C\underline{x}(t) \end{aligned} \qquad (2.1\text{-}1)$$

where the state vector $\underline{x}(t)$ has dimension n, the control vector $\underline{u}(t)$ has m inputs, and the measurement vector $\underline{y}(t)$ has p outputs.

The filter incorporates the model:

$$\begin{aligned} \dot{\hat{\underline{x}}}(t) &= A\hat{\underline{x}}(t) + B\underline{u}(t) + D[\underline{y}(t) - \hat{\underline{y}}(t)] \\ \hat{\underline{y}}(t) &= C\hat{\underline{x}}(t) \end{aligned} \qquad (2.1\text{-}2)$$

where $\hat{\underline{x}}$ is the filter state vector and $\hat{\underline{y}}(t)$ the predicted measurement vector.

The state and measurement error dynamics are

$$\begin{aligned} \dot{\underline{e}}(t) &= [A - DC]\,\underline{e}(t) \\ \underline{\epsilon}(t) &= C\,\underline{e}(t) \end{aligned} \qquad (2.1\text{-}3)$$

where $\underline{e}(t) \triangleq \underline{x}(t) - \hat{\underline{x}}(t)$ and $\underline{\epsilon}(t) = \underline{y}(t) - \hat{\underline{y}}(t)$. If the eigenvalues of A-DC have negative real parts, the steady state error will be zero so $\hat{\underline{x}}(t)$ will track $\underline{x}(t)$. Equation 2.1-3 is valid, however, only when $\{A,B,C\}$ of the filter in (2.1-2) match that of the true system represented by (2.1-1).

2.2 Failure Models

As mentioned above, (2.1-3) is valid only when the triplets $\{A,B,C\}$ characterizing the system and filter match. If there is a component failure or change in plant dynamics, (2.1-1) will no longer represent the true system.

Consider an actuator j that has either failed completely or gone awry. The control vector, $\underline{u}(t)$, is given by

$$\underline{u}(t) = \underline{u}_c(t) + \hat{\underline{e}}_{m_j} n(t) \qquad (2.2\text{-}1)$$

where $\underline{u}_c(t)$ is the commanded input (and filter input), $\hat{\underline{e}}_{m_j}$ is a unit vector in the j^{th} direction of dimension m, and n(t) is a scalar time function depending upon the type of failure. The new state equation becomes

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B\underline{u}_c(t) + \underline{b}_j n(t) \qquad (2.2\text{-}2)$$

where $\underline{b}_j$ is the column of B corresponding to the j^{th} actuator and $\eta(t)$ is some scalar time function. If the actuator fails completely in the off mode, $n(t)$ is simply minus the j^{th} input, $-u_{c_j}(t)$, and the control effectiveness matrix for the true system is the same as for the filter except that the j^{th} column of B is deleted.

Subtracting (2.1-2) from (2.2-2), the error equations become

$$\begin{aligned} \dot{\underline{e}}(t) &= [A-DC]\,\underline{e}(t) + \underline{b}_j n(t) \\ \underline{\epsilon}(t) &= C\,\underline{e}(t) \end{aligned} \tag{2.2-3}$$

The vector $\underline{b}_j$ is called the "event vector" because it is the driving vector in the event of failure of the j^{th} actuator. Detection filter theory will show that for any event vector $\underline{f}$, it is possible to find a D such that $\underline{\epsilon}(t)$ maintains a fixed direction in response to $\underline{f}n(t)$.

Similarly, consider a sensor failure represented by the new measurement relation

$$\underline{y}(t) = C\underline{x}(t) + \hat{\underline{e}}_{p_j} n(t) \tag{2.2-4}$$

where $\hat{\underline{e}}_{p_j}$ is a unit vector in the j^{th} direction and n(t) is an arbitrary scalar function of time. Complete failure of the j^{th} sensor in the zero-output mode is modeled by letting $n(t) = -\underline{c}_j^T\,\underline{x}(t)$, the predicted sensor output, and the measurement matrix for the true system is that of the filter with the j^{th} row deleted.

The new error equations are generated analogously with (2.2-4) replacing $\underline{y}(t)$ in (2.1-1):

$$\begin{aligned} \dot{\underline{e}}(t) &= [A-DC]\,\underline{e}(t) - \underline{d}_j n(t) \\ \underline{\epsilon}(t) &= C\,\underline{e}(t) + \hat{\underline{e}}_{p_j} n(t) \end{aligned} \tag{2.2-5}$$

Unlike the actuator case, the output error in the sensor failure mode will have two directions: $\hat{\underline{e}}_{p_j}$ due directly to the sensor failure and another component caused by the measurement error effect in the filter feedback loop. Even though D can be chosen to make $\underline{\epsilon}(t)$ due to $\underline{d}_j$ unidirectional, this direction will usually not be $\hat{\underline{e}}_{p_j}$, and therefore the output error will span a plane rather than lie in a fixed direction.

Finally, let us consider changes in plant dynamics. Let the ij th element of A, for instance, change by the amount Δa_{ij}. The new state equation is:

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B\underline{u}(t) + \Delta a_{ij} x_j(t)\, \hat{\underline{e}}_{n_i} \tag{2.2-6}$$

This model represents the effect of an alteration in the derivative of $x_i(t)$ due to dynamics involving $x_j(t)$. The error equations become

$$\dot{\underline{e}}(t) = [A-DC]\,\underline{e}(t) + \Delta a_{ij} x_j(t)\, \hat{\underline{e}}_{n_i} \tag{2.2-7}$$
$$\underline{\epsilon}(t) = C\,\underline{e}(t)$$

Comparing this with Eq. (2.2-3), we sée that the two pairs of equations have the same form with the scalar time function in this case being $\Delta a_{ij} x_j(t)$ and the new event vector as the n-dimensional i th unit vector.

Actuator and plant dynamics failure models are, in fact, called "input failure models" since both can be represented as extraneous additive inputs to the system. When the failure term also appears in the measurement equation [as in Eq. (2.2-5) for sensor failures], the error equations are referred to as the "measurement failure models." Only the actuator input and measurement failures will be considered in this work.

2.3 Detection Filter Design

The preceding discussion of failure models motivates the following definition of "detectability" found in Beard [1]: The event associated with the vector $\underline{f}$ in the state error equation

$$\dot{\underline{e}}(t) = [A-DC]\underline{e}(t) + \underline{f}\,n(t) \tag{2.3-1}$$

is "detectable" if there exists a gain matrix D such that:

(i) $C\underline{e}_p(t)$ maintains a fixed direction in the output space ($\underline{e}_p(t)$ is the settled-out solution of Eq. 2.3-1)

(ii) All the eigenvalues of (A-DC) can be arbitrarily specified (this condition may be relaxed for fixed but stable eigenvalues).

If $\underline{f}$ is detectable, then one can identify the failed component by checking the direction of the output error. For a measurement failure, the definition of detectability must be revised (see Section 2.3.3) and the output error is confined to a plane.

In the next several sections, the design of D for both fully measurable systems (rank C=n) and partially measurable systems (rank C<n) will be presented. Finding D for the first case is simple since the state vector $\underline{x}(t)$ can be solved uniquely given $\underline{y}(t)$. However, when the state vector is not fully measurable, the design of D is much more difficult and generates the need for a more advanced methodology.

2.3.1 Fully Measurable Systems

Let NS be the number of sensors or rows of C. To satisfy condition (ii) of detectability, choose A-DC to equal $-\sigma I$ where σ is a positive scalar constant. Then the eigenvalues of A-DC are all $-\sigma$ and D is given uniquely by

$$D = (A + \sigma I) C^{-1} \tag{2.3-2}$$

for rankC=n=NS and nonuniquely as

$$D = (A + \sigma I)(C^T C)^{-1} C^T \tag{2.3-3}$$

for rankC=n<NS.

For an actuator failure, $\underline{f} = \underline{b}_i$ and the solution of Eq. (2.2-3) is

$$\underline{e}(t) = e^{-\sigma(t-t_o)} \underline{e}(t_o) + \int_{t_o}^{t} e^{-\sigma(t-\tau)} \underline{b}_i n(\tau)\, d\tau \tag{2.3-4}$$

The first term on the right is due to the initial conditions of the filter and will asymptotically approach zero for $\sigma > 0$. The second term is the driven or settled solution—it comes about because the filter models the i^{th} actuator as working properly ($\underline{u}(t) = \underline{u}_c(t)$) whereas in actuality it is not.

Since $\underline{b}_i$ is constant, the settled-out solution is

$$\underline{e}_s(t) = \underline{b}_i \int_{t_o}^{t} e^{-\sigma(t-\tau)} n(\tau)\, d\tau \tag{2.3-5}$$

Therefore, $\underline{e}_s(t)$ lies in the direction $\underline{b}_i$ and a settled output error in the direction $C\underline{b}_i$ indicates a malfunction of the i^{th}

actuator.

Now consider a failure in the i^{th} sensor as modeled by (2.2-4). Retaining the same choice for A-DC, the solution to (2.2-5) is

$$\underline{e}(t) = e^{-\sigma(t-t_o)}\underline{e}(t_o) - D\hat{\underline{e}}_{pi}\int_{t_o}^{t} e^{-\sigma(t-\tau)} n(\tau)\, d\tau \qquad (2.3\text{-}6)$$

The settled output error is given by

$$\underline{\epsilon}_s(t) = -CD\hat{\underline{e}}_{pi}\int_{t_o}^{t} e^{-\sigma(t-\tau)} n(\tau)\, d\tau + \hat{\underline{e}}_{pi} n(t)$$

Therefore, an output error lying in the plane spanned by $CD\hat{\underline{e}}_{pi}$ and $\hat{\underline{e}}_{pi}$ indicates a failure of the i^{th} sensor.

Thus we see that finding D in the fully measurable case is indeed trivial and given by (2.3-2) for A-DC = $-\sigma$I. Choosing A-DC = $-\sigma$ I enabled the assignment of eigenvalues directly and also produced readily identifiable output error directions for component malfunctions because of the simple form of the state error transition matrix.

2.3.2 Partially Measurable Systems

Because C is not invertible in partially measurable systems, D cannot be solved by a simple relation like (2.3-2). What this means physically is that when an event occurs, an actuator failure for instance, the output error which results will not be uniquely associated with the state error caused by the event. Thus since the output error is the only accessible signal, D will not be unique in satisfying the conditions of detectability. It will therefore be necessary to explore the relation of D to the space spanned by state errors during an event.

2.3.2.1 Detection Generator

Consider the system equations (2.1-1) for the case of a single actuator:

$$\dot{\underline{x}}(t) = A\underline{x}(t) + \underline{b}_i u(t) \tag{2.3-7}$$

From linear systems theory, the controllable space of u(t), [the space of states which are possible to reach from the origin using u(t)] is the range space of

$$W_i \triangleq [\underline{b}_i \,\vdots\, A\underline{b}_i \,\vdots\, \cdots \,\vdots\, A^{n-1}\underline{b}_i] \tag{2.3-8}$$

The event vector $\underline{f}$ has a similar meaning with respect to the error space $\underline{e}(t)$ of (2.3-1). In this case, the error space that can be reached in the event of a failure is the range space of

$$W_f = [\underline{f} \,\vdots\, (A-DC)\underline{f} \,\vdots\, \cdots \,\vdots\, (A-DC)^{n-1}\underline{f}] \tag{2.3-9}$$

or the "detection space" of $\underline{f}$. The output error space therefore spans CW_f, and condition (i) of detectability is satisfied only if the rank of CW_f is 1. Any matrix D which satisfies this condition is called a "detector gain" for $\underline{f}$.

Associated with $\underline{f}$ is another vector also in W_f defined by the following theorem proven in Beard:

Detection generator theorem -

If (i) (A,C) is an observable pair

(ii) rank $W_f = k$

(iii) rank $CW_f = 1$

Then there exists an n-vector $\underline{g}$ in W_f such that

$$[\, C \,\vdots\, CA \,\vdots\, \cdots \,\vdots\, CA^{k-2}\,]\, \underline{g} = \underline{0} \tag{2.3-10}$$

and $$CA^{k-1}\underline{g} \neq \underline{0}$$

The latter relation in (2.3-10) guarantees that the controllable space of $\underline{g}$ with respect to (A-DC) is of dimension k and so matches the controllable space of $\underline{f}$. Note also that the first equation in (2.3-10) yields

$$\begin{aligned}&[\, \underline{g} \,\vdots\, (A-DC)\underline{g} \,\vdots\, \cdots \,\vdots\, (A-DC)^{k-1}\underline{g}\,] \\ &\quad = [\, \underline{g} \,\vdots\, A\underline{g} \,\vdots\, \cdots \,\vdots\, A^{k-1}\underline{g}\,]\end{aligned} \tag{2.3-11}$$

so the set of vectors $\{\underline{g}, A\underline{g}, \ldots, A^{k-1}\underline{g}\}$ form a basis for the controllable space of $\underline{f}$.

Since $\underline{f}$ is the generator of W_f, it can be expressed as

$$\underline{f} = \alpha_1 \underline{g} + \alpha_2 A\underline{g} + \ldots + \alpha_k A^{k-1}\underline{g} \tag{2.3-12}$$

where the α_i are scalars. Premultiplying by C and using (2.3-10),

$$C\underline{f} = \alpha_k CA^{k-1}\underline{g} \tag{2.3-13}$$

If $C\underline{f} \neq 0$, then $\alpha_k \neq 0$, and since the magnitude of $\underline{g}$ is not restricted by (2.3-10), we are free to choose $\alpha_k = 1$.

If in general, for some nonnegative integer μ,

$$CA^j\underline{f} = \underline{0} \qquad j = 0, \ldots, \mu-1 \tag{2.3-14}$$
$$CA^{\mu}\underline{f} \neq \underline{0}$$

then

$$\alpha_{k-j} = 0 \qquad \text{for} \quad j = 0, \ldots, \mu-1 \tag{2.3-15}$$
$$\alpha_{k-\mu} \neq 0$$

and $\underline{g}$ is chosen so that $\alpha_{k-\mu}=1$. The fact that (A,C) is an observable pair will guarantee that (2.3-14) holds for $\mu \leq k-1$. Using (2.3-15) in (2.3-12) yields

$$\underline{f} = \alpha_1 \underline{g} + \ldots + \alpha_{k-\mu-1} A^{k-\mu-2} \underline{g} + A^{k-\mu-1} \underline{g} \tag{2.3-16}$$

where μ satisfies (2.3-14).

The n-vector $\underline{g}$ which satisfies (2.3-10) and (2.3-16) is called a k^{th} order "detection generator" for $\underline{f}$. The detection generator theorem has shown that as long as (A,C) is an observable pair, there always exists a detection generator associated with a detector gain.

2.3.2.2 Detector Gain

The preceding development of the detection generator serves as a prelude to the following theorem from which D can be found:

Detector gain theorem —

If (i) (A,C) is an observable pair

(ii) rank $W_f = k$

(iii) rank $CW_f = 1$

and k eigenvalues of [A-DC] associated with W_f are given by the roots of

$$s^k + p_k s^{k-1} + \ldots + p_2 s + p_1 = 0 \qquad (2.3\text{-}17)$$

Then D is a solution of

$$DCA^{k-1}\underline{f} = p_1\underline{g} + p_2 A\underline{g} + \ldots + p_k A^{k-1}\underline{g} + A^k\underline{g} \qquad (2.3\text{-}18)$$

where p_i are scalars and $\underline{g}$ is a k^{th} order detection generator for $\underline{f}$

Conversely, any D satisfying (2.3-18) is a detector gain for $\underline{f}$ and k eigenvalues of [A-DC] associated with W_f are given by the roots of (2.3-17).

This theorem is perhaps the main result of detection filter theory and follows mainly from the Cayley-Hamilton Theorem applied to [A-DC] and invoking (2.3-11).

Premultiplying (2.3-16) by CA^μ and using (2.3-10) yields

$$CA^\mu\underline{f} = CA^{k-1}\underline{g} \qquad (2.3\text{-}19)$$

which when substituted into (2.3-18) gives

$$DCA^\mu\underline{f} = p_1\underline{g} + \ldots + p_k A^{k-1}\underline{g} + A^k\underline{g} \qquad (2.3\text{-}20)$$

where $\mu = 0$ corresponds to the condition $C\underline{f} \neq 0$. The rest of the results will be derived for $\mu = 0$, but the corresponding general results can be obtained by simply replacing $\underline{f}$ by $A^\mu\underline{f}$.

The general solution of (2.3-20) is expressible in the form

$$D = D_P + D_H$$

where (2.3-21)

$$D_P = \underline{q}_D[(C\underline{f})^T C\underline{f}]^{-1}(C\underline{f})^T$$
$$D_H = D'[I - C\underline{f}[(C\underline{f})^T C\underline{f}]^{-1}(C\underline{f})^T]$$

with $\underline{q}_D$ being the right side of (2.3-20) and D' an arbitrary matrix the same dimensions as D. D_P is the particular solution of D and satisfies (2.3-20). Therefore, D_P is alone sufficient to guarantee condition (i) of detectability and specify k eigenvalues of [A-DC]. D_H is the homogeneous solution of D, i.e., $D_H C\underline{f} = \underline{0}$, and represents the freedom left in D after having satisfied (2.3-20). Condition (ii) of detectability will be satisfied only if the remaining n-k eigenvalues of [A-DC] can be assigned using D'.

2.3.2.3 Detection Space

Beard proposes the following auxiliary lemma to show that the remaining eigenvalues of [A-DC] can be assigned (the actual assignment of these remaining eigenvalues is discussed in Chapter 3):

Auxiliary detector gain lemma –

For A', C', and D' given by

$$\begin{aligned} A' &= A - q_o[(C\underline{f})^T C\underline{f}]^{-1}(C\underline{f})^T C \\ C' &= C - C\underline{f}[(C\underline{f})^T C\underline{f}]^{-1}(C\underline{f})^T C \\ D' & \text{ arbitrary} \end{aligned} \tag{2.3-22}$$

The number of eigenvalues of [A'-D'C'] = [A-DC] that can be specified by D' is q' = rank M' where

$$M' = [C' \mid C'A' \mid \cdots \mid C'A'^{n-1}]^T \tag{2.3-23}$$

This lemma shows that the total number of eigenvalues of [A-DC] that can be specified while constraining D to be a detector gain is k + q'. Therefore, condition (ii) of detectability will be satisfied only if n = k + q'.

The next lemma shows that M' does not depend upon the particular detection generator used or its order k even though it appears to through A':

Detection space theorem —

For K defined by

$$K = A - A\underline{f}[(C\underline{f})^T C\underline{f}]^{-1}(C\underline{f})^T C$$

M' is also equivalent to

$$M' = [C' \mid C'K \mid \cdots \mid C'K^{n-1}]^T \tag{2.3-24}$$

Since C' and K depend only on A, C, and $\underline{f}$, M' is the same regardless of what detection generator is used. Thus the number of eigenvalues of [A-DC] that can be specified while constraining D to be a detector gain is always the same no matter what $\underline{g}$ is used to find D_p. The null space of M' is called the "detection space" of $\underline{f}$ and corresponds to the controllable space of $\underline{f}$ with respect to [A-DC]. The dimension of the detection space is called the "detection order" ν, and a detection generator of order ν is called the "maximal generator" for $\underline{f}$.

The rank of the null space of M' is $\nu = n - q'$, so if q' eigenvalues of [A-DC] are assignable using D', then a detection generator of order $k = \nu$ is needed to assign all the eigenvalues and satisfy condition (ii) of detectability. Beard shows that if (A,C) is an observable pair, $\underline{f}$ will have a maximal generator and it is unique. Thus, the event $\underline{f}$ is guaranteed detectable if (A,C) is observable. Beard also shows that although observability is necessary to satisfy condition (ii) of detectability, it is not necessary to satisfy condition (i) if $\underline{f}$ is in the observable space. The number of assignable eigenvalues in this case is the rank of the observability matrix M defined by (2.3-23) without the primes. Therefore, if one is willing to accept n-rank M unassignable eigenvalues, every observable event is detectable in this weaker sense for the unobservable pair (A,C).

The following theorem summarizes the conditions for detectability:

Detectability Theorem —

Every vector in the state space R^n is detectable if and only if (A,C) is an observable pair.

2.3.3 Sensor Detectability

In deriving (2.2-5), we noted a fundamental difference between measurement and input failures; whereas an actuator failure only drives the state error through the event vector $\underline{b}_j$, a sensor failure drives the state error through the output feedback term as well as the event vector $\underline{d}_j$. Consequently, the best the filter can do is constrain the output error to a two-dimensional plane. Even though this was shown only for the fully measurable case, Beard proves this is true for partially measurable systems as well.

Because the event vector $\underline{d}_j$ is at the discretion of the designer and is not related at all to the sensor itself, the theorems in the previous sections are not directly applicable. Instead, the i^{th} row of C will be associated with a sensor failure just as the i^{th} column of B was associated with an actuator failure. This motivates the following modified definition of detectability for sensor failures:

The i^{th} row of C, $\underline{c}_i^T$, is "sensor·detectable" if there exists a gain matrix D such that:

(i) $\underline{\epsilon}_p(t)$ is constrained to a 2D-plane (where $\underline{\epsilon}_p(t)$ is the settled-out solution of (2.2-5))

(ii) the eigenvalues of [A-DC] can be specified arbitrarily

The following theorem is the sensor counterpart to the Detectability Theorem:

Sensor Detectability Theorem —

If (A,C) is an observable pair and $\underline{c}_i^T$ is linearly independent of all the other rows of C, then $\underline{c}_i^T$ is sensor detectable.

The filter design procedure for sensors is stated in two steps which also serve as a basis for the proof of this theorem:

(1) first choose D to be a detector gain for $\underline{f}$, by making it a solution of (2.3-20), where

$$C\underline{f}_i = \hat{\underline{e}}_{Pi} \tag{2.3-25}$$

(2) then make D' a detector gain for $\underline{d}_i$ with respect to the pair (A',C')

Before proving the theorem, we first outline two implications of (2.3-25). First, the existence of $\underline{f}$ is guaranteed because $\underline{c}_i^T$ is linearly independent of the other rows of C (pseudo-inverse of C exists). Therefore by the detectability theorem, $\underline{f}$ is detectable. Second, $\underline{d}_i$ is now fixed because (2.3-20) becomes

$$\underline{d}_i = p_1\underline{g} + \ldots + p_\nu A^{\nu-1}\underline{g} + A^\nu\underline{g} \tag{2.3-26}$$

where ν is the detection order of $\underline{f}$, and $\underline{g}$ is its maximal generator.

Assume for the moment that $\underline{d}_i$ lies in the observable space of (A',C') and prove that condition (i) of sensor detectability holds. This assumption plus the detectability theorem guarantee that the detector gain D' exists for the event $\underline{d}_i$ with respect to (A',C'). Furthermore, $C'\underline{e}'_s(t)$ lies in a fixed direction where $\underline{e}'_s(t)$ is the settled-out solution of

$$\dot{\underline{e}}'(t) = (A'-D'C')\underline{e}'(t) - \underline{d}_i n(t) \tag{2.3-27}$$

Since A'-D'C' is equal to A-DC, $\underline{e}'_s(t)$ is the settled state error for the pair (A,C) using the gain D as well. This fixed direction can therefore be expressed as

$$C'\underline{e}'_s(t) = \underline{y}_d m_d(t) \tag{2.3-28}$$

where $\underline{y}_d$ is this fixed direction and $m_d(t)$ is a scalar function depending upon n(t).

From (2.3-22), C' is given by

$$C' = C - \hat{\underline{e}}_{p_i} \underline{c}_i^T \tag{2.3-29}$$

so that

$$C\underline{e}'_s(t) = \underline{y}_d m_d(t) + \hat{\underline{e}}_{p_i} \underline{c}_i^T \underline{e}'_s(t) \tag{2.3-30}$$

From (2.2-5), the output error is

$$\underline{\epsilon}(t) = \underline{y}_d m_d(t) + \hat{\underline{e}}_{p_i}[n(t) + \underline{c}_i^T \underline{e}'_s(t)] \tag{2.3-31}$$

Since the quantity in brackets is a scalar, $\underline{\epsilon}(t)$ lies in the plane spanned by $\underline{y}_d$ and $\hat{\underline{e}}_{p_i}$. Beard shows that $\underline{y}_d$ can be broken up into a component in the $\hat{\underline{e}}_{p_i}$ direction and the $CK^\ell A\underline{f}$ direction, where K is given in (2.3-24) and ℓ is the lowest integer such that $CK^\ell A\underline{f} \neq 0$. Thus the output error lies in the plane formed by $\hat{\underline{e}}_{p_i}$ and $CK^\ell A\underline{f}$.

Let us now relax the assumption that $\underline{d}_i$ lies in the observable space of (A'C'). The observability matrix is given by M' defined by (2.3-23), so if $\underline{d}_i$ is unobservable, M' $\underline{d}_i$ = $\underline{0}$. Therefore, $\underline{d}_i$ lies in the nullspace of M', or equivalently, in the detection space of $\underline{f}$. As Beard shows, every vector contained in the detection space of $\underline{f}$ has the same detection order and detection space as $\underline{f}$, so D is a detector gain for $\underline{d}_i$ as well as for $\underline{f}$ (i.e., $\underline{d}_i$ is detectable in the traditional sense). Therefore, the second step of making D' a detector gain for $\underline{d}_i$ is unnecessary, and $C\underline{e}_s'(t)$ will lie in a fixed direction.

Finally, it remains to show that all the eigenvalues of A-DC are assignable. When making D a detector gain for $\underline{f}$, ν eigenvalues can be assigned using (2.3-17) since (A,C) is an observable pair. If $\underline{d}_i$ is unobservable with respect to (A',C'), then D' is unconstrained, and the remaining n-ν eigenvalues are assignable by the free choice of D' by the auxiliary detector gain lemma. Now if $\underline{d}_i$ is observable for (A',C'), a total of only n-rank M' eigenvalues can be assigned while making D' a detector gain for $\underline{d}_i$ and using D", the auxiliary detector gain for D' (D' is the auxiliary detector gain for D). The remaining ν eigenvalues are associated with the unobservable space of (A',C') which coincides with the detection space of $\underline{f}$ and have already been specified by making D a detector gain for $\underline{f}$. Thus all the eigenvalues are assignable.

One last remark helps to simplify the computation of D'. D' can be made a detector gain for $A\underline{f}$ instead of $\underline{d}_i$ because $A\underline{f} \equiv \underline{d}_i$(mod E), where E denotes the unobservable space of (A',C'). This means that the difference $A\underline{f} - \underline{d}_i$ lies in E. With this substitution, the need to compute (2.3-26) is avoided.

2.3.4 Sets of Events

Up to this point, we have only concerned ourselves with detectability of a single event. Now consider the set of event vectors $\{\underline{f}_1,\ldots,\underline{f}_r\}$. The set is considered "mutually detectable" if there exists a D that satisfies the conditions of detectability for all r events. If the set of vectors satisfy

$$\text{rank } CF = r$$
$$F \triangleq [A^{\mu_1}\underline{f}_1, \ldots, A^{\mu_r}\underline{f}_r] \tag{2.3-32}$$

with μ defined for each $\underline{f}_i$ as in (2.3-14), then the events are called "output separable." Two events are not output separable if the output error for both events lie in the same direction. All sets of events considered in this work were output separable.

A necessary and sufficient condition for a set of output separable events to be mutually detectable is given by the following theorem:

Group Detection Theorem —

The output separable vectors $\{\underline{f}_1,\ldots,\underline{f}_r\}$ are mutually detectable if and only if the sum of the individual detection orders of the $\underline{f}_i$ is equal to the group detection order given by the dimension of the nullspace of M_G', (n-rank M_G'), where M_G' is defined as M' in (2.3-24) with $\underline{f}$ replaced by F.

The basic outline of the proof proceeds as follows. Since the set of events $\{\underline{f}_1,\ldots,\underline{f}_r\}$ are output separable, the events have nonintersecting subspaces (shown by Beard), and thus ν_i eigenvalues can be specified for each $\underline{f}_i$ while satisfying (2.3-20). An additional q_G' = rank M_G' eigenvalues can be specified by free choice of D'. Therefore, condition (ii) of detectability is satisfied if and only if $\Sigma \nu_i = n - q_G'$. If the set is not mutually detectable, $n - q_G' - \Sigma \nu_i$ eigenvalues cannot be controlled.

2.3.4.1 Actuator Set

The design of the detection filter for a set of actuator events proceeds as follows:

(1) determine the maximal generator for each $\underline{b}_i$ using (2.3-10)

(2) form F as defined in (2.3-32) and divide the set up into output separable subsets

(3) subdivide the set further, if necessary, until the subsets are also mutually detectable

(4) solve (2.3-20) for each $\underline{b}_i$ to make D a detector gain for F while using the p_i's to specify $\Sigma \nu_i$ eigenvalues

(5) specify the remaining eigenvalues of A-DC using D'

Step 3 is unnecessary if the uncontrollable eigenvalues resulting from nonmutual detectability are satisfactory. Beard generates an algorithm that aids in finding these fixed eigenvalues and can also expedite step 3.

2.3.4.2 Sensor Set

One design method for a filter detecting sensor failures proceeds as follows:

(1) for each $\underline{c}_i^T$, determine $\underline{f}_i$ such that $C\underline{f}_i = \hat{\underline{e}}_{p_i}$ where the sensors are linearly independent

(2) form F as in (2.3-32) and subdivide into mutually detectable sets (events are automatically output separable by step 1)

(3) form the set of vectors $\{A\underline{f}_1, \ldots, A\underline{f}_{k_1}\}$ where $F_{k_1} = \{\underline{f}_1, \ldots, \underline{f}_{k_1}\}$ is the set resulting from step 2; let A' and C' be defined as in (2.3-22) with $\underline{f}$ replaced by F_{k_1}; categorize the $A\underline{f}_i$ as follows, and remove those events falling under (iii):

(i) $M'_F A\underline{f}_i \neq \underline{0}$

(ii) $M'_F A\underline{f}_i = \underline{0}$; $\text{rank}\{CA\underline{f}_i, CA'A\underline{f}_i, \ldots, CA'^{n-1}A\underline{f}_i\} = 1$

(iii) $M'_F A\underline{f}_i = \underline{0}$; $\text{rank}\{CA\underline{f}_i, CA'A\underline{f}_i, \ldots, CA'^{n-1}A\underline{f}_i\} > 1$

(4) let $F_{k_2} = \{\underline{f}_1, \ldots, \underline{f}_{k_2}\}$ be the set resulting from step 3; let A' and C' be defined as in (2.3-22) with $\underline{f}$ replaced by F_{k_2}; repeat step 2 with A, C, and $\underline{f}_i$ replaced by A', C', and the $A\underline{f}_i$ in category (i); remove $A\underline{f}_i$'s until the $A\underline{f}_i$ are mutually detectable with respect to (A',C'); if any $A\underline{f}_i$ are removed, some $A\underline{f}_i$ may move from category (ii) to (i) and mutual detectability with the new members must be rechecked

(5) solve (2.3-20) for each $\underline{f}_i$ in the set F_{k_3} resulting from step 4 to make D a detector gain for F_{k_3} while specifying $\Sigma \nu_i$ eigenvalues

(6) solve (2.3-20) for each $A\underline{f}_i$ in the set F_{k_3} to make D' a detector gain for AF_{k_3} with respect to (A',C') while specifying $\Sigma \nu_{i_{mod}}$ eigenvalues ($\nu_{i_{mod}}$ denotes the maximum order a detection generator for $A\underline{f}_i$ can have)

(7) use D" to specify the remaining eigenvalues of (A'-D'C') = A-DC [D" is to D' as D' is to D in section 2.3.4.1.]

A few remarks concerning step 3 are in order. If category (i) is the case, $A\underline{f}_i$ does not lie in the unobservable space of C' with respect to A' and thus $A\underline{f}_i$ is detectable. For categories (ii) and (iii), $A\underline{f}_i$ does lie in the unobservable space so $A\underline{f}_i$ is not detectable; this is of no consequence, however, if D is a detector gain for $A\underline{f}_i$ as well as $\underline{f}_i$ as is the case for category (ii) (see end of Section 2.3.3). The rank test given in step 3 determines whether the given sequence of vectors lie in a fixed direction—if they do, category (ii) applies.

The assignment of eigenvalues in the last three steps is depicted in Fig. 2.2: $\Sigma\nu_i$ eigenvalues are assigned using D_p and $n - \Sigma\nu_i$ are specified using D'; of the $n - \Sigma\nu_i$, $\Sigma\nu_{i_{mod}}$ are assigned using D_P' and the rest using D_H' which depends on D" (subscripts P and H denote particular and homogeneous solutions, respectively).

2.4 Two-mode Design Example

As a simple example of the analytic design procedure for the failure detection filter, consider the normal mode equations

$$\ddot{\Psi}_r(t) + \omega_r^2 \Psi_r(t) = G_r u(t) \qquad r=1,2,\ldots \qquad (2.4\text{-}1)$$

where Ψ_r represents the amplitude of the r^{th} mode, ω_r its natural frequency, and G_r the amplitude of the input u. In a single degree of freedom mass-spring system, for instance, Ψ is just the displacement x and ω^2 is the spring constant k divided by the mass m.

In particular, consider the system of equations for two flexible modes of a beam given in the form (2.1-1):

$$\begin{bmatrix} \dot{\psi}_1 \\ \ddot{\psi}_1 \\ \dot{\psi}_2 \\ \ddot{\psi}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\omega_2^2 & 0 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \dot{\psi}_1 \\ \psi_2 \\ \dot{\psi}_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{M}\phi_1(\epsilon_1) & \frac{1}{M}\phi_1(\epsilon_2) \\ 0 & 0 \\ \frac{1}{M}\phi_2(\epsilon_1) & \frac{1}{M}\phi_2(\epsilon_2) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad (2.4\text{-}2)$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 & \phi_1(\alpha_1) & 0 & \phi_2(\alpha_1) \\ 0 & \phi_1(\alpha_2) & 0 & \phi_2(\alpha_2) \end{bmatrix} \begin{bmatrix} \psi_1 \\ \dot{\psi}_1 \\ \psi_2 \\ \dot{\psi}_2 \end{bmatrix}$$

where force actuators at ϵ_1, ϵ_2 and translation rate sensors at α_1, α_2 are used [$\phi_i(x_j)$ represents value of the i^{th} mode shape at x_j].

To simplify the design procedure, Beard shows that if the matrix A is replaced by

$$\bar{A} = A - \bar{D}C \qquad (2.4\text{-}3)$$

with $\bar{D}$ arbitrary, the detector gain is unaffected. Following this suggestion, let's replace A by the simpler form

$$\bar{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\omega_1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\omega_2^2 & 0 \end{bmatrix} \qquad (2.4\text{-}4)$$

Now let's proceed to design a detection filter that can detect failures of either actuator #1 or actuator #2. The two event vectors for this case are $\underline{b}_1$ and $\underline{b}_2$, the two columns of B. If the two actuators are not both at the same location or placed symmetrically about the center (for a free-free beam), the directions $C\underline{b}_1$ and $C\underline{b}_2$ will be linearly independent and thus output separable. To determine mutual detectability, the detection order for each $\underline{f}$ must be found.

For an appropriate choice of D, K is found from (2.3-24) to be equal to $\bar{A}$, and C' using (2.3-22) becomes

$$C' = \begin{bmatrix} 0 & c_1 & 0 & c_3 \\ 0 & c_2 & 0 & c_4 \end{bmatrix} \tag{2.4-5}$$

where the c_i's are functions of the elements of $C\underline{b}$ and C. Now form

$$M' = \begin{bmatrix} C' \\ \hdashline C'K \\ \hdashline C'K^2 \\ \hdashline C'K^3 \end{bmatrix} = \begin{bmatrix} 0 & c_1 & 0 & c_3 \\ 0 & c_2 & 0 & c_4 \\ \hdashline -\omega_1^2 c_1 & 0 & -\omega_2^2 c_3 & 0 \\ -\omega_1^2 c_2 & 0 & -\omega_2^2 c_4 & 0 \\ \hdashline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.4-6}$$

The two rows of C' are linearly dependent so that the rank of M' is 2 and thus the detection order of $\underline{f}$ is 2 for both $\underline{b}_1$ and $\underline{b}_2$. Since M_G' is the null matrix, the group detection order is four and the events are mutually detectable.

The maximal generator for $\underline{f}$ is second order in this case and so must satisfy the set (2.3-10):

$$\begin{aligned} C\underline{g} &= \underline{0} \\ CA\underline{g} &= \underline{0} \end{aligned} \tag{2.4-7}$$

If $\underline{g}$ is represented by $[g_a\ g_b\ g_c\ g_d]^T$, the first of (2.4-7) becomes

$$\begin{aligned} \phi_{11} g_b + \phi_{21} g_d &= 0 \\ \phi_{21} g_b + \phi_{22} g_d &= 0 \end{aligned} \tag{2.4-8}$$

where ϕ_{ij} is used to denote $\phi_i(\alpha_j)$. Since we have assumed linear independence of the sensors,

$$\begin{vmatrix} \phi_{11} & \phi_{21} \\ \phi_{12} & \phi_{22} \end{vmatrix} \neq 0 \tag{2.4-9}$$

and we must accept the trivial solution $g_b = g_d = 0$. The second equation in (2.4-7) transforms into

$$\begin{bmatrix} g_a \\ g_c \end{bmatrix} = \begin{bmatrix} -\omega_1^2 \phi_{11} & -\omega_2^2 \phi_{21} \\ -\omega_1^2 \phi_{12} & -\omega_2^2 \phi_{22} \end{bmatrix}^{-1} \begin{bmatrix} \phi_{11} & \phi_{21} \\ \phi_{12} & \phi_{22} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \tag{2.4-10}$$

with β_1 and β_2 denoting the nonzero elements of $\underline{b}$, and (2.4-9) guaranteeing existence of the inverted matrix.

All that remains is to solve (2.3-20) with k = 2 for each event. If λ is chosen for all the filter eigenvalues, $p_1 = \lambda^2$ and $p_2 = -2\lambda$. The particular solution of D is found to be

$$D_p = \begin{bmatrix} (p_1-\omega_1^2)g_{1a} & (p_1-\omega_1^2)g_{2a} \\ -p_2\omega_1^2 g_{1a} & -p_2\omega_1^2 g_{2a} \\ (p_1-\omega_2^2)g_{1c} & (p_1-\omega_2^2)g_{2c} \\ -p_2\omega_2^2 g_{1c} & -p_2\omega_2^2 g_{2c} \end{bmatrix} \begin{bmatrix} \phi_{11}\beta_{11}+\phi_{21}\beta_{12} & \phi_{11}\beta_{21}+\phi_{21}\beta_{22} \\ \phi_{12}\beta_{11}+\phi_{22}\beta_{12} & \phi_{12}\beta_{21}+\phi_{22}\beta_{22} \end{bmatrix} \qquad (2.4\text{-}11)$$

where β_{ij} denotes β_j for $\underline{b}_i$. [Note that the original A is used in obtaining this result.]

Since all four eigenvalues have been assigned using (2.3-20), $D=D_p$ and the design is complete. If the gain were being designed for only one actuator, the remaining two eigenvalues would have to be assigned by some other means. This is often not easy to do analytically using determinants, and is best done computationally using D', especially for higher order systems.

CHAPTER III

COMPUTATIONAL DESIGN OF FILTER

For systems of order four and greater, the analytic design approach outlined in the previous chapter becomes very cumbersome. The design example was actually a relatively simple case facilitated by the fact that the same types of sensors were used and the filter was designed to detect two events. In most cases, it is not possible to solve for the detection generator explicitly nor is it possible to assign all filter eigenvalues using D_p alone (definitely not for sensor failure events). This chapter will present a design alternative based upon a process called "orthogonal reduction" which is best implemented on a computer. All of these algorithms were first proposed by Beard but also appear in Jones as well.

3.1 Orthogonal Reduction

Many of the processes involved in the filter design involve finding the rank of a matrix or finding a vector in the nullspace of a certain matrix. The detection generator, for example, is in the nullspace of M', the detection space of $\underline{f}$, and the detection order is n-rankM'. The orthogonal reduction procedure is an iterative process which generates a positive semi-definite matrix, the range space of which coincides with the nullspace of a given matrix V, N(V).

Let V be an mxn matrix given by

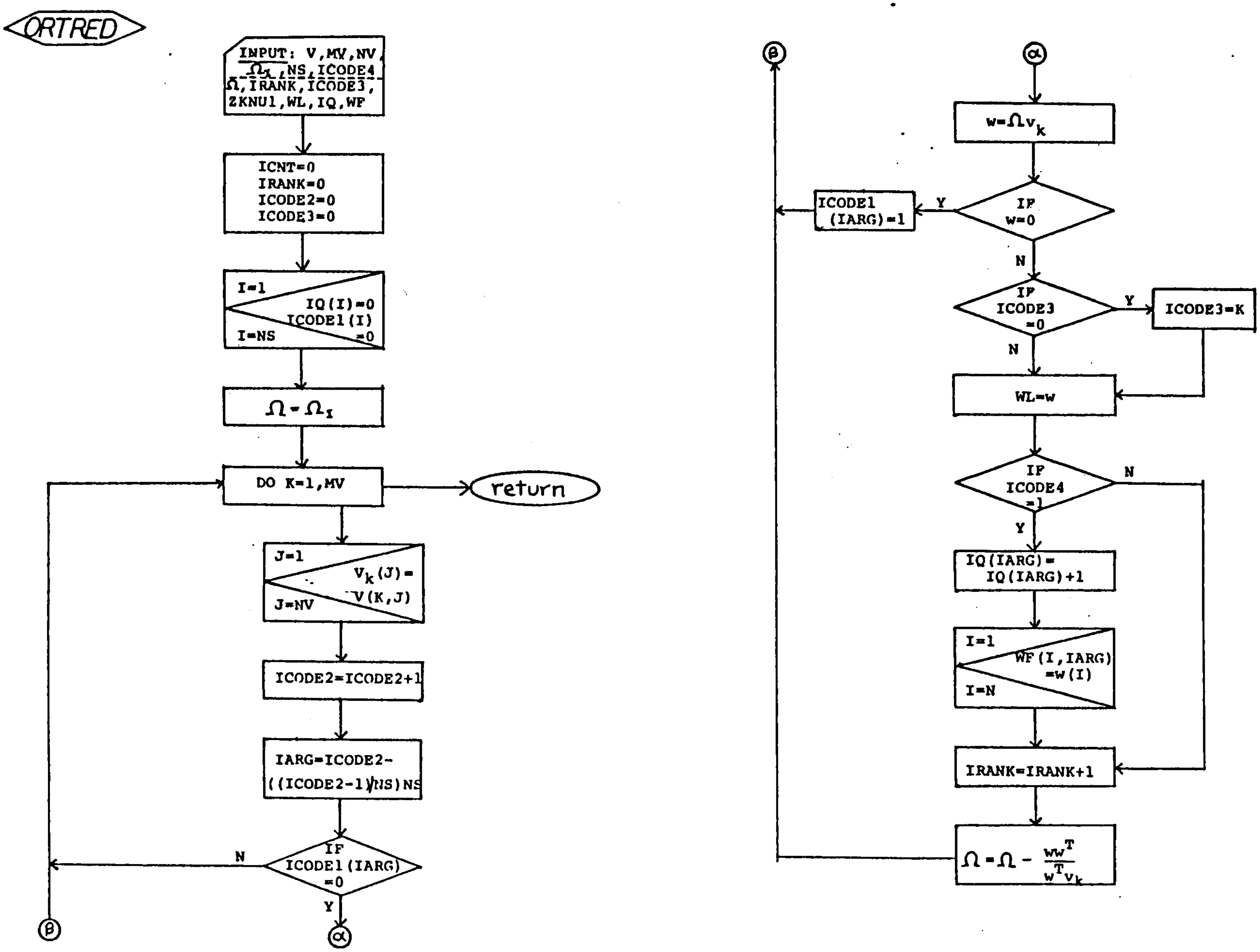


Fig. 3.1: Flowchart for ORTRED.

$$V = \begin{bmatrix} \underline{v}_1^T \\ \cdot \\ \cdot \\ \underline{v}_m^T \end{bmatrix} \qquad (3.1\text{-}1)$$

where $\underline{v}_i^T$ are the rows of V. Let Ω denote the symmetric matrix whose range space is to coincide with the nullspace of V. Each row of V is then tested to see whether it is orthogonal to the range space of V (the rows of V span $N^{\perp}(V)$) by defining the auxiliary vector

$$\underline{w}_i = \Omega \underline{v}_i \qquad (3.1\text{-}2)$$

If $\underline{w}_i \neq \underline{0}$, $\underline{v}_i$ is not perpendicular to the columns of Ω (Ω is symmetric) and Ω is reduced so that $\underline{w} = 0$. This is repeated until the columns of Ω , $R(\Omega)$, are all orthogonal to the rows of V, $N^{\perp}(V)$.

The reduction algorithm proceeds as follows:

(i) with Ω_i from the previous iteration, form the auxiliary vector

$$\underline{w}_i = \Omega \underline{v}_i$$

(ii) if $\underline{w}_i = \underline{0}$, set

$$\Omega_{i+1} = \Omega_i$$

if $\underline{w}_i \neq \underline{0}$, set

$$\Omega_{i+1} = \Omega_i - \frac{\underline{w}_i \underline{w}_i^T}{\underline{w}_i^T \underline{v}_i} \qquad (3.1\text{-}3)$$

The procedure must begin with Ω_1 positive definite to ensure Ω remains positive semi-definite and $\underline{w}_i^T \underline{v}_i \neq 0$. The rank of the matrix is equal to the number of reductions performed.

The algorithm for subroutine ORTRED is presented in Fig. 3.1 with several modifications tailored to the types of matrices we will be dealing with. First of all, a row can be skipped in the reduction process if it is known that it is linearly dependent on the preceding rows. This is especially useful for matrices such as M' in (2.3-24) which are generated cyclically. When the auxiliary vector for a particular row of M', $(\underline{c}_j^T K^{\ell})^T$ for example, is zero, then this row is linearly dependent on the preceding rows of M'. Also, the remaining rows of M' that depend upon $\underline{c}_j$, $(\underline{c}_j^T K^i)^T$ for $i > \ell$, are also dependent upon the preceding rows of M'. Thus, after the appearance of the first zero auxiliary vector, the rest of the rows that depend on $\underline{c}_j$ needn't be considered. [ICODE1(J) is set equal to one at this termination point.] When the termination points for all c_j's have been reached, M' is completely reduced.

ICODE3 and WL are used when computing the maximal generator. ICODE3 is the row of C associated with the last nonzero auxiliary vector WL. The arrays IQ and WF are used when calculating D'. IQ stores the orders of K associated with the termination points for each row of C, and WF stores the last nonzero auxiliary vector for each row.

3.2 Input failure event design

The computation of D for actuators is accomplished by calling three subroutines: SEPDET, DETGEN, and DGAIN. Subroutine SEPDET first determines whether the events are output separable—it then determines whether the events are mutually detectable.

In the process of determining mutual detectability, SEPDET calls subroutine DETGEN to find the maximal generator for each event. Finally, subroutine DGAIN is called to calculate D using the detection orders and generators from SEPDET. Inputs to the main program FDFIL (see Appendix A) are the system matrix A and its order N, the measurement matrix C and number of sensors NS, the number of actuators NA, the matrix of event vectors FALL and number of events NF, and the filter eigenvalues EV.

3.2.1 Subroutine SEPDET

The flowchart for SEPDET appears in Fig. 3.2. The program first computes μ_i for each f_i according to (2.3-14) and replaces f_i by $A^{\mu}f_i$ in FALL and CF. The rank of CF is found using ORTRED to see whether the events are separable. If not, the error message 'NOT SEPARABLE' is printed and the designer must remove the dependent events from FALL and start again.

Once the events are output separable, the program proceeds to determine whether the events are mutually detectable. The matrix M' in (2.3-24) is generated using C' and K for the full set of events and the rank is found using ORTRED to determine the group detection order, ν_G = N - rankM_F'. The call to DETGEN yields the individual detection orders ν_i. If $\nu_G \neq \sum \nu_i$, the events are not mutually detectable, and one or more events will have to be removed and the process started over.

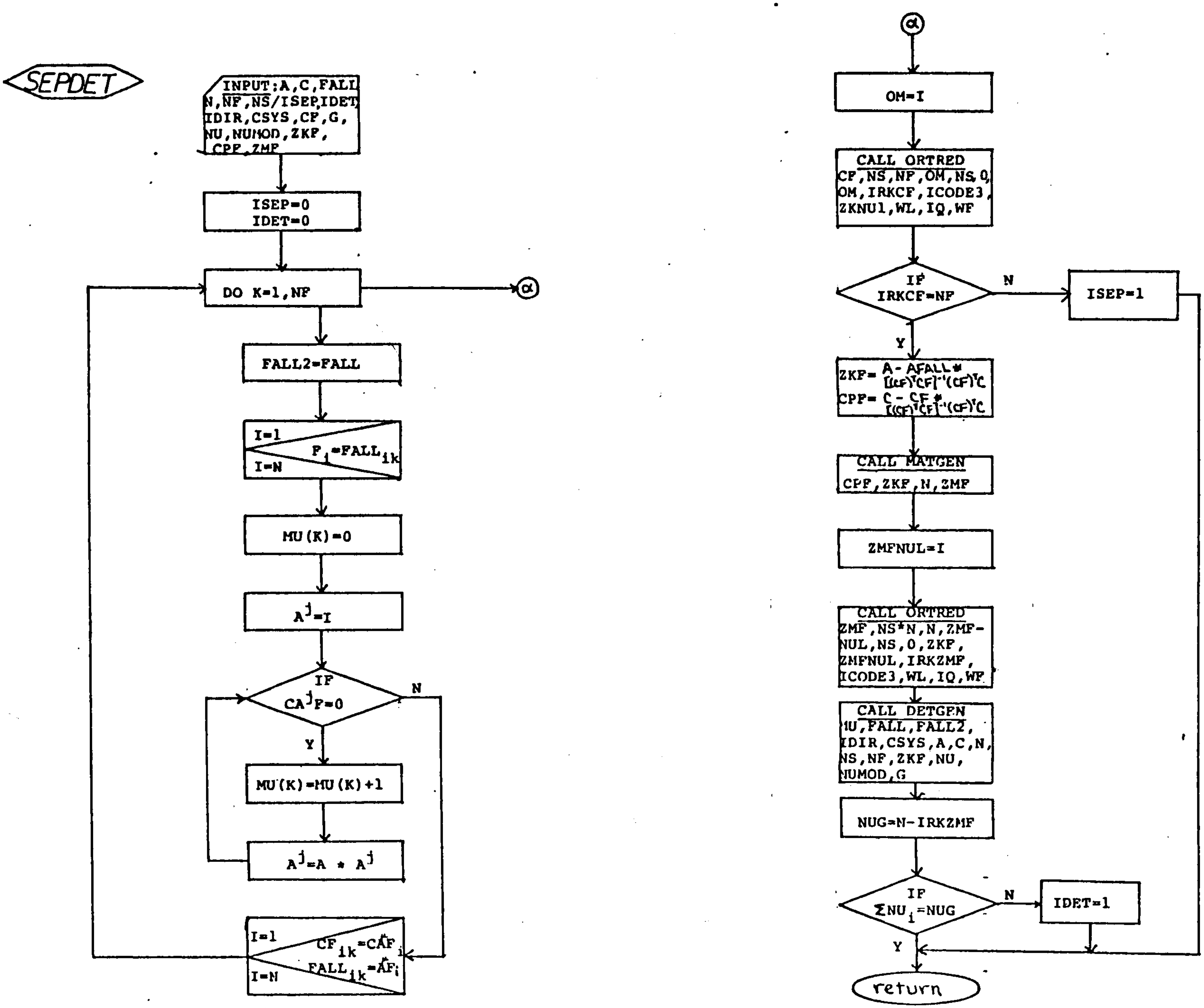


Fig. 3.2: Flowchart for SEPDET.

3.2.2 Subroutine DETGEN

This routine will calculate the "maximal" generator and detection order ν for an event, or alternatively, the "maximum" generator and its order ν_{mod} if (C,A) is an unobservable pair. The process is illustrated in Fig. 3.3 and begins with calculating M' in (2.3-24) for the single event and then using ORTRED to find the detection order, ν=N-rankM'. The resulting nullspace in this case is the detection space of $\underline{f}$ and coincides with the range space of the final Ω .

In addition to being in the detection space of $\underline{f}$, the detection generator must satisfy (2.3-10). This is accomplished by applying ORTRED to the matrix

$$M_K = \left[C \,\vdots\, CK \,\vdots\, \cdots \,\vdots\, CK^{\nu-1} \right] \tag{3.1-4}$$

starting with the final Ω_f from the first step. Since C spans a subspace one dimension greater than C' and the process is started with Ω_f which is orthogonal to C', only one row of C will not be terminated when first encountered. If WL is the last nonzero auxiliary vector for this row $\underline{c}_j$, the maximal generator is given by

$$\underline{g} = \left(\frac{\underline{c}_j^T A^{\mu} \underline{f}}{\underline{c}_j^T K^{\nu-1} \underline{w}_{\ell}} \right) \underline{w}_{\ell} \tag{3.1-5}$$

As stated earlier, if (C,A) is not observable, the detection generator found in (3.1-5) is not maximal (order is not ν), but it is the generator of maximum possible order; this

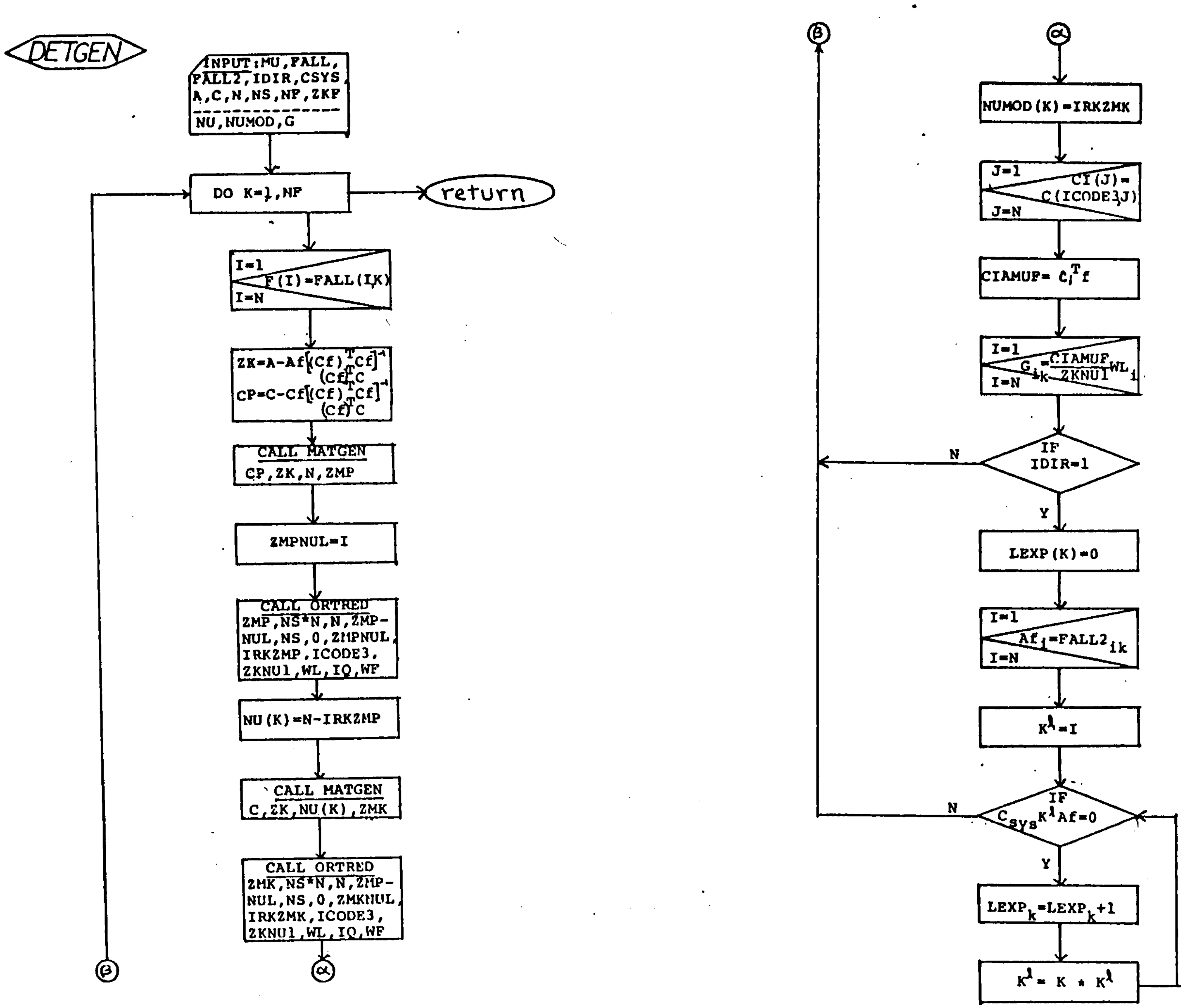


Fig. 3.3: Flowchart for DETGEN.

generator will be referred to as the "modified" generator and has order ν_{mod} = rank M_k. If (C,A) is an observable pair, ν_{mod} will equal ν . The rest of the algorithm computes the vector $CK^{\ell} A\underline{f}$ which is one component of the output error for sensor failures.

3.2.3 Subroutine DGAIN

This subroutine will finally compute the detector gain D while assigning the filter eigenvalues. The process begins (see Fig. 3.4) by finding the particular solution of D for

$$DCF = Q_D \tag{3.1-5}$$

where

$$Q_D = [q_1, \ldots, q_{NF}]$$

$$q_{D_i} = p_1 g_i + \ldots + p_{\nu_{MOD_i}} A^{\nu_{MOD_i}-1} g_i + A^{\nu_{MOD_i}} g_i$$

and ν_{mod_i} and $\underline{g}_i$ have already been calculated by SEPDET. The coefficients p_i are found by calling EVAS which computes the coefficients in (2.3-17) generated by $(s-\lambda)^{\nu_{mod_i}}$ where λ is the chosen filter eigenvalue (all are same).

If $\Sigma\nu_{mod_i}$ = n, all eigenvalues are assigned and the detector gain is simply D_p. Otherwise, the remaining eigenvalues must be assigned using D'. Calculating D' is very similar to the procedure used in calculating D_p except that instead of using maximal generators, the last nonzero auxiliary vectors before termination found in reducing M_F' are used. Jones refers to this process as "completion of the state space" because one is assigning eigenvalues associated with that portion of the

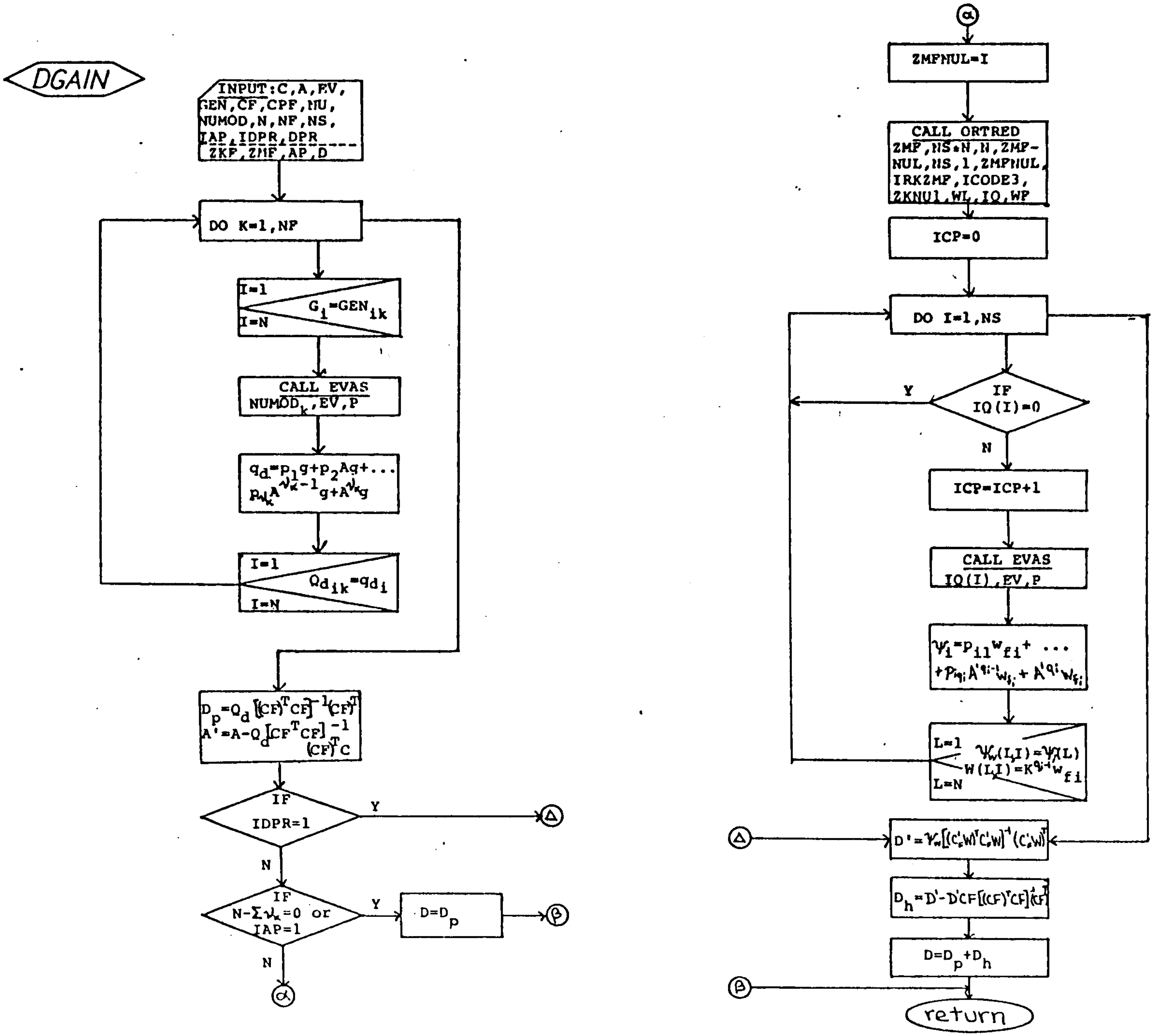

DGAIN
INPUT: C, A, EV, GEN, CF, CPF, NU, NUMOD, N, NF, NS, IAP, IDPR, DPR
ZKF, ZMF, AP, D
DO K=1, NF
I=1
I=N
G_i=GEN_{ik}
CALL EVAS
$NUMOD_k$, EV, P
$q_d=p_1g+p_2Ag+\dots p_{\nu_k}A^{\nu_k-1}g+A^{\nu_k}g$
$Q_{d_{ik}}=q_{d_i}$
$D_p=Q_d[(CF)^TCF]^{-1}(CF)^T$
$A'=A-Q_d[CF^TCF]^{-1}(CF)^TC$
IF IDPR=1
Y
N
IF $N-\Sigma\nu_k=0$ or IAP=1
$D=D_p$
α
Δ
β
ZMFNUL=I
CALL ORTRED
ZMF, NS*N, N, ZMF-NUL, NS, 1, ZMFNUL, IRKZMF, ICODE3, ZKNU1, WL, IQ, WF
ICP=0
DO I=1, NS
IF IQ(I)=0
ICP=ICP+1
CALL EVAS
IQ(I), EV, P
$\psi_i=p_{i1}w_{fi}+\dots+p_{iq_i}A'^{q_i-1}w_{f_i}+A'^{q_i}w_{f_i}$
L=1
L=N
$\psi_w(L,I)=\psi_i(L)$
$W(L,I)=K^{q_i-1}w_{fi}$
$D'=\psi_w[(C_1'W)^T C_1'W]^{-1}(C_1'W)^T$
$D_h=D'-D'CF[(CF)^TCF]^{-1}(CF)^T$
$D=D_p+D_h$
return

Fig. 3.4: Flowchart for DGAIN.

state space not covered by a detection space.

Finding D' begins by applying ORTRED to M'_F (M' for the full set F) to obtain IQ and WF as explained in section 3.1. D' is then any solution satisfying

$$D'C'_F W = \Psi_W \tag{3.1-6}$$

where

$$W = [K^{q_1-1}\underline{w}_{f_1}, \ldots, K^{q_{ICP}-1}\underline{w}_{f_{ICP}}]$$

$$q_i \neq 0 \quad i = 1, \ldots, NS$$

$$\Psi_W = [\underline{\psi}_1, \ldots, \underline{\psi}_{ICP}]$$

$$\underline{\psi}_i = p_{i1}\underline{w}_{f_i} + \ldots + p_{iq_i}A'^{(q_i-1)}\underline{w}_{f_i} + A'^{q_i}\underline{w}_{f_i}$$

The $\underline{w}_{f_i}$ are the columns of WL corresponding to the rankC' nonzero q_i, and A'_F, C'_F correspond to A',C' in (2.3-22) for the full set F. The coefficients p_{ij} are found by calling EVAS for q_i eigenvalues. One solution of (3.1-6) is simply the particular solution

$$D' = \Psi_W[(C'_F W)^T C'_F W]^{-1}(C'_F W)^T \tag{3.1-7}$$

The general solution for D is found by applying (2.3-21) to the full set F.

3.3 Measurement failure event design

The algorithm for sensor filter design does not use any new subroutines (except SENSOR which calls the routines) but the order and implementation are different. A schematic for the sensor design process is given in Fig. 3.5. SEPDET is called first for (A,C,F) to determine whether the set is

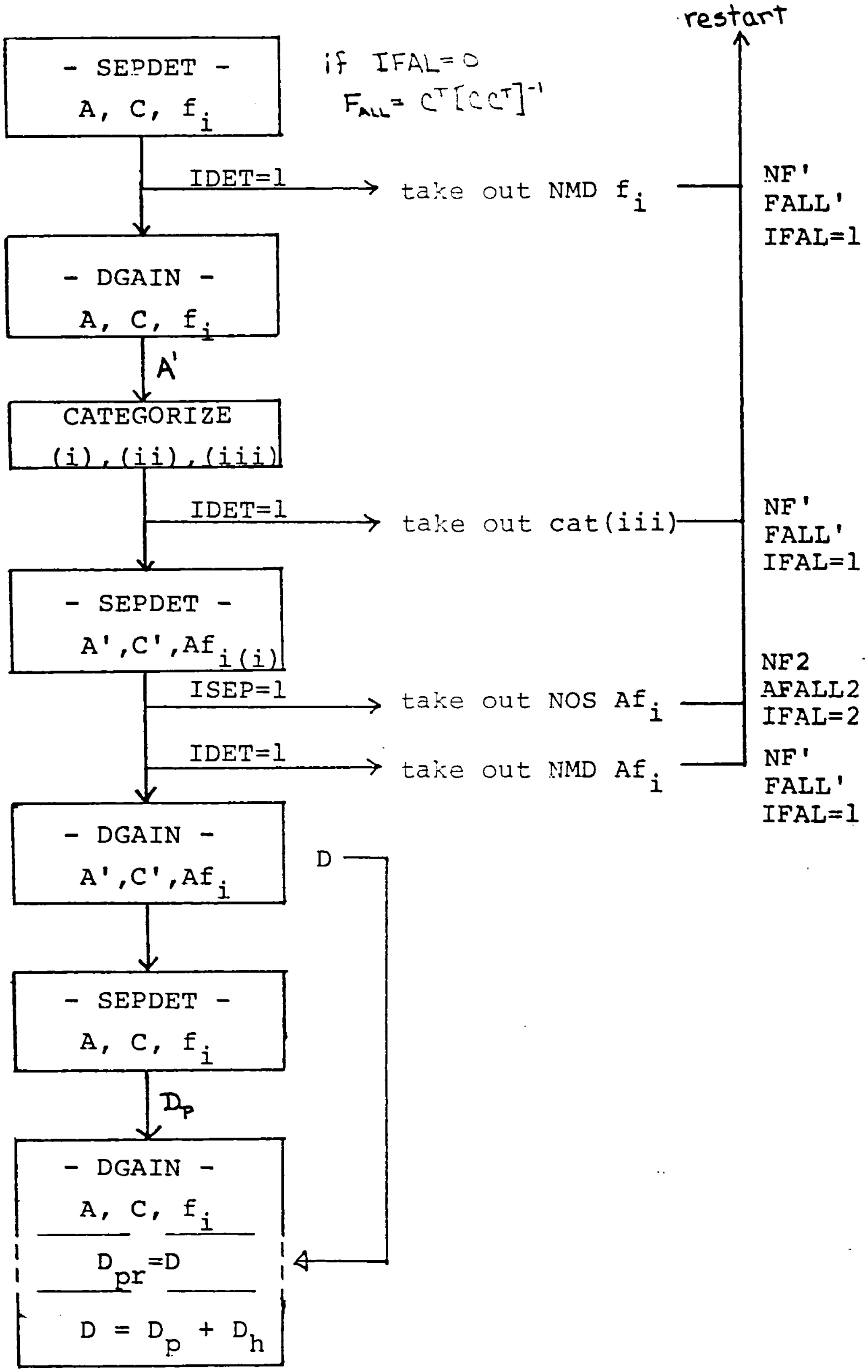


Fig. 3.5: Sensor design process schematic.

mutually detectable (since $C\underline{f}_i = \hat{\underline{e}}_i$, $i = 1,\ldots NF$, it is automatically separable). If the set is not mutually detectable, events must be removed from F until it is and NF (no. events) set accordingly (F must now be inputted directly into the program instead of being calculated in SENSOR).

DGAIN is now called to calculate A'_F (set IAP=1) for use in computing M'_F, the observable space of (C',A') (this has been denoted W_o in the program). The $A\underline{f}_i$ are then generated (recall that $\underline{f}$ is actually $A^{\mu}\underline{f}$ when $\mu \neq 0$) and categorized according to step 3 in Section 2.3.4.2. The $\underline{f}_i$ corresponding to the $A\underline{f}_i$ falling in category (iii) are then removed from F, and the process begins again with the new F and NF until no $A\underline{f}_i$ falls under (iii).

The next step is to call SEPDET for $(A',C',A\underline{f}_i)$. If the $A\underline{f}_i$ are not separable, there is no problem since we still have the output error directions $\hat{\underline{e}}_i$ from the events $\underline{f}_i$ to distinguish between sensor failures. Removing the dependent $A\underline{f}_i$ from the set AF_1 we only need make D' a detector gain for this new set AF_2 of NF_2 events. [The program can be easily modified to handle this case by inputting AF_2 and NF_2 into SENSOR and bypassing the statements NF_2=NF and AF_2=AF.] If the $A\underline{f}_i$ are not mutually detectable, however, the corresponding f_i will have to be removed from F and the whole process restarted with the new F.

Once the $A\underline{f}_i$ are separable/detectable, DGAIN is called to calculate D' (D' is the output D of DGAIN in this case). SEPDET is recalled to regenerate ν_i and $\underline{g}_i$ for the original $\underline{f}_i$ to be used in calculating D_p. Now DGAIN is called a final time for $(A,C,\underline{f}_i)$ with the input D' calculated above to compute D.

CHAPTER IV

APPLICATION TO FLEXIBLE BEAM

In demonstrating various control and failure detection algorithms for large space structures, researchers often use a long beam or rectangular plate as a typical structural element. A beam is usually chosen because of its simplicity and resemblance to a long truss, though plates find their usefulness in simulating the closely spaced frequencies of a flat solar array. In this chapter, the experimental beam at NASA Langley Research Center will be described along with the design of the filter using the finite element description of the beam. The effectiveness of the filter in detecting failures of force actuators on the beam will then be tested using various input frequencies and filter bandwidths in mismatched filter-system models. A position sensor failure will also be simulated for the case where there is only an initial condition on modal amplitudes present. The effect of data sampling will also be investigated.

4.1 NASA LaRC Experimental Beam

In all the failure detection simulation tests, the dynamics model and actuator/sensor types and locations were chosen to correspond to those of the experimental beam at NASA LaRC to predict the performance of the filter in subsequent tests using the actual beam. The Langley beam is made of aluminum (M=0.502 slugs) and is twelve feet long with a 6" x 3/16" cross section (see reference 4). Actuation is provided by electro-

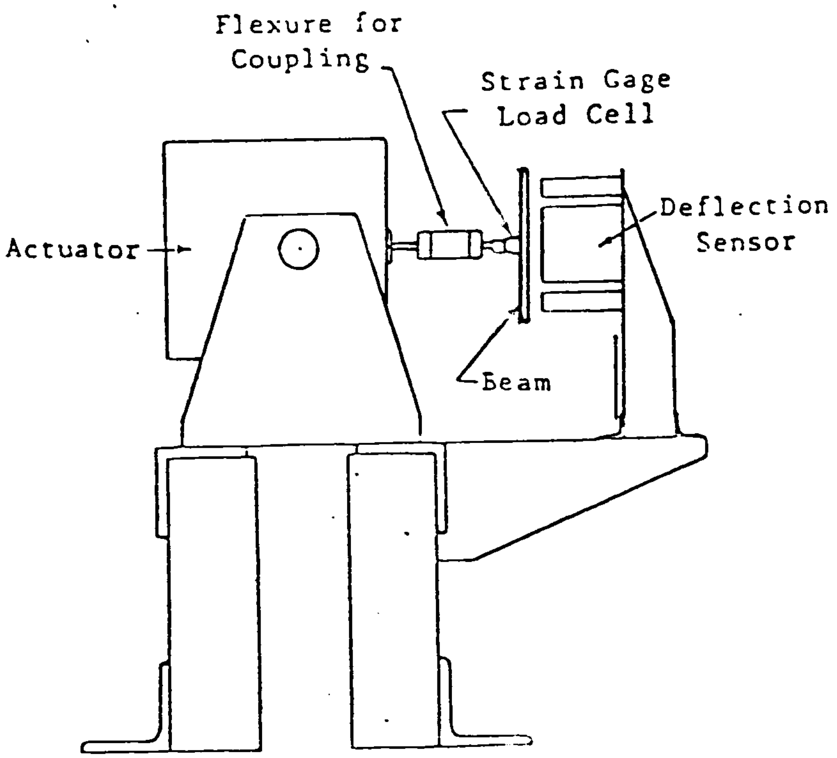


Fig. 4.1: End view of NASA LaRC experimental beam set-up. (taken from reference[4])

TABLE 4.1: Modal frequencies and shapes for first eight modes of LaRC beam (taken from reference[4]).

MODE #	MODAL FREQUENCY (rad/sec)	MODE SHAPE VALUE		
		x=0.5ft	x=2.5ft	x=6.0ft
1	0	7.06	6.18	4.64
2	0	-5.83	-3.15	1.53
3	11.418	-7.83	-0.60	5.91
4	31.360	6.45	-4.28	0.00
5	61.258	5.07	-6.36	6.90
6	100.900	-3.68	5.31	0.00
7	150.185	-2.27	1.74	6.84
8	209.004	-0.88	-2.70	0.00

magnetic shakers at four discrete locations along the beam, and the horizontal deflections are measured by nine Kaman KD-2300 probes, four of which are colocated with the actuators (see Fig. 4.1). All of the deflections of interest to us will take place in the horizontal plane of the beam.

Because the beam is supported in a gravity field and the actuator dynamics are significant, the experimental beam does not behave exactly as a uniform beam would. Researchers at Langley have therefore found it convenient to perform their structural analysis using a finite element mode. The beam model consists of 24 sections, each of which is constrained so that it can translate and rotate in only one degree of freedom. The SPAR program was then used to obtain the natural frequencies and orthogonal mode shapes for the first ten modes of the beam (see Fig. 4.2a-h for first eight modes). The SPAR generated mode shape values at positions 0.5 ft, 2.5 ft, and 6.0 ft from the end of the beam are given in Table 4.1 for the two rigid body modes (translation and rotation) and first six flex modes along with the modal frequencies.

4.2 State equations for beam model

In order to obtain the modal equations for the beam, we begin with the series expansion for the beam displacement

$$y(\epsilon, t) = \sum_{i=1}^{\infty} \phi_i(\epsilon)\Psi_i(t) \qquad (4.2\text{-}1)$$

where $\phi_i(\epsilon)$ is an orthogonal set of mode shapes and $\Psi_i(t)$ are

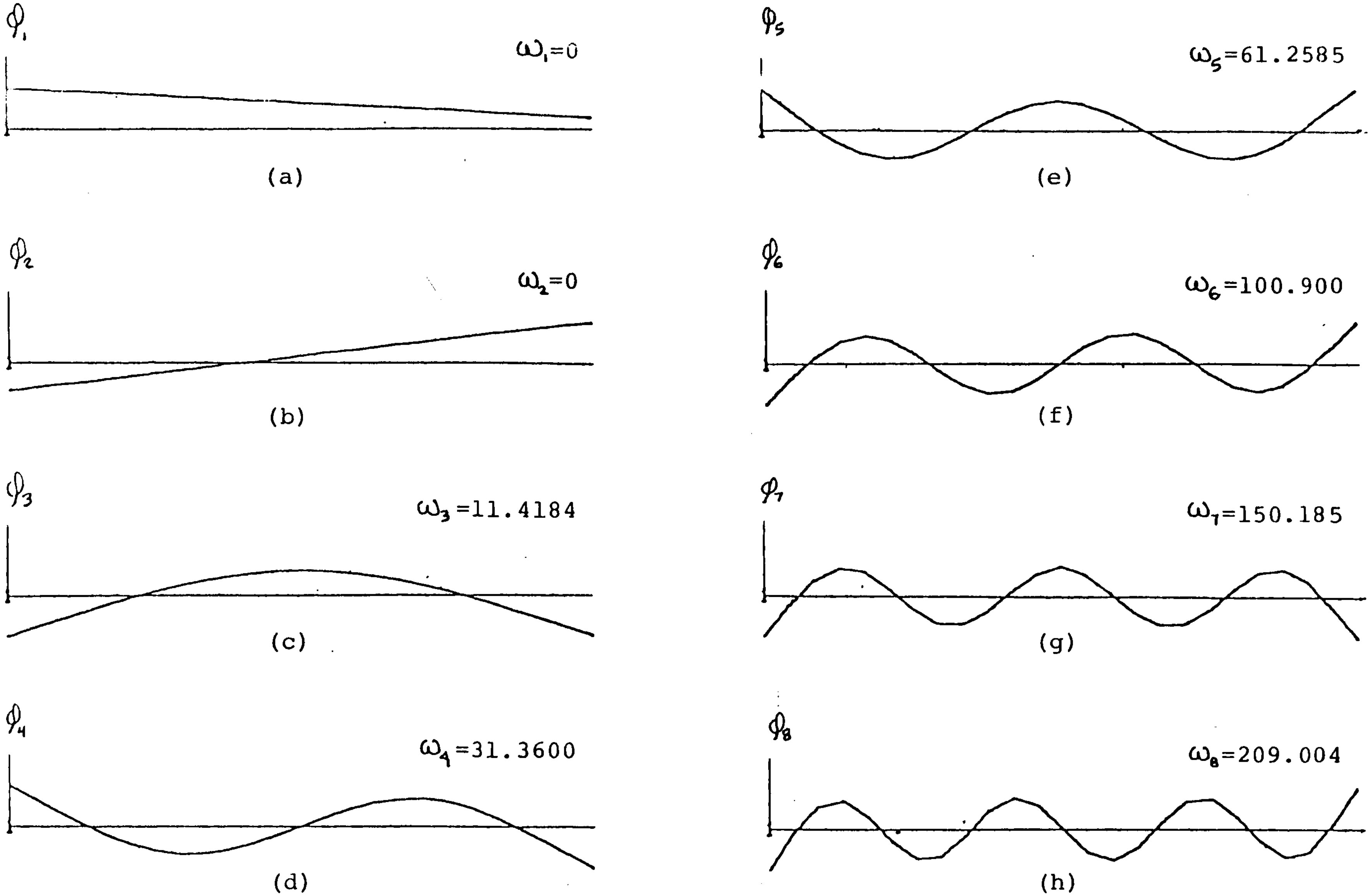


Fig. 4.2: SPAR beam modal frequencies and shapes (frequencies in rad/sec). (taken from reference[4])

the modal amplitudes. Next substitute this into the governing differential equation for the undamped beam

$$EI \frac{\partial^4 y}{\partial \epsilon^4} + m \frac{\partial^2 y}{\partial t^2} = f(\epsilon, t) \tag{4.2-2}$$

where f is the forcing term and m, E, and I are the beam mass (M) per unit length, modulus, and cross-section inertia, respectively. Assuming the use of m point force actuators,

$$f(\epsilon, t) = \sum_{j=1}^{m} \delta(\epsilon - \epsilon_j)\, u_j(t) \tag{4.2-3}$$

with ϵ_j being the actuator positions and $u_j(t)$ the control magnitudes, one obtains the set of relations

$$\omega_i^2 \Psi_i(t) + \frac{d^2\Psi_i}{dt^2} - \frac{1}{M} \sum_{j=1}^{m} \phi_i(\epsilon_j)\, u_j(t) = 0 \tag{4.2-4}$$

where ω_i is the natural frequency of the i^{th} mode and ϕ_i is the normalized mode shape (for ϕ_i not normalized, the input amplitudes must be scaled).

Next, consider casting (4.2-4) in the state-space form

$$\begin{aligned} \dot{\underline{x}} &= A\underline{x} + B\underline{u} \\ y &= C\underline{x} \end{aligned} \tag{4.2-5}$$

where

$$\underline{x}^T = [\Psi_1 \ \dot{\Psi}_1 \ \Psi_2 \ \dot{\Psi}_2 \ \cdots \ \Psi_N \ \dot{\Psi}_N]$$

$$A = \left[\begin{array}{cc:cc:c:cc} 0 & 1 & & & & & \\ -\omega_1^2 & 0 & & & & \bigcirc & \\ \hdashline & & 0 & 1 & & & \\ & & -\omega_2^2 & 0 & & & \\ \hdashline & & & & \ddots & & \\ \hdashline & \bigcirc & & & & 0 & 1 \\ & & & & & -\omega_N^2 & 0 \end{array}\right]$$

$$B = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \frac{1}{M}\phi_1(\epsilon_1) & \frac{1}{M}\phi_1(\epsilon_2) & \cdots & \frac{1}{M}\phi_1(\epsilon_m) \\ 0 & 0 & \cdots & 0 \\ \frac{1}{M}\phi_2(\epsilon_1) & \frac{1}{M}\phi_2(\epsilon_2) & \cdots & \frac{1}{M}\phi_2(\epsilon_m) \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \\ \frac{1}{M}\phi_N(\epsilon_1) & \frac{1}{M}\phi_N(\epsilon_2) & \cdots & \frac{1}{M}\phi_N(\epsilon_m) \end{bmatrix}$$

$$C = \begin{bmatrix} \phi_1(\alpha_1) & 0 & \phi_2(\alpha_1) & 0 & \cdots & \phi_N(\alpha_1) \\ \phi_1(\alpha_2) & 0 & \phi_2(\alpha_2) & 0 & \cdots & \phi_N(\alpha_2) \\ \vdots & & & & & \\ \phi_1(\alpha_p) & 0 & \phi_2(\alpha_p) & 0 & \cdots & \phi_N(\alpha_p) \end{bmatrix}$$

where the number of modes has been truncated at N, and the use of m force actuators at positions ϵ_j and p translation deflection sensors at positions ϵ_i have been assumed.

4.3 Failure detection filter design for the beam

For all the beam simulations that will appear later in this chapter, the actuators and sensors located, at 0.5 ft, 2.5 ft, and 6.0 ft from the end of the beam were used. The colocation was chosen simply as a matter of convenience, whereas the sensor located at 9.5 ft was not used since it is not output separable from the one located at 2.5 ft. In order for the state equations (4.2-5) to be a true representation of the beam, normalized mode shapes must be used. Thus, the orthogonal mode shapes in Table 4.1 were normalized according to (L = beam length)

$$\int_0^L \phi_i^2(x)\,dx = L \qquad (4.3\text{-}1)$$

before being used to compute A, B, and C. The orthogonal mode shape values at the actuator locations appear in Table 4.3.

For actuator failure detection, it was always possible to design a filter that could detect a failure of two actuators, but never for all three. This is because the individual detection orders are always two, but the group detection order for the three events is always n (the system order) making the events nonmutually detectable. The events are always output separable since the actuators are not symmetrically opposed with respect to the center of the beam. By extrapolation, it will be possible to detect up to p-1 input failure events with a single filter where p is the number of position sensors.

The design for sensor failures was made somewhat more difficult by the fact that when designing for two sensors, it was found that $A\underline{f}_1$ and $A\underline{f}_2$ were not output separable, with respect to (C',A'). This simply meant that D' only needed to be made a detector gain for either $A\underline{f}_1$ or $A\underline{f}_2$ and would automatically be a detector gain for the other. The rest of the design proceeded normally resulting in a filter that could detect two failure events. As in the actuator case, it was not possible to detect three sensor events with a single filter.

4.4 Computer simulation of beam

In order to simulate the dynamics of the system and filter, a fifth order Runge-Kutta integration routine was used. The step size was always maintained at a sufficiently small size compared to the period of the highest frequency mode being simulated in order not to misrepresent the continuity of the system dynamics. A listing of the beam simulation program appears in Appendix B.

The program accepts as input the detector gain D calculated by FDFIL and the nominal system (filter) matrix A, actuator effectiveness matrix BF, and measurement matrix CF used in computing D. BS and CS for the system are also initially set equal to the nominal matrices BF and CF, respectively. When the failure time, TFAIL, is reached, the column of BS associated with the failed actuator, or correspondingly, the row of CS associated with the failed sensor, is zeroed out to represent a failure in the off position. The values of BF and CF retain the nominal system values thus creating the driving term in the state error equation.

To easily verify whether the steady-state output error lies in the direction $C\underline{b}_i$ subsequent to the failure of the i^{th} actuator, the output is transformed so that any vector lying in the direction $C\underline{b}_i$ is changed to the direction $\hat{\underline{e}}_{p_i}$. The desired transformation then is

$$R = [(CF)^T CF]^{-1} (CF)^T \tag{4.4-1}$$

where F is the full matrix of event vectors $\underline{b}_i$. It is easily verified by direct substitution of (4.4-1) that $RC\underline{b}_i = \hat{\underline{e}}_{p_i}$ for i=1,2,3.

Similarly, it would be desirable to transform the output error for the i^{th} sensor event so that an error in the plane spanned by $\hat{\underline{e}}_{p_i}$ and $CK^{\ell}A\underline{f}_i$ will lie in an easily identifiable plane. Fortunately, the use of three sensors permits a geometrical construction of the desired transformation. Consider the plane spanned by $\hat{\underline{e}}_1$ and $\underline{w} = CK^{\ell}A\underline{f}_1$ in Fig. 4.3. This is the plane in which the output error is constrained to lie in the event of a sensor #1 failure. Now if the xyz coordinate system is rotated counterclockwise about the x-axis through the angle γ (the angle that the projection of $\underline{w}$ on the y-z plane makes with the y-axis), the ξ-η plane coincides with the plane spanned by $\hat{\underline{e}}_1$ and $\underline{w}$. Thus under the transformation

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix} \tag{4.4-2}$$

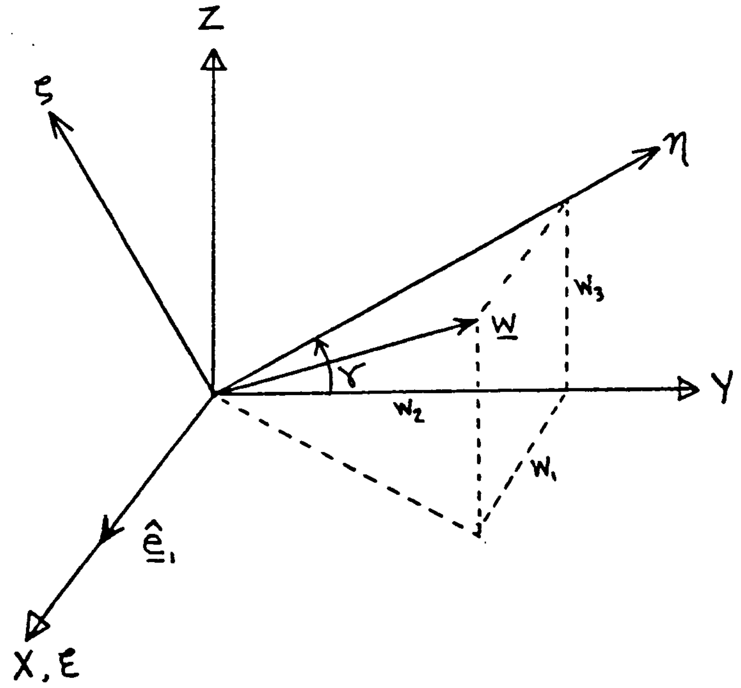


Fig. 4.3: Transformation of output error for sensor #1 failures.

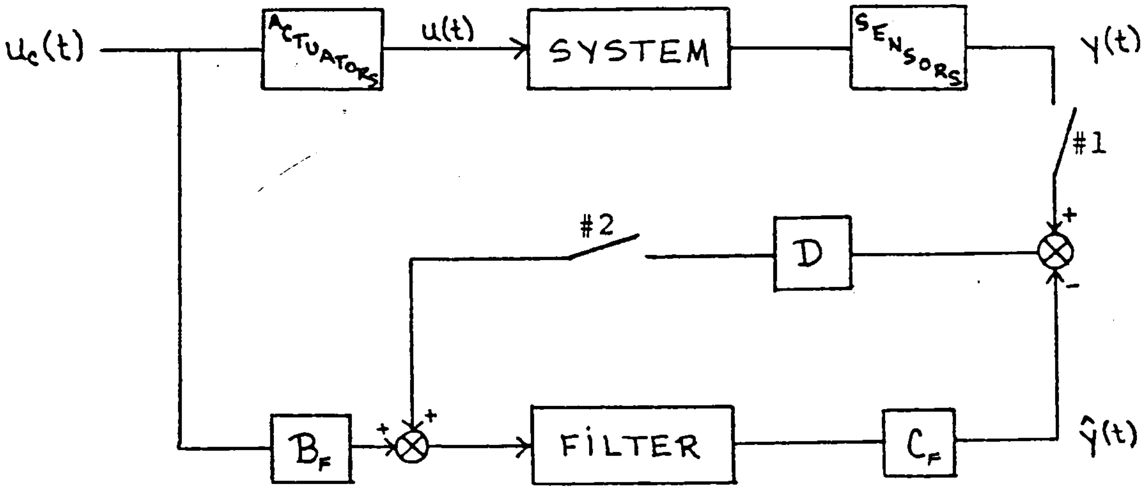


Fig. 4.4: Sampled-data system.

an error in the plane spanned by $\hat{\underline{e}}_1$ and $\underline{w}$ will have a first and second component but no third.

In the next two sections, the effect of model error in the filter and data sampling of the output will be investigated for the deterministic system. These two effects will be of the greatest concern in actual tests of the beam since the filter must necessarily incorporate a truncated model and data processor capability limits the sampling frequency. The effect of actuator, sensor, and plant noise will not be investigated here because they enter the realm of stochastic filter design which is beyond the scope of this work. Even after satisfying the requirements of being a detector gain, there is usually some freedom left in D that can be used to make the behavior of the detection filter approach that of the Kalman filter. Since no attempt was made here to optimize D for noisy systems under the constraints of being a detector gain, results of noise simulations would not reflect the true capabilities of the detection filter.

4.4.1 Effect of model error

Consider the case where the filter model contains fewer modes than the system model——this will always be the case for the actual beam. Let the state and output equations for the system be

$$\begin{bmatrix} \dot{\underline{x}}_M \\ \dot{\underline{x}}_U \end{bmatrix} = \left[\begin{array}{c:c} A_M & 0 \\ \hdashline 0 & A_U \end{array}\right] \begin{bmatrix} \underline{x}_M \\ \underline{x}_U \end{bmatrix} + \left[\begin{array}{c} B_M \\ \hdashline B_U \end{array}\right] \underline{u}$$

$$\underline{y} = \left[\begin{array}{c:c} C_M & C_U \end{array}\right] \begin{bmatrix} \underline{x}_M \\ \underline{x}_U \end{bmatrix} \qquad (4.4\text{-}3)$$

and for the filter

$$\dot{\hat{\underline{x}}}_M = A_M \hat{\underline{x}}_M + B_M \underline{u} + D(\underline{y} - \hat{\underline{y}}) \qquad (4.4\text{-}4)$$

$$\hat{\underline{y}} = C_M \hat{\underline{x}}_M$$

where the subscripts M and U denote modeled and unmodeled modes, respectively.

The new state error equations for the modeled modes are

$$\dot{\underline{e}}_M = (A_M - DC_M)\underline{e}_M - DC_U \underline{x}_U \qquad (4.4\text{-}5)$$

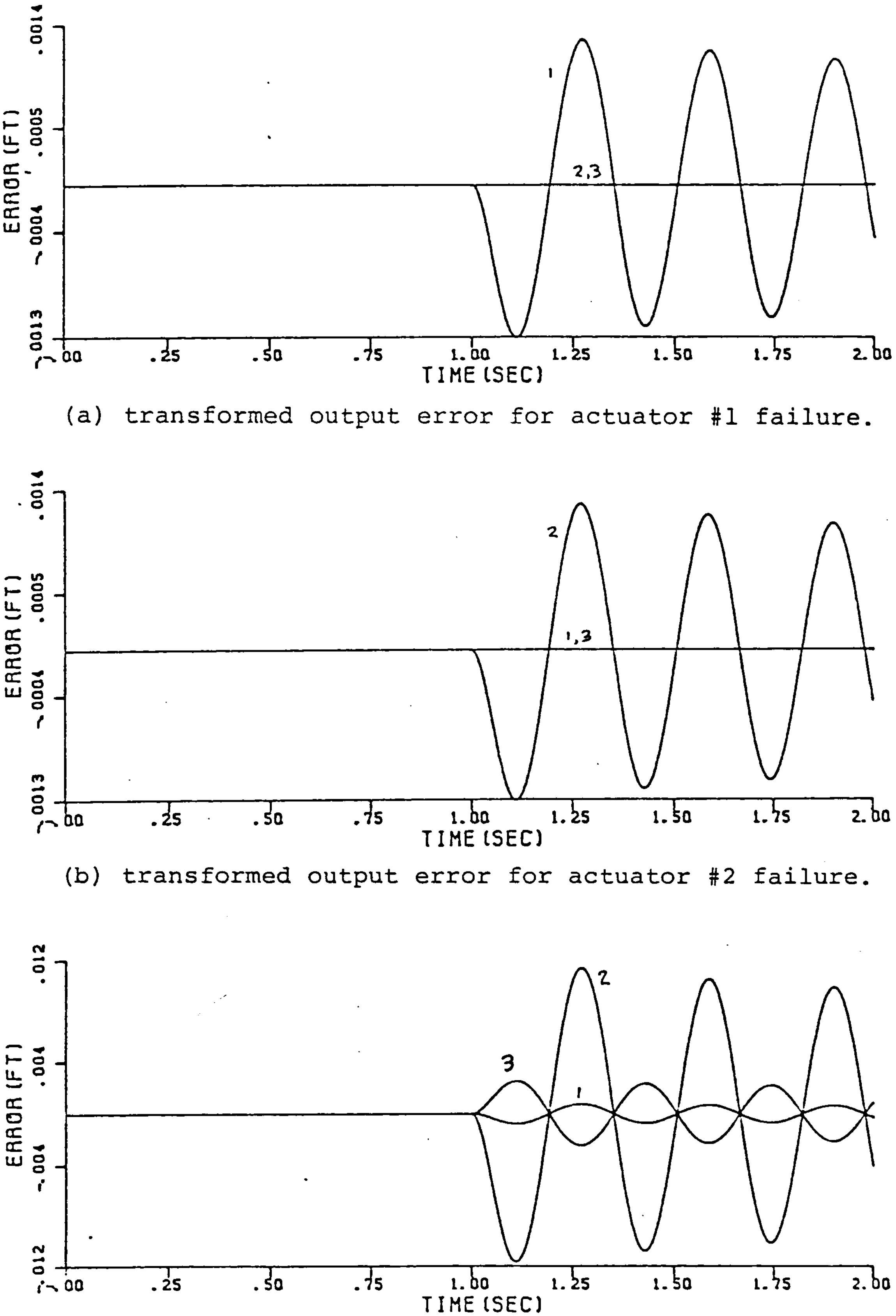
and the coupled equations for the system and filter become

$$\begin{bmatrix} \dot{\underline{x}}_M \\ \dot{\underline{x}}_U \\ \dot{\underline{e}} \end{bmatrix} = \left[\begin{array}{cc:c} A_M & 0 & 0 \\ 0 & A_U & 0 \\ \hdashline 0 & -DC_U & A_M - DC_M \end{array}\right] \begin{bmatrix} \underline{x}_M \\ \underline{x}_U \\ \underline{e} \end{bmatrix} + \left[\begin{array}{c} B_M \\ B_U \\ \hdashline 0 \end{array}\right] \underline{u} \qquad (4.4\text{-}6)$$

The effect of B_u and C_u on the system of (4.4-3) are known as control and observation spillover, respectively. Spillover was first pointed out by Balas (7) in his work on active control of flexible structures. B_u in (4.4-3) has the effect of driving the unmodeled modes, and C_u creates an extra driving term in the state error equation (4.4-5)—however, neither type of spillover can cause filter instability since the triangularity of the dynamics matrix in (4.4-6) is preserved.

The first two simulations performed were for a perfectly matched system and filter (no spillover present) in order to verify the filter design. The detection filter was set up to detect a failure of either actuator #1 at 0.5 ft from the end of the beam, or actuator #2 at 2.5 ft. Results for the simulation of failures of both actuators at 1.0 sec are shown in Fig. 4.5 for a five-mode model and 15 rad/sec filter bandwidth. All three actuators were driving the system with an exponentially decaying sinusoid of the form, $e^{-0.2t} \sin \omega_u t$, with an input frequency of ω_u = 20 rad/sec. The three transformed output errors are plotted on the same graph with the nonzero-signal in Fig. 4.5a representing the failure signal corresponding to the direction $\hat{\underline{e}}_1$ and, in Fig. 4.5b, the direction $\hat{\underline{e}}_2$. The verification is good to within the number of significant figures in the detector gain used in the simulation.

The effect of model error is dramatically illustrated in Fig. 4.6 for failure of the first actuator. Actuators #1,2,3 were operating at 20, 5, and 50 rad/sec, respectively, and the filter bandwidth was 15 rad/sec. Figure 4.6a is for a four-mode filter model and eight-mode system. The failure signal is not distinguishable from the other signals because

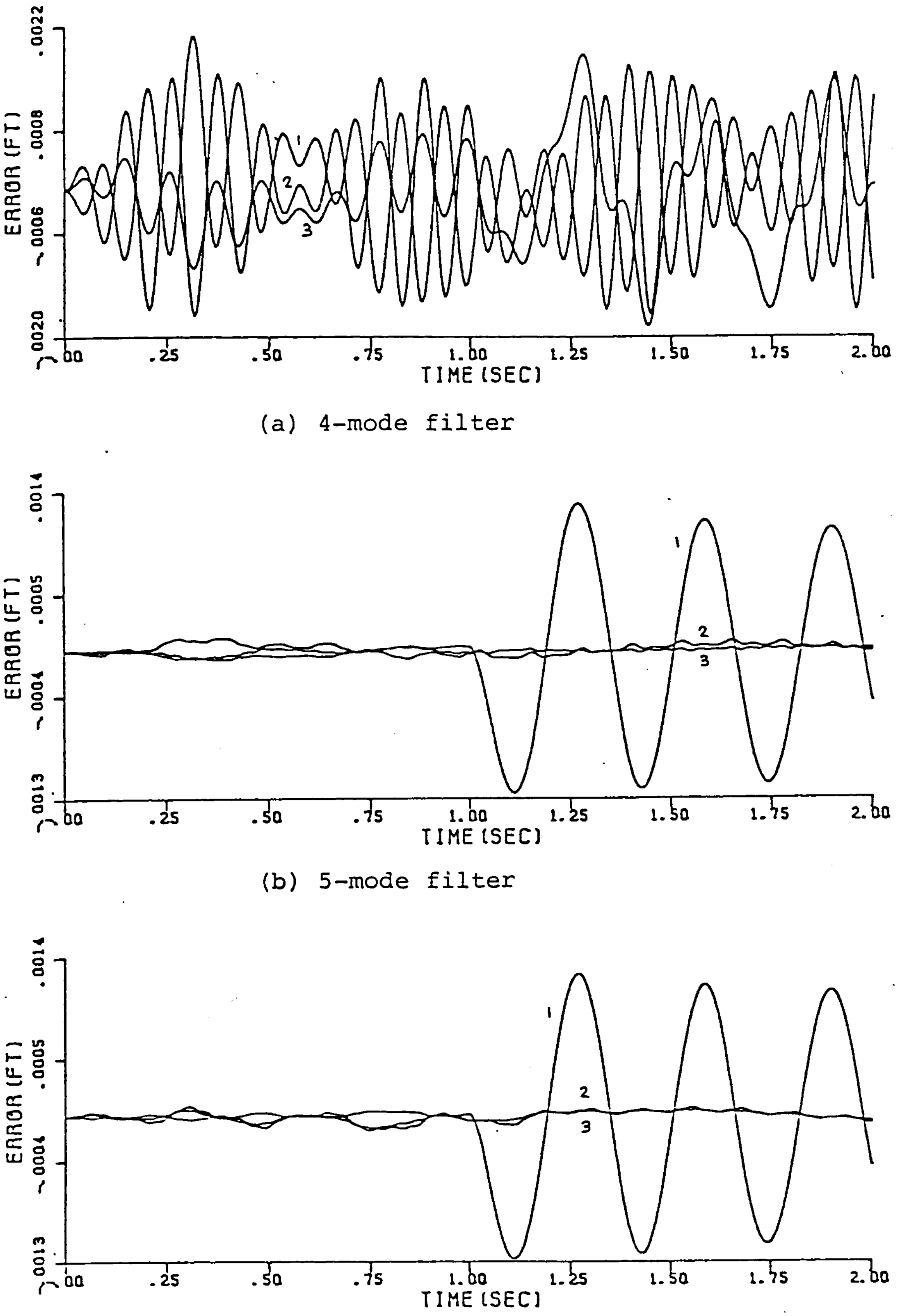


(a) transformed output error for actuator #1 failure.

(b) transformed output error for actuator #2 failure.

(c) real output error for actuator #2 failure.

Figs.4.5a-c: Actuator failure for matched models.



(a) 4-mode filter

(b) 5-mode filter

(c) 6-mode filter

Figs.4.6a-c: Model error effect for different filter models.

the unmodeled fifth mode with a natural frequency of 61 rad/sec is being excited heavily by the input frequency of 50 rad/sec. This excitation is much less for the five-mode filter of Fig. 4.6b since the mode at 61 rad/sec is now included in the model, and the effect in Fig. 4.6c for the six-mode filter is barely noticeable. Therefore, one should include in his model at least the first mode whose natural frequency lies beyond the highest input frequency.

The results of an eight-mode system/five-mode filter set of tests for various input frequencies and bandwidths are shown in Table 4.2. The peak failure signal for actuator #1 is given along with the peak signal-to-noise ratio—the peak noise is the peak signal achieved in direction $\hat{\underline{e}}_2$ and $\hat{\underline{e}}_3$. It appears that for a constant filter bandwidth, the performance of the filter drops off sharply as the input frequency is pushed farther past the filter poles. This is due to faster attenuation of the failure signal than the spillover noise outside the filter bandwidth. For a fixed input frequency, the signal-to-noise ratio reaches a maximum of 45 for a bandwidth of 15 rad/sec and input frequency of 5 rad/sec and a maximum of 36 for a bandwidth of 25 rad/sec and ω_u = 20 rad/sec. Results for three filter bandwidths and three input frequencies are plotted in Figs. 4.7 to 4.9.

It is interesting to note than when the input frequency is being held constant, the trends we are observing are for

TABLE 4.2: Peak failure signals and signal-to-noise ratios for various input frequencies and filter bandwidths in actuator failure tests.

FILTER POLES (rad/sec)	INPUT FREQUENCY (rad/sec)					
	5		20		50	
	*PS	+S/N	PS	S/N	PS	S/N
5	.0158	16	.00181	4	-	-
10	.00595	40	.00150	21	.000284	5
15	.00297[x]	45	.00119	31	.000255	5
20	.00175	41	.000925	35	.000238	5
25	.00114	35	.000709	36	-	-
30	.000804	29	.000562	23	-	-

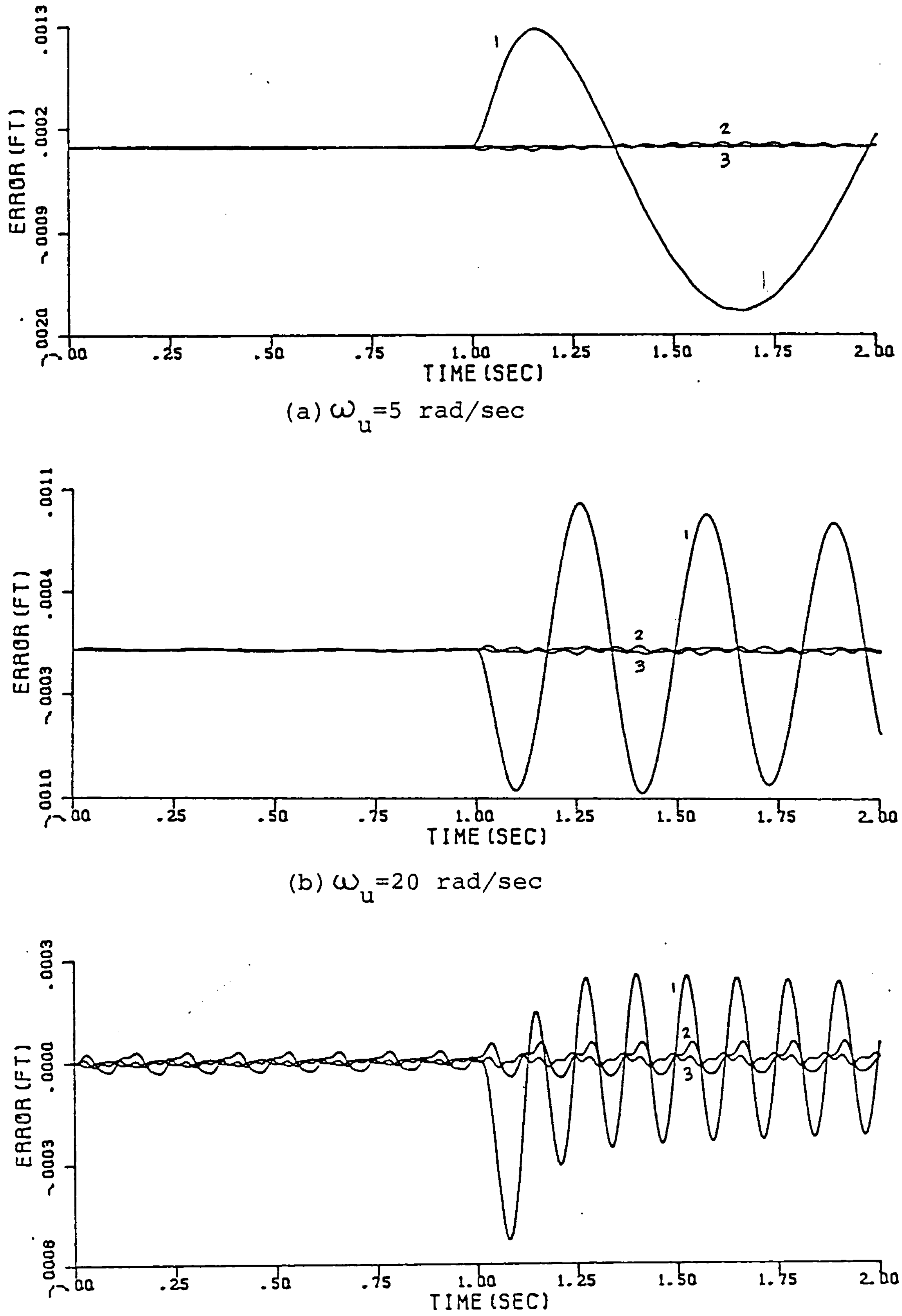
*peak failure signal in 1.5-2.0 sec interval

+peak failure signal divided by peak noise in other two output directions (1.5-2.0 sec)

[x] the corresponding results with the detector gain designed for detecting only actuator#1 failures were: PS=0.00121,S/N=1.5

TABLE 4.3: Normalized mode shape values at sensor and actuator locations (taken from reference[5]).

MODE SHAPE	MODE SHAPE VALUE at x=0.5 ft	x=2.5 ft	x=6.0 ft
ϕ_1	1.00	1.00	1.00
ϕ_2	-1.59	-1.01	0.00
ϕ_3	-1.60	-0.123	1.21
ϕ_4	1.32	-0.876	0.00
ϕ_5	1.04	-1.30	1.41
ϕ_6	-0.753	1.09	0.00
ϕ_7	-0.465	0.356	1.40
ϕ_8	-0.181	-0.553	0.00

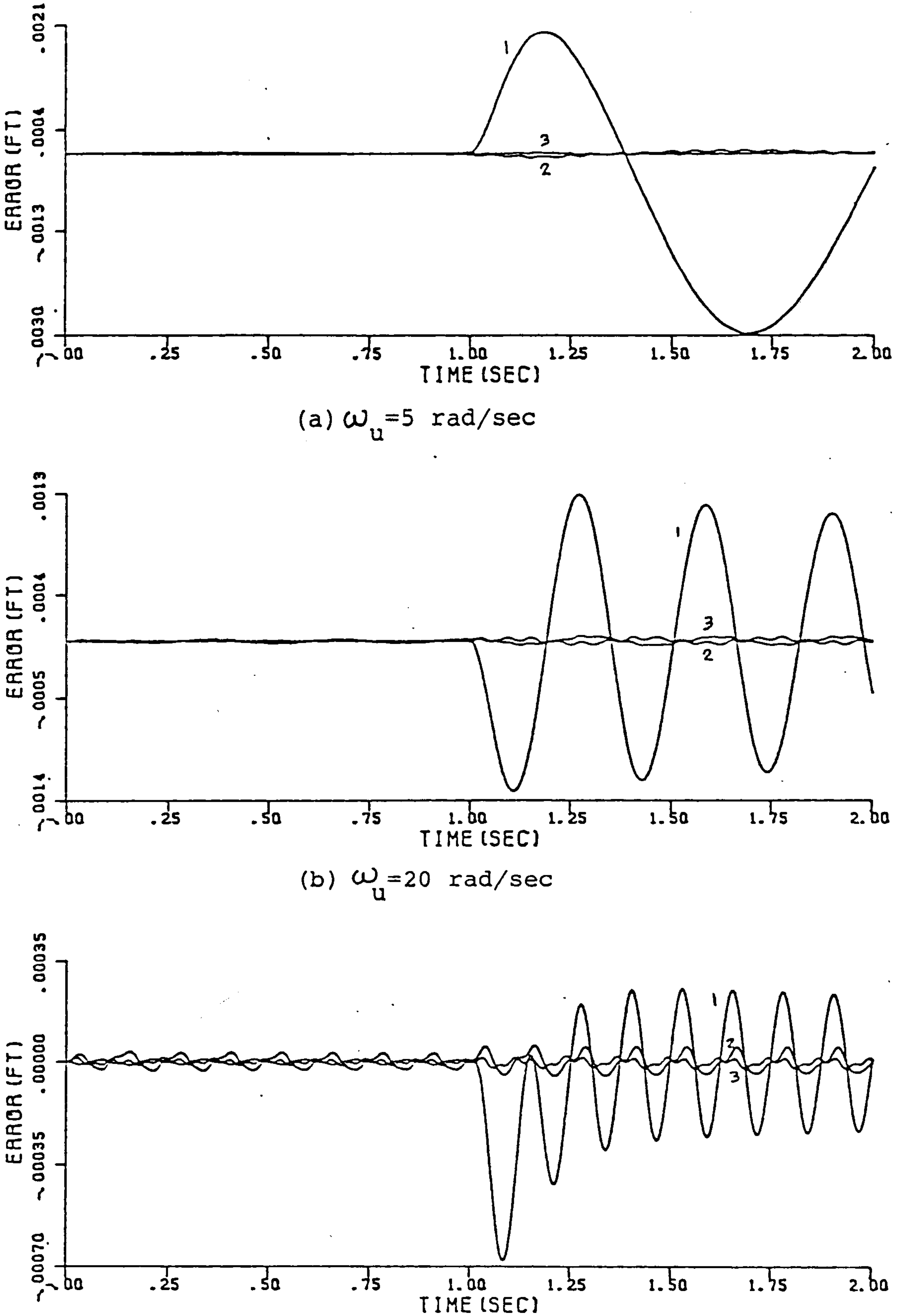


(a) ω_u=5 rad/sec

(b) ω_u=20 rad/sec

(c) ω_u=50 rad/sec

Figs.4.7a-c: Actuator tests for λ=-10 rad/sec.

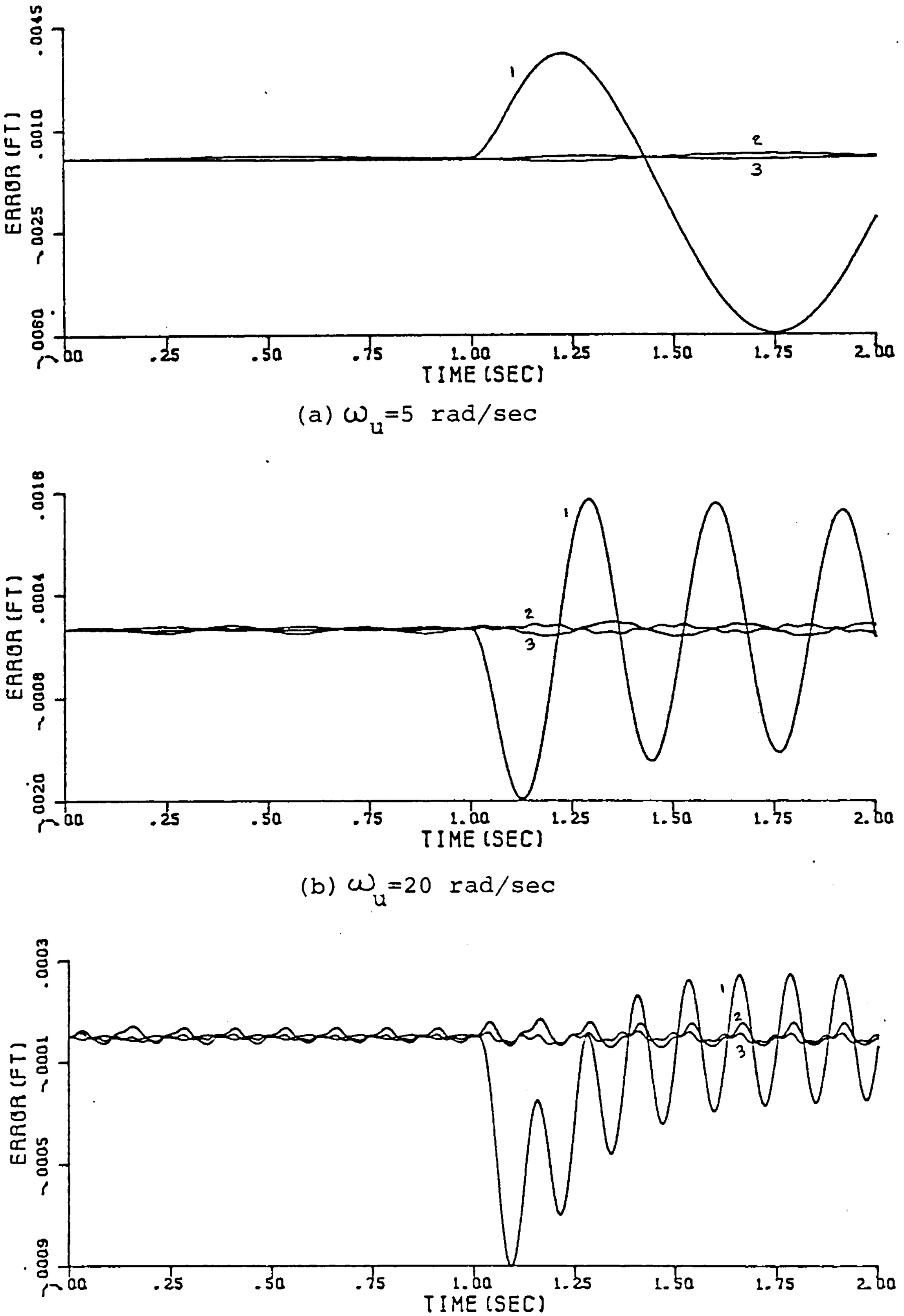


(a) ω_u=5 rad/sec

(b) ω_u=20 rad/sec

(c) ω_u=50 rad/sec

Figs.4.8a-c: Actuator tests for λ=-15 rad/sec.



(a) ω_u=5 rad/sec

(b) ω_u=20 rad/sec

(c) ω_u=50 rad/sec

Figs.4.9a-c: Actuator tests for λ=-20 rad/sec.

constant control spillover—however, the observation spillover coefficient DC_u in (4.4-5) changes since D and the filter poles change. But even if the filter poles are held constant and D is changed, the effects can be quite dramatic. For instance, if the detector gain is designed to monitor failures of just actuator #1 instead of both actuators, the signal-to-noise ratios for ω_u = 5 rad/sec between filter poles of 15 and 20 rad/sec are about two rather than 40 to 50. The failure signal itself, however, is relatively unaffected.

A set of simulations was also run for detection of complete failure (zero signal) of the sensor colocated with actuator #1. Instead of driving the system with actuator inputs, initial amplitudes were placed directly on the various modes in order to excite them. Since the output transformation for the sensor case is a pure rotation of coordinate axes, the magnitude of the outputs will actually be in feet if normalized mode shapes are used. Thus, the transformed output errors have physical significance, whereas their meaning was obscured by the transformation in the actuator case.

The results of the sensor failure simulations for filter poles at 15 rad/sec and varying initial conditions are given in Table 4.4 and Fig. 10. In all cases, the two rigid body modes had no initial conditions, and the amplitudes on the first three flex modes were 0.12 ft, 0.09 ft, and 0.06 ft, respectively. The corresponding filter initial conditions

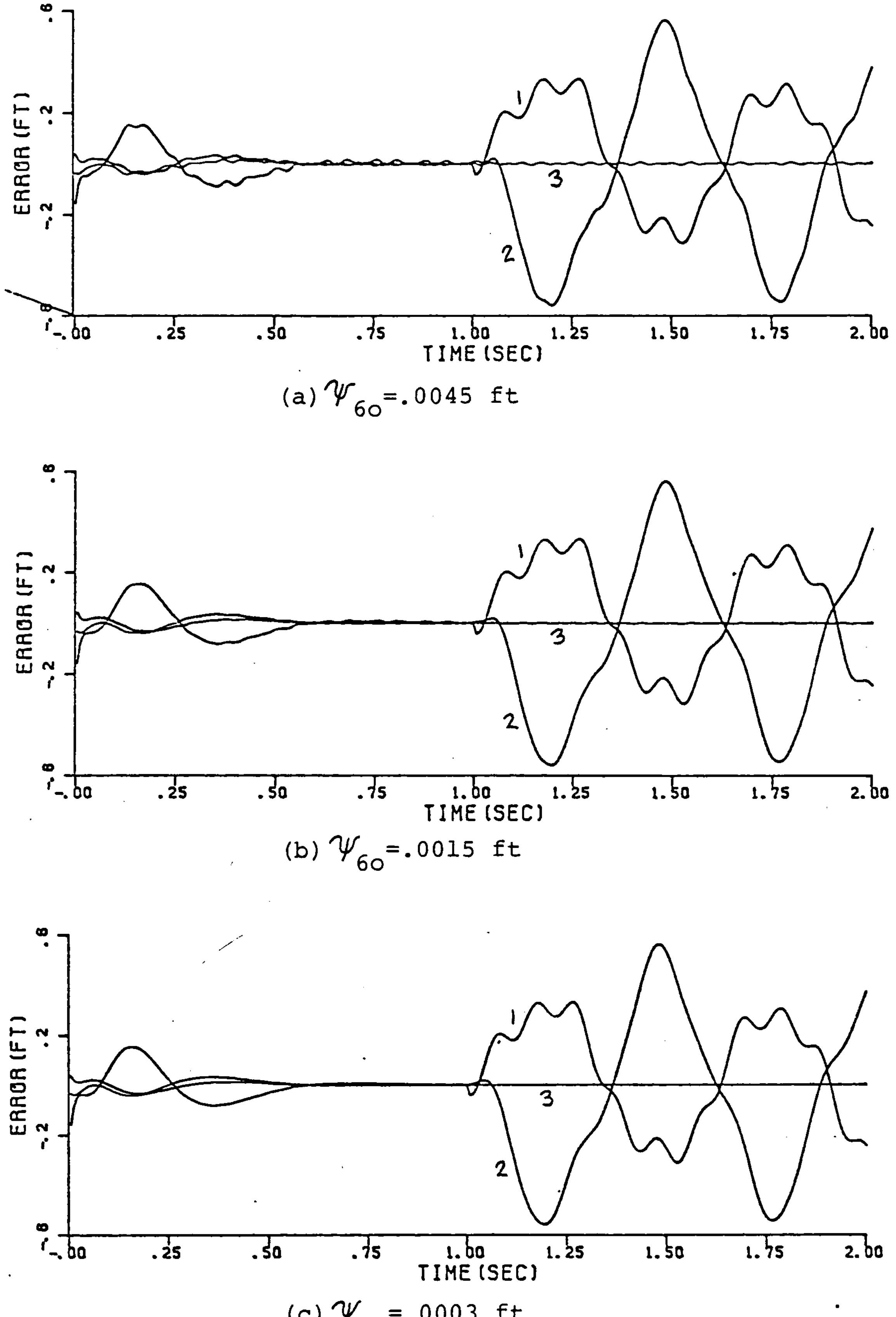
TABLE 4.4: Sensor failure results for filter poles at 15 rad/sec and various initial conditions.

INITIAL AMPLITUDES (ft)			PEAK OUTPUT SIGNALS (ft)			
ψ_{6o}	ψ_{7o}	ψ_{8o}	$^{+}PS_1$	$^{*}PS_2$	^{-}PN	$^{x}S/N$
.0045	.0030	.0015	.317	.561	$5.92x10^{-3}$	95
.0030	.0020	.0010	.317	.561	$3.95x10^{-3}$	142
.0015	.0010	.0005	.317	.561	$1.97x10^{-3}$	285
.0003	.0002	.0001	.317	.561	$3.13x10^{-4}$	1790
0	0	0	.317	.561	$5.36x10^{-6}$	10^{5}

TABLE 4.5: Sensor failure results for various filter bandwidths and fixed initial conditions (ψ_{6o}=.003ft, ψ_{7o}=.002ft, ψ_{8o}=.001ft).

FILTER POLES (rad/sec)	PEAK OUTPUT SIGNALS (ft)			
	$^{+}PS_1$	$^{*}PS_2$	^{-}PN	^{x} S/N
5	.781	1.191	.00810	147
10	.491	.800	.00456	175
15	.316	.561	.00395	142
20	.224	.396	.00394	100
25	.177	.286	.00288	99
30	.144	.211	.00287	74

$^{+}$peak signal in direction #1 (1.5-2.0 sec)
*same for direction #2
$^{-}$same for direction #3
^{x}peak signal in direction #1 and #2 divided by peak noise in direction #3

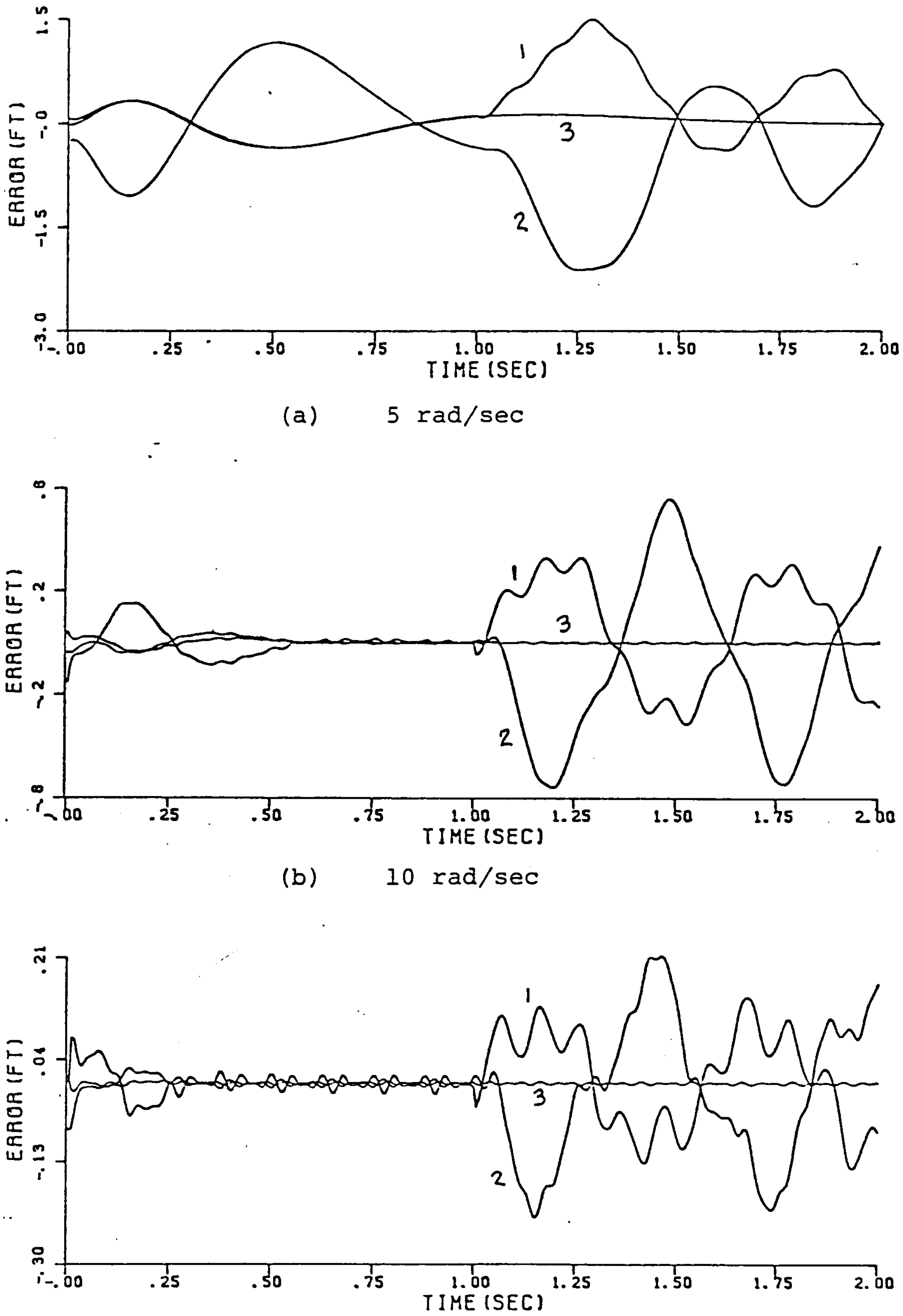


(a) ψ_{6o}=.0045 ft

(b) ψ_{6o}=.0015 ft

(c) ψ_{6o}=.0003 ft

Figs.4.10a-c: Sensor tests for various initial conditions.



(a) 5 rad/sec

(b) 10 rad/sec

(c) 20 rad/sec

Figs.4.11a-c: Sensor tests for various filter bandwidths.

were all zero. The amplitudes on the three unmodeled modes (next three flex modes) are given in the table. The failure signal plane is $\hat{\underline{e}}_1 \times \hat{\underline{e}}_2$. The model error noise is very sensitive to initial conditions on the unmodeled modes, varying almost proportionally. The failure signal itself though is unaffected.

Table 4.5 shows results when the amplitudes on the three unmodeled modes are held constant at 0.003, 0.002, and 0.001 ft and the filter bandwidth is varied. The signal-to-noise ratio reaches a peak near 10 rad/sec where the attenuation of the spillover noise relative to the failure signal is greatest. Between 25 and 30 rad/sec, there was almost no attenuation of noise whereas there was a 25% reduction in the failure signal resulting in loss of signal-to-noise ratio. Plots for several cases appear in Fig. 4.11.

4.4.2 Sampled-data systems

Instead of monitoring a system continuously, it is usually the case because of computational considerations that sensor data is taken at regular intervals and the output held constant between sampling periods. This scheme is depicted in Fig. 4.4 for the detection filter. The switch at #1 closes once every sampling period to receive an update, and the signal stays constant between sampling times. Switch #2 closes at the same instant switch #1 closes, and is shown in order to emphasize that the output error to the filter remains constant between sampling updates.

The effect of data sampling on filter performance in detection of actuator #1 failures for filter poles at 15 rad/sec and input frequencies of 20 rad/sec is shown in Table 4.6. As expected, the signal-to-noise ratio increases with sampling rate, but even at 160 hz, it is still a long way from the continuous sampling case where it is 31 (see Table 4.2). As illustrated in Fig. 4.12, the failure signal remains constant for different sampling rates, whereas the noise decreases for faster rates.

As a final illustration of data sampling, consider the case where the sampling rate is held constant at 64 hz, and filter bandwidth is varied. Table 4.7 shows results for filter poles between 5 and 20 rad/sec. At 5 rad/sec, the error signal due to spillover noise reached values nearly equal to the failure signal. Between 10 and 20 rad/sec the signal-to-noise ratio increased as it did for the continuous case. Finally, at 25 rad/sec the filter became unstable because the filter time constants were too fast for the sampled-error feedback to have a stabilizing effect. Plots for several cases appear in Fig. 4.13.

It is important to understand that these data-sampling results are for a case in which the detection filter was designed on a continuous data basis, and then data sampling

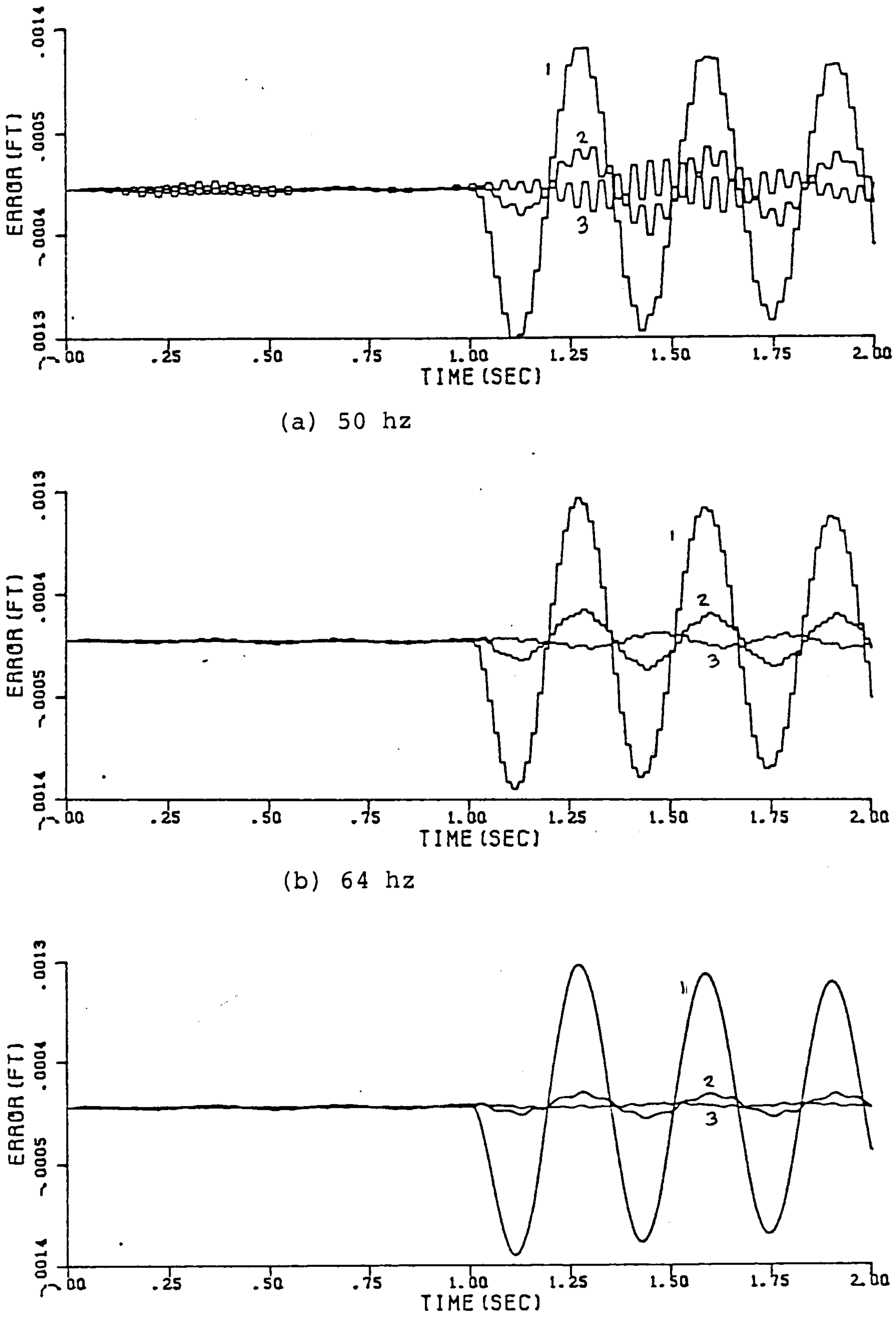
TABLE 4.6: Data-sampling results for input frequencies of 20 rad/sec, filter poles at 15 rad/sec, and various sampling rates.

SAMPLING FREQUENCY (hz)	PEAK OUTPUT ERROR SIGNALS ($\cdot 10^4$)			
	+PS	*PN$_2$	-PN$_3$	xS/N
50	11.7	3.41	2.21	3.4
64	11.7	2.17	0.70	5.4
96	11.7	1.53	0.43	7.6
160	11.7	1.02	0.25	11.5

TABLE 4.7: Data-sampling results for input frequencies of 20 rad/sec, 64 hz sampling rate, and various filter bandwidths.

FILTER POLES	PEAK OUTPUT ERROR SIGNALS ($\cdot 10^4$)			
(rad/sec)	PS	PN$_2$	PN$_3$	S/N
5	17.9	7.49	12.5	1.4
10	14.0	3.53	2.0	4.0
15	11.7	1.53	0.4	5.4
20	9.3	1.35	0.3	6.8

+peak signal in direction #1 (1.5-2.0 sec)
*same for direction #2
-same for direction #3
xpeak signal in direction #1 divided by peak signal in direction #2 and #3 (1.5-2.0 sec)

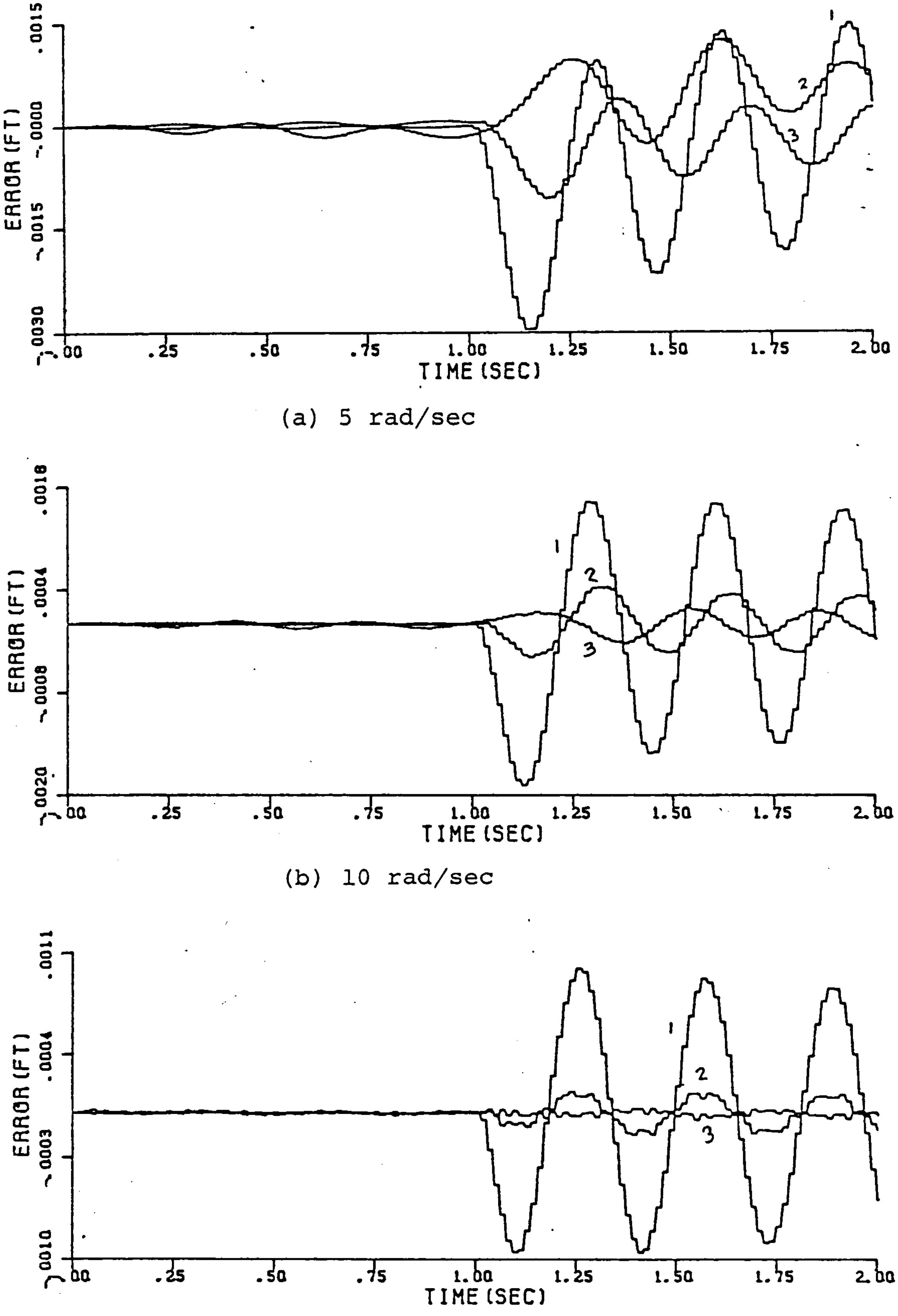


(a) 50 hz

(b) 64 hz

(c) 160 hz

Figs.4.12a-c: Data sampling tests for various sampling rates(filter bandwidth is 15 rad/sec).



(a) 5 rad/sec

(b) 10 rad/sec

(c) 20 rad/sec

Figs.4.13a-c: Data sampling tests for various filter bandwidths(sampling rate = 64 hz).

was imposed on the resulting filter. It is possible to design the failure detection filter on the basis of a sampled-data description of the dynamics. This protects against the instability observed in these results, but data sampling fundamentally limits the ability of the filter to restrict a continuous failure signature to a single direction or plane in the output space.

CHAPTER V

SUMMARY AND CONCLUSIONS

The failure detection filter performs very well in the context of flexible structure dynamics when the dynamic model in the filter matches that of the system it is monitoring. It was found that the filter usually works quite well even when the description of the system includes modes which are not modeled in the filter. This performance rapidly diminishes, however, as the input frequencies become close to the natural frequencies of the unmodeled modes. For a given input frequency, an optimum filter bandwidth for maximum signal-to-noise ratio usually exists not far beyond the input frequency.

It is not always the case, however, that the filter will perform satisfactorily in the presence of modeling error. Sometimes two different detector gains capable of detecting the same failure event and designed with the same filter poles will yield grossly different detection performances—in one case the signal-to-noise ratio might be 50 and in the other case it might be less by an order of magnitude. This effect is caused by the direct appearance of the detector gain in the filter input due to observation spillover, the same phenomenon encountered when designing control systems for flexible structures.

The filter worked well for sensor failures, even when the initial amplitudes on the unmodeled modes were a significant fraction of those on the modeled modes (up to 7.5% were tested).

By varying filter bandwidth, it was possible to obtain very high signal-to-noise ratios for fixed initial conditions.

As expected, satisfactory performance of the filter designed for continuous data processing when used in a sampled-data mode depended strongly on a very high sampling rate. Only when the sampling rate reached five times the natural frequency of the last system mode did the signal-to-noise ratio surpass 10. As can be anticipated, when the sampling rate to filter bandwidth ratio is too low, the phase lag caused by sampling tended to destabilize the system.

These observations bring to mind several suggestions with regard to both the actual beam experiment and areas for further study:

(1) To ensure adequate filter performance, it may be necessary to eliminate observation spillover as much as possible—it might be possible to accomplish this using a phase lock loop[10] to comb out only the modeled modes from the output or by reducing the bandwidth of sensor data (one must be careful in doing this, however, not to attenuate the failure signal as well).

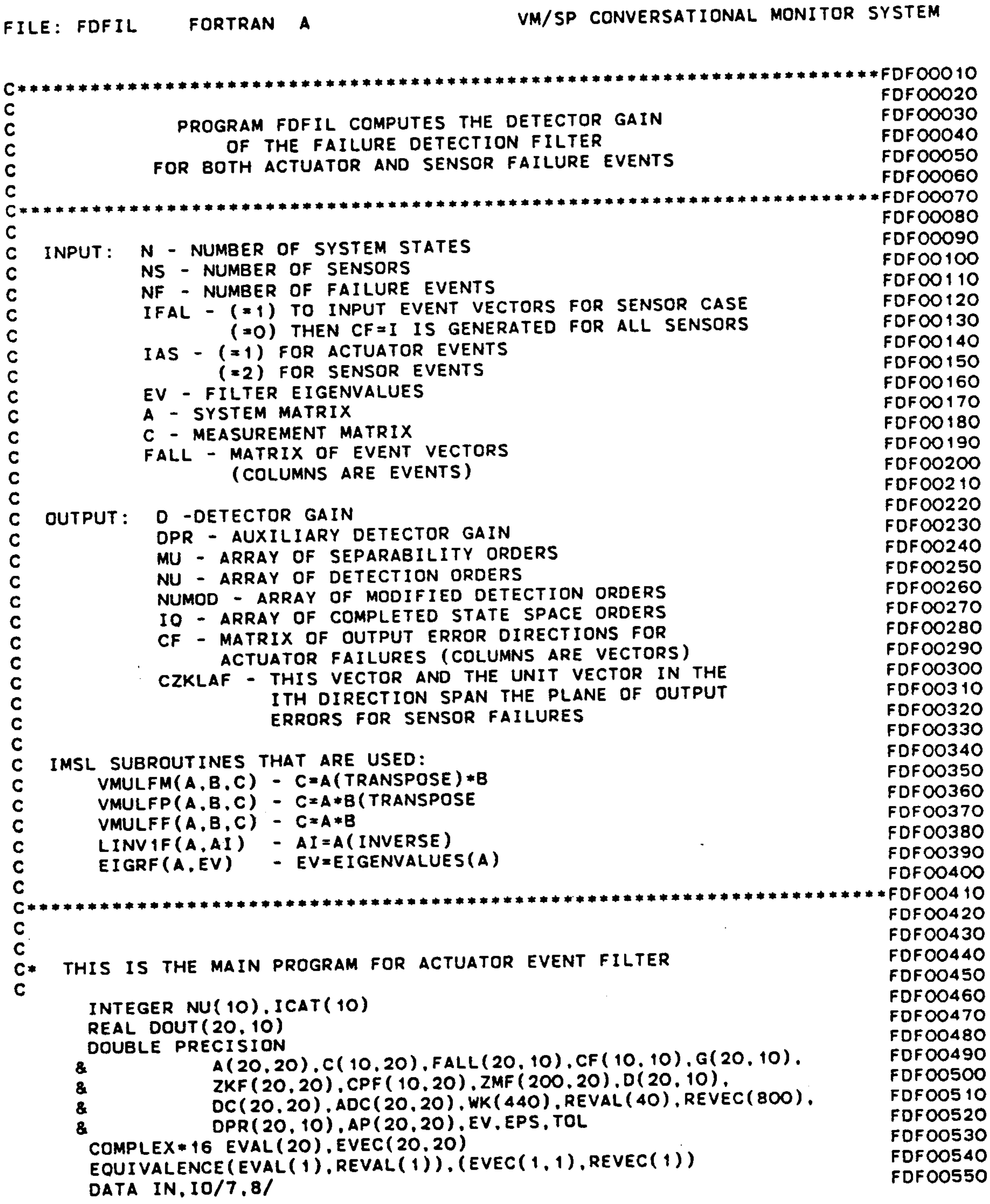
(2) It may be necessary to use a sampling rate of at least 64 hz in order to get stable and adequate performance from the filter—this will depend upon the degree of excitation of the higher modes.

(3) Even after specifying D to be a detector gain and assign-

ing all the filter eigenvalues, there is still a lot of freedom left in D (especially when D is designed to detect only one event) that could be used to reduce spillover noise—using this freedom would require a more analytic approach to solving for D and involve solving a set of nonlinear simultaneous equations when assigning eigenvalues.

(4) The whole issue of stochastic failure detection filter theory has not even been addressed here—this will be an additional consideration in designing the filter when optimizing signal-to-noise ratios in the presence of plant, actuator, and sensor noise.

APPENDIX A



```
FILE: FDFIL    FORTRAN  A                 VM/SP CONVERSATIONAL MONITOR SYSTEM

C**********************************************************************
C
C              PROGRAM FDFIL COMPUTES THE DETECTOR GAIN
C                  OF THE FAILURE DETECTION FILTER
C            FOR BOTH ACTUATOR AND SENSOR FAILURE EVENTS
C
C**********************************************************************
C
C  INPUT:   N - NUMBER OF SYSTEM STATES
C          NS - NUMBER OF SENSORS
C          NF - NUMBER OF FAILURE EVENTS
C          IFAL - (=1) TO INPUT EVENT VECTORS FOR SENSOR CASE
C                 (=0) THEN CF=I IS GENERATED FOR ALL SENSORS
C          IAS - (=1) FOR ACTUATOR EVENTS
C                (=2) FOR SENSOR EVENTS
C          EV - FILTER EIGENVALUES
C          A - SYSTEM MATRIX
C          C - MEASUREMENT MATRIX
C          FALL - MATRIX OF EVENT VECTORS
C                 (COLUMNS ARE EVENTS)
C
C  OUTPUT:   D -DETECTOR GAIN
C            DPR - AUXILIARY DETECTOR GAIN
C            MU - ARRAY OF SEPARABILITY ORDERS
C            NU - ARRAY OF DETECTION ORDERS
C            NUMOD - ARRAY OF MODIFIED DETECTION ORDERS
C            IO - ARRAY OF COMPLETED STATE SPACE ORDERS
C            CF - MATRIX OF OUTPUT ERROR DIRECTIONS FOR
C                 ACTUATOR FAILURES (COLUMNS ARE VECTORS)
C            CZKLAF - THIS VECTOR AND THE UNIT VECTOR IN THE
C                     ITH DIRECTION SPAN THE PLANE OF OUTPUT
C                     ERRORS FOR SENSOR FAILURES
C
C  IMSL SUBROUTINES THAT ARE USED:
C      VMULFM(A,B,C) - C=A(TRANSPOSE)*B
C      VMULFP(A,B,C) - C=A*B(TRANSPOSE
C      VMULFF(A,B,C) - C=A*B
C      LINV1F(A,AI)  - AI=A(INVERSE)
C      EIGRF(A,EV)   - EV=EIGENVALUES(A)
C
C**********************************************************************
C
C
C*  THIS IS THE MAIN PROGRAM FOR ACTUATOR EVENT FILTER
C
      INTEGER NU(10),ICAT(10)
      REAL DOUT(20,10)
      DOUBLE PRECISION
     &          A(20,20),C(10,20),FALL(20,10),CF(10,10),G(20,10),
     &          ZKF(20,20),CPF(10,20),ZMF(200,20),D(20,10),
     &          DC(20,20),ADC(20,20),WK(440),REVAL(40),REVEC(800),
     &          DPR(20,10),AP(20,20),EV,EPS,TOL
      COMPLEX*16 EVAL(20),EVEC(20,20)
      EQUIVALENCE(EVAL(1),REVAL(1)),(EVEC(1,1),REVEC(1))
      DATA IN,IO/7,8/
      READ(IN,10) N,NS,NF,IFAL,IAS,EV
      READ(IN,20) ((A(I,J),J=1,16),I=1,16),
     &            ((C(I,J),J=1,16),I=1,8),
     &            ((FALL(I,J),J=1,8),I=1,16)
   10 FORMAT(5I5/E14.4)
   20 FORMAT(32(8F10.3/),16(8F10.3/),15(8F10.3/),8F10.3)
      WRITE(IO,25) N,NS,NF,IFAL,IAS,EV
   25 FORMAT('N=',I2,5X,'NS=',I2,5X,'NF=',I2,5X,'IFAL=',I2,5X,
     &       'IAS=',I2/'EV=',E14.4/)
      WRITE(IO,26) ((A(I,J),J=1,8),I=1,8),
     &             ((C(I,J),J=1,8),I=1,8),
     &             ((FALL(I,J),J=1,8),I=1,8)
   26 FORMAT(//'A='/8(8F10.4/)//'C='/8(8F10.4/)//'FALL='/8(8F10.4/))
      IF(IAS.EQ.2) GO TO 27
      CALL SEPDET(A,C,FALL,N,NF,NS,ISEP,IDET,IDIR,C,CF,G,NU,NUMOD,
     &            ZKF,CPF,ZMF)
      IF((ISEP.EQ.1).OR.(IDET.EQ.1)) GO TO 100
      CALL DGAIN(C,A,EV,G,CF,CPF,NU,NUMOD,N,NF,NS,0,0,DPR,
     &           ZKF,ZMF,AP,D)
      GO TO 28
C
C*  CALL THE MAIN PROGRAM FOR SENSOR EVENT FILTER
C
   27 CALL SENSOR(C,A,EV,G,NU,CF,CPF,N,NF,NS,
     &            ZKF,ZMF,FALL,IFAL,IDET,ICAT,D)
       DO 271 I=1,10
         IF(ICAT(I).EQ.3) STOP
  271  CONTINUE
      IF(IDET.EQ.0) GO TO 28
      WRITE(IO,277) IDET,((FALL(I,J),J=1,4),I=1,6)
  277 FORMAT('IDET=',I1//'FALL='/6(4E14.4/)//)
      GO TO 200
   28 CALL VMULFF(D,C,N,NS,N,20,10,DC,20,IER)
      CALL MATSUB(A,DC,20,20,N,N,ADC)
          WRITE(IO,29) ((ADC(I,J),J=1,4),I=1,4)
29        FORMAT('A-DC='/4(4E14.4/)/)
C*  VERIFY THE PLACEMENT OF FILTER EIGENVALUES
      CALL EIGRF(ADC,N,20,0,REVAL,REVEC,20,WK,IER)
      WRITE(IO,30) (EVAL(I),I=1,16)
   30 FORMAT('CLOSED-LOOP EVS ARE: '/16(2F10.4/)/)
      WRITE(IO,40) ((D(I,J),J=1,3),I=1,16)
   40 FORMAT('GAIN MATRIX D='/16(3E16.6/)//)
      DO 48 I=1,N
        DO 48 J=1,NS
          DOUT(I,J)=SNGL(D(I,J))
   48 CONTINUE
      WRITE(11,50) ((DOUT(I,J),J=1,3),I=1,16)
   50 FORMAT(15(3E16.6/),3E16.6)
      GO TO 200
  100 WRITE(IO,110) ISEP,IDET
  110 FORMAT('THE SYSTEM IS NOT SEPARABLE OR DETECTABLE:'/
     &       'ISEP=',I1,10X,'IDET=',I1)
  200 STOP
      END
C
```

```
C*   SEPDET DETERMINES SEPARABILITY AND DETECTABILITY OF EVENTS
C
        SUBROUTINE SEPDET(A,C,FALL,N,NF,NS,ISEP,IDET,IDIR,CSYS,CF,G,NU,
      &                  NUMOD,ZKF,CPF,ZMF)
       INTEGER MU(10),NU(10),IQ(10),NUMOD(10)
       DOUBLE PRECISION
      &          A(20,20),C(10,20),FALL(20,10),CF(10,10),CPF(10,20),
      &          G(20,10),ZKF(20,20),ZMF(200,20),F(20),CAJ(10,20),
      &          CAJF(10),OM(10,10),OMSUB(10,10),WL(20),WF(20,10),
      &          CFTC(10,20),CFTCF(10,10),CFTCFI(10,10),WKAR(460),
      &          CFTTI(10,20),AFALL(20,10),AFCF(20,20),CFCF(10,20),
      &          ZMFNUL(20,20),AJ(20,20),ZMFSUB(20,20),FALL2(20,10),
      &          ZMFSB(10,20),ZMFSB2(10,20),AJ2(20,20),AMUF(20),
      &          CSYS(10,20),EPS,TOL,CAJEPS,ZKNU1
       DATA EPS,IO/1.D-4,8/
       ISEP=0
       IDET=0
       DO 3 I=1,N
         DO 3 J=1,NF
           FALL2(I,J)=FALL(I,J)
     3 CONTINUE
       DO 20 K=1,NF
         DO 5 I=1,N
           F(I)=FALL(I,K)
     5   CONTINUE
         MU(K)=0
         CALL MATID(AJ,20,N)
     6   CALL VMULFF(C,AJ,NS,N,N,10,20,CAJ,10,IER)
         CALL MATVEC(CAJ,F,10,20,NS,N,CAJF)
         TOL=0.D0
         DO 7 I=1,NS
           DO 7 J=1,N
             CAJEPS=CAJ(I,J)*EPS
             TOL=DABS(DMAX1(TOL,CAJEPS))
     7   CONTINUE
         NZERO=0
         DO 10 I=1,NS
           IF(DABS(CAJF(I)).LT.TOL) NZERO=NZERO+1
    10   CONTINUE
         IF(NZERO.NE.NS) GO TO 12
         MU(K)=MU(K)+1
         CALL VMULFF(A,AJ,N,N,N,20,20,AJ2,20,IER)
         DO 11 I=1,N
           DO 11 J=1,N
             AJ(I,J)=AJ2(I,J)
    11   CONTINUE
         GO TO 6
    12     WRITE(IO,13) K,MU(K)
    13     FORMAT('K=',I2,5X,'MU=',I2)
           CALL MATVEC(AJ,F,20,20,N,N,AMUF)
         DO 15 I=1,N
           FALL(I,K)=AMUF(I)
           CF(I,K)=CAJF(I)
    15   CONTINUE
    20 CONTINUE
```

```
      WRITE(IO,205) ((CF(I,J),J=1,3),I=1,3)
205   FORMAT('CF='/3(3E16.6/)/)
      CALL MATID(OM,10,NS)
      CALL ORTRED(CF,10,10,NS,NF,OM,NS,0,OM,OMSUB,
     &           IRKCF,ICODE3,ZKNU1,WL,IQ,WF)
      WRITE(IO,207) IRKCF
  207 FORMAT('RANK CF IS ',I2)
      IF(IRKCF.EQ.NF) GO TO 21
      ISEP=1
      WRITE(IO,209)
209   FORMAT('THE EVENTS ARE NOT SEPARABLE')
      RETURN
   21 CALL VMULFM(CF,C,NS,NF,N,10,10,CFTC,10,IER)
      CALL VMULFM(CF,CF,NS,NF,NF,10,10,CFTCF,10,IER)
      CALL LINV1F(CFTCF,NF,10,CFTCFI,0,WKAR,IER)
      CALL VMULFF(CFTCFI,CFTC,NF,NF,N,10,10,CFTTI,10,IER)
      CALL VMULFF(A,FALL,N,N,NF,20,20,AFALL,20,IER)
      CALL VMULFF(AFALL,CFTTI,N,NF,N,20,10,AFCF,20,IER)
      CALL MATSUB(A,AFCF,20,20,N,N,ZKF)
      CALL VMULFF(CF,CFTTI,NS,NF,N,10,10,CFCF,10,IER)
      CALL MATSUB(C,CFCF,10,20,NS,N,CPF)
      CALL MATGEN(CPF,ZKF,10,20,200,NS,N,N,ZMFSB,ZMFSB2,ZMF)
          WRITE(IO,303) ((ZKF(I,J),J=1,4),I=1,4),
     &                  ((CPF(I,J),J=1,4),I=1,3)
303           FORMAT('ZKF='/4(4E14.4/)//'CPF='/3(4E14.4/)//)
      CALL MATID(ZMFNUL,20,N)
      CALL ORTRED(ZMF,200,20,NS*N,N,ZMFNUL,NS,0,ZMFNUL,ZMFSUB,
     &            IRKZMF,ICODE3,ZKNU1,WL,IQ,WF)
      CALL DETGEN(MU,FALL,FALL2,IDIR,CSYS,A,C,N,NS,NF,ZKF,NU,NUMOD,G)
        WRITE(IO,315) IRKZMF,(NU(I),I=1,2)
315     FORMAT('IRKZMF=',I2,10X,'NU1-2=',2I5//)
      NUG=N-IRKZMF
      INUSUM=0
      DO 25 I=1,NF
        INUSUM=INUSUM+NU(I)
   25 CONTINUE
      IF(INUSUM.EQ.NUG) GO TO 30
      IDET=1
   30 RETURN
      END
C
C*  DETGEN FINDS THE DETECTION ORDER AND GENERATOR FOR THE EVENTS
C
      SUBROUTINE DETGEN(MU,FALL,FALL2,IDIR,CSYS,A,C,N,NS,NF,ZKF,
     &                  NU,NUMOD,G)
      INTEGER MU(10),NU(10),IQ(10),NUMOD(10),LEXP(10)
      DOUBLE PRECISION
     &          FALL(20,10),A(20,20),C(10,20),CSUB(10,20),
     &          G(20,10),F(20),CSF(10),CSFTC(20),CT(20,10),
     &          ASF(20),ASUB(20,20),ZK(20,20),CP(10,20),ZMP(200,20),
     &          ZMK(200,20),ZMKNUL(10,10),CI(20),AMU(20,20),
     &          WL(20),WF(20,10),ZKF(20,20),ZMPNUL(20,20),ZMKSUB(20,20),
     &          ZMPSUB(20,20),ZMPSB(10,20),ZMKSB(10,20),
     &          ZMPSB2(10,20),ZMKSB2(10,20),AMU2(20,20),
     &          AF(20),FALL2(20,10),ZKL(20,20),CZKL(10,20),
     &          CZKLAF(10),ZKL2(20,20),CSYS(10,20),CZKEPS,               FDF02210
     &          CFTCF,ZKNU1,CIAMUF,EPS,TOL                               FDF02220
      DATA IO,EPS/8,1.D-4/                                               FDF02230
      DO 45 K=1,NF                                                       FDF02240
        DO 5 I=1,N                                                       FDF02250
          F(I)=FALL(I,K)                                                 FDF02260
    5   CONTINUE                                                         FDF02270
        CALL MATVEC(C,F,10,20,NS,N,CSF)                                  FDF02280
        DO 3 I=1,NS                                                      FDF02290
          DO 3 J=1,N                                                     FDF02300
            CT(J,I)=C(I,J)                                               FDF02310
    3   CONTINUE                                                         FDF02320
        CALL MATVEC(CT,CSF,20,10,N,NS,CSFTC)                             FDF02330
        CFTCF=0.                                                         FDF02340
        DO 10 I=1,NS                                                     FDF02350
          CFTCF=CFTCF+CSF(I)**2                                          FDF02360
   10   CONTINUE                                                         FDF02370
        DO 15 I=1,N                                                      FDF02380
          CSFTC(I)=CSFTC(I)/CFTCF                                        FDF02390
   15   CONTINUE                                                         FDF02400
        CALL MATVEC(A,F,20,20,N,N,ASF)                                   FDF02410
        DO 17 I=1,N                                                      FDF02420
          DO 17 J=1,N                                                    FDF02430
            ASUB(I,J)=ASF(I)*CSFTC(J)                                    FDF02440
   17   CONTINUE                                                         FDF02450
        DO 18 I=1,NS                                                     FDF02460
          DO 18 J=1,N                                                    FDF02470
            CSUB(I,J)=CSF(I)*CSFTC(J)                                    FDF02480
   18   CONTINUE                                                         FDF02490
        CALL MATSUB(A,ASUB,20,20,N,N,ZK)                                 FDF02500
        CALL MATSUB(C,CSUB,10,20,NS,N,CP)                                FDF02510
        CALL MATGEN(CP,ZK,10,20,200,NS,N,N,ZMPSB,ZMPSB2,ZMP)             FDF02520
        CALL MATID(ZMPNUL,20,N)                                          FDF02530
        CALL ORTRED(ZMP,200,20,NS*N,N,ZMPNUL,NS,0,ZMPNUL,ZMPSUB,         FDF02540
     &              IRKZMP,ICODE3,ZKNU1,WL,IQ,WF)                        FDF02550
          WRITE(IO,206) IRKZMP                                           FDF02560
206       FORMAT('RANK OF ZMP IS ',I2)                                   FDF02570
        NU(K)=N-IRKZMP                                                   FDF02580
        CALL MATGEN(C,ZK,10,20,200,NS,N,NU(K),ZMKSB,ZMKSB2,ZMK)          FDF02590
        CALL ORTRED(ZMK,200,20,NS*N,N,ZMPNUL,NS,0,ZMKNUL,ZMKSUB,         FDF02600
     &              IRKZMK,ICODE3,ZKNU1,WL,IQ,WF)                        FDF02610
        NUMOD(K)=IRKZMK                                                  FDF02620
          WRITE(IO,216) NUMOD(K)                                         FDF02630
216       FORMAT('NUMOD=',I2)                                            FDF02640
        DO 20 J=1,N                                                      FDF02650
          CI(J)=C(ICODE3,J)                                              FDF02660
   20   CONTINUE                                                         FDF02670
        CIAMUF=0.D0                                                      FDF02680
        DO 25 I=1,N                                                      FDF02690
          CIAMUF=CIAMUF+CI(I)*F(I)                                       FDF02700
   25   CONTINUE                                                         FDF02710
        DO 40 I=1,N                                                      FDF02720
          G(I,K)=(CIAMUF/ZKNU1)*WL(I)                                    FDF02730
   40   CONTINUE                                                         FDF02740
   43     WRITE(IO,225) (G(I,K),I=1,4)                                   FDF02750
```

```
225       FORMAT('G='/4E14.4)
        IF(IDIR.NE.1) GO TO 45
        LEXP(K)=0
        DO 441 I=1,N
          AF(I)=FALL2(I,K)
  441   CONTINUE
        CALL MATID(ZKL,20,N)
  442   CALL VMULFF(CSYS,ZKL,NS,N,N,10,20,CZKL,10,IER)
        CALL MATVEC(CZKL,AF,10,20,NS,N,CZKLAF)
        TOL=0.D0
        DO 443 I=1,NS
          DO 443 J=1,N
            CZKEPS=CZKL(I,J)*EPS
            TOL=DABS(DMAX1(TOL,CZKEPS))
  443     CONTINUE
          NZERO=0
          DO 444 I=1,NS
            IF(DABS(CZKLAF(I)).LT.TOL) NZERO=NZERO+1
  444     CONTINUE
          IF(NZERO.NE.NS) GO TO 446
          LEXP(K)=LEXP(K)+1
          CALL VMULFF(ZK,ZKL,N,N,N,20,20,ZKL2,20,IER)
          DO 445 I=1,N
            DO 445 J=1,N
              ZKL(I,J)=ZKL2(I,J)
  445     CONTINUE
          GO TO 442
  446     WRITE(IO,447) LEXP(K),(CZKLAF(I),I=1,NS)
  447     FORMAT('L=',I2,5X,'CZKLAF=',3E16.6)
   45 CONTINUE
      RETURN
      END
C
C*  DGAIN DETERMINES THE DETECTOR GAIN FOR THE FILTER
C
      SUBROUTINE DGAIN(C,A,EV,GEN,CF,CPF,NU,NUMOD,N,NF,NS,IAP,IDPR,DPR,
     &                 ZKF,ZMF,AP,D)
      INTEGER NU(10),IQ(10),NUMOD(10)
      DOUBLE PRECISION
     &          C(10,20),A(20,20),GEN(20,10),CF(10,10),CPF(10,20),
     &          G(20),D(20,10),P(20),Q(20),AJ(20,20),AJG(20),
     &          QD(20,10),CFTCF(10,10),CFTCFI(10,10),WK(460),
     &          CFTTI(10,10),DP(20,10),ZMF(200,20),ZMFNUL(20,20),
     &          WL(20),WF(20,10),AP(20,20),DPC(20,20),PSI(20),
     &          APJ(20,20),WFI(20),APJWF(20),PSIW(20,10),ZKF(20,20),
     &          ZKFQ(20,20),ZKFWF(20),W(20,10),CPFW(10,10),CPFWT(10,10),
     &          CPFWTI(10,10),CFWTTI(10,10),DPR(20,10),CF2TTI(10,10),
     &          DPRSUB(20,10),DH(20,10),OMSUB(10,10),AJ2(20,20),
     &          APJ2(20,20),ZKFQ2(20,20),ZMFSUB(20,20),EV,EPS,TOL
      DATA IO/8/
      DO 35 K=1,NF
        DO 5 I=1,N
          G(I)=GEN(I,K)
    5   CONTINUE
        CALL EVAS(20,NUMOD(K),EV,P)
```

```
          DO 10 I=1,N
            Q(I)=P(1)*G(I)
   10     CONTINUE
          NUMOD1=NUMOD(K)-1
          IF(NUMOD1.EQ.0) GO TO 21
          DO 20 J=1,NUMOD1
            CALL MATPOW(A,20,N,J,AJ2,AJ)
            CALL MATVEC(AJ,G,20,20,N,N,AJG)
            DO 15 I=1,N
              Q(I)=Q(I)+P(J+1)*AJG(I)
   15       CONTINUE
   20     CONTINUE
   21     CALL MATPOW(A,20,N,NUMOD(K),AJ2,AJ)
          CALL MATVEC(AJ,G,20,20,N,N,AJG)
          DO 25 I=1,N
            Q(I)=Q(I)+AJG(I)
   25     CONTINUE
          DO 30 I=1,N
            QD(I,K)=Q(I)
   30     CONTINUE
   35 CONTINUE
      CALL VMULFM(CF,CF,NS,NF,NF,10,10,CFTCF,10,IER)
      CALL LINV1F(CFTCF,NF,10,CFTCFI,0,WK,IER)
      CALL VMULFP(CFTCFI,CF,NF,NF,NS,10,10,CFTTI,10,IER)
      CALL VMULFF(QD,CFTTI,N,NF,NS,20,10,DP,20,IER)
      CALL VMULFF(DP,C,N,NS,N,20,10,DPC,20,IER)
      CALL MATSUB(A,DPC,20,20,N,N,AP)
          WRITE(IO,391) ((DP(I,J),J=1,3),I=1,4),
     &                  ((AP(I,J),J=1,4),I=1,4)
391        FORMAT('DP='/4(3E14.4/)//'AP='/4(4E14.4/)/)
      NUSUM=0
      DO 40 K=1,NF
        NUSUM=NUSUM+NU(K)
   40 CONTINUE
      IF(IDPR.EQ.1) GO TO 90
      IF((N-NUSUM.NE.0).AND.(IAP.NE.1)) GO TO 50
      DO 45 I=1,N
        DO 45 J=1,NS
          D(I,J)=DP(I,J)
   45 CONTINUE
      RETURN
   50 CALL MATID(ZMFNUL,20,N)
      CALL ORTRED(ZMF,200,20,NS*N,N,ZMFNUL,NS,1,ZMFNUL,ZMFSUB,
     &            IRKZMF,ICODE3,ZKNU1,WL,IQ,WF)
          WRITE(IO,411) (IQ(I),I=1,4)
 411      FORMAT('IQ='/4I4)
      ICP=0
      DO 85 I=1,NS
        IF(IQ(I).EQ.0) GO TO 85
        ICP=ICP+1
        CALL EVAS(20,IQ(I),EV,P)
        DO 55 L=1,N
          WFI(L)=WF(L,I)
   55   CONTINUE
        DO 60 J=1,N
```

```
            PSI(J)=P(1)*WFI(J)
   60     CONTINUE
          IQIM1=IQ(I)-1
          IF(IQIM1.EQ.0) GO TO 71
          DO 70 J=1,IQIM1
            CALL MATPOW(AP,20,N,J,APJ2,APJ)
            CALL MATVEC(APJ,WFI,20,20,N,N,APJWF)
            DO 65 K=1,N
              PSI(K)=PSI(K)+P(J+1)*APJWF(K)
   65       CONTINUE
   70     CONTINUE
   71     CALL MATPOW(AP,20,N,IQ(I),APJ2,APJ)
          CALL MATVEC(APJ,WFI,20,20,N,N,APJWF)
          DO 75 K=1,N
            PSI(K)=PSI(K)+APJWF(K)
   75     CONTINUE
          CALL MATPOW(ZKF,20,N,IQ(I)-1,ZKFQ2,ZKFQ)
          CALL MATVEC(ZKFQ,WFI,20,20,N,N,ZKFWF)
          DO 80 L=1,N
            PSIW(L,ICP)=PSI(L)
            W(L,ICP)=ZKFWF(L)
   80     CONTINUE
   85   CONTINUE
        CALL VMULFF(CPF,W,NS,N,ICP,10,20,CPFW,10,IER)
        CALL VMULFM(CPFW,CPFW,NS,ICP,ICP,10,10,CPFWT,10,IER)
        CALL LINV1F(CPFWT,ICP,10,CPFWTI,0,WK,IER)
        CALL VMULFP(CPFWTI,CPFW,ICP,ICP,NS,10,10,CFWTTI,10,IER)
        CALL VMULFF(PSIW,CFWTTI,N,ICP,NS,20,10,DPR,20,IER)
   90       WRITE(IO,457) ((DPR(I,J),J=1,3),I=1,4)
457         FORMAT('DPR='/4(3E14.4/)/)
        CALL VMULFF(CF,CFTTI,NS,NF,NS,10,10,CF2TTI,10,IER)
        CALL VMULFF(DPR,CF2TTI,N,NS,NS,20,10,DPRSUB,20,IER)
        CALL MATSUB(DPR,DPRSUB,20,10,N,NS,DH)
        CALL MATADD(DP,DH,20,10,N,NS,D)
        RETURN
        END
C
C*   SENSOR IS THE MAIN PROGRAM FOR SENSOR EVENT FILTER DESIGN
C
        SUBROUTINE SENSOR(C,A,EV,G,NU,CF,CPF,N,NF,NS,
       &                  ZKF,ZMF,FALL,IFAL,IDET,ICAT,D)
        INTEGER NU(10),IQ(10),NUMOD(10),ICAT(10)
        DOUBLE PRECISION
       &           A(20,20),C(10,20),G(20,10),CF(10,10),CPF(10,20),
       &           ZKF(20,20),ZMF(200,20),FALL(20,10),D(20,10),CCT(10,10),
       &          CCTI(10,10),AP(20,20),WOSUB(10,20),WOSUB2(10,20),
       &           WO(200,20),F(20),AF(20),WOAF(200),APAF(20),WK(40),
       &           APAF2(20),CAPAF(10),CAAFN1(10,20),CAFNUL(20,20),
       &           CAFSUB(20,20),WL(20),WF(20,10),AFALL(20,10),
       &           DPR(20,10),CPF2(10,20),AP2(20,20),AFALL2(20,10),
       &           EPS,TOL,WOEPS
        DATA IO,EPS/8,1.D-4/
        IDIR=0
        IF(IFAL.EQ.1) GO TO 5
        CALL VMULFP(C,C,NS,N,NS,10,10,CCT,10,IER)
```

```
      CALL LINV1F(CCT,NS,10,CCTI,0,WK,IER)
      CALL VMULFM(C,CCTI,NS,N,NS,10,10,FALL,20,IER)
          WRITE(IO,121) ((FALL(I,J),J=1,4),I=1,4)
121       FORMAT('FALL='/4(4E14.4/)//)
    5 CALL SEPDET(A,C,FALL,N,NF,NS,ISEP,IDET,IDIR,C,CF,G,NU,NUMOD,
     &               ZKF,CPF,ZMF)
      IF(IDET.EQ.1) RETURN
      CALL DGAIN(C,A,EV,G,CF,CPF,NU,NUMOD,N,NF,NS,1,0,DPR,
     &               ZKF,ZMF,AP,D)
      CALL MATGEN(CPF,AP,10,20,200,NS,N,N,WOSUB,WOSUB2,WO)
          WRITE(IO,132) ((WO(I,J),J=1,4),I=1,12)
132       FORMAT('WO='/12(4E14.4/)//)
      NSN=NS*N
      TOL=0.
      DO 10 I=1,NS
        DO 10 J=1,N
          WOEPS=WO(I,J)*EPS
          TOL=DABS(DMAX1(TOL,WOEPS))
   10 CONTINUE
          WRITE(IO,143) TOL
143       FORMAT('TOL=',E14.4/)
      DO 60 K=1,NF
        DO 15 I=1,N
          F(I)=FALL(I,K)
   15   CONTINUE
        CALL MATVEC(A,F,20,20,N,N,AF)
        CALL MATVEC(WO,AF,200,20,NS*N,N,WOAF)
          WRITE(IO,156) (WOAF(I),I=1,8)
156       FORMAT('WOAF='/2(4E14.4/)//)
        NO=0
        DO 20 I=1,NSN
          IF(DABS(WOAF(I)).LT.TOL) NO=NO+1
   20   CONTINUE
        IF(NO.EQ.NSN) GO TO 25
        ICAT(K)=1
        GO TO 60
   25   ICNT=0
        DO 30 I=1,N
          APAF(I)=AF(I)
   30   CONTINUE
   35   ICNT=ICNT+1
        CALL MATVEC(C,APAF,10,20,NS,N,CAPAF)
        DO 40 I=1,NS
          CAAFN1(I,ICNT)=CAPAF(I)
   40   CONTINUE
        IF(ICNT.EQ.N) GO TO 50
        CALL MATVEC(AP,APAF,20,20,N,N,APAF2)
        DO 45 I=1,N
          APAF(I)=APAF2(I)
   45   CONTINUE
        GO TO 35
   50   CALL MATID(CAFNUL,20,N)
        CALL ORTRED(CAAFN1,NS,N,10,20,CAFNUL,NS,0,CAFNUL,
     &               CAFSUB,IRKCAF,ICODE3,ZKNU1,WL,IQ,WF)
        IF(IRKCAF.NE.1) GO TO 55
```

```
         ICAT(K)=2
         GO TO 60
   55    ICAT(K)=3
   60 CONTINUE
            WRITE(IO,166) (ICAT(K),K=1,4)
 166        FORMAT('ICAT=',4I3/)
      DO 65 K=1,NF
         IF(ICAT(K).EQ.3) RETURN
   65 CONTINUE
      CALL VMULFF(A,FALL,N,N,NF,20,20,AFALL,20,IER)
            WRITE(IO,177) ((AFALL(I,J),J=1,4),I=1,6)
 177        FORMAT('AFALL='/6(4E14.4/)//)
            DO 68 I=1,NS
              DO 68 J=1,N
                CPF2(I,J)=CPF(I,J)
   68       CONTINUE
            DO 69 I=1,N
              DO 69 J=1,N
                AP2(I,J)=AP(I,J)
   69       CONTINUE
      IDIR=1
C*  IF AFALL2 IS INPUTTED TO PROGRAM,
C*  DELETE THESE NEXT 5 STATEMENTS
      NF2=NF
      DO 70 I=1,N
        DO 70 J=1,NF2
          AFALL2(I,J)=AFALL(I,J)
   70 CONTINUE
      CALL SEPDET(AP2,CPF2,AFALL2,N,NF2,NS,ISEP,IDET,IDIR,C,CF,G,
     &                NU,NUMOD,ZKF,CPF,ZMF)
      IF(IDET.EQ.1) RETURN
      CALL DGAIN(CPF2,AP2,EV,G,CF,CPF,NU,NUMOD,N,NF2,NS,0,0,DPR,
     &              ZKF,ZMF,AP,DPR)
            WRITE(IO,127) ((DPR(I,J),J=1,4),I=1,6)
 127        FORMAT('DPR='/6(4E14.4/)//)
      IDIR=0
      CALL SEPDET(A,C,FALL,N,NF,NS,ISEP,IDET,IDIR,C,CF,G,NU,NUMOD,
     &                ZKF,CPF,ZMF)
      CALL DGAIN(C,A,EV,G,CF,CPF,NU,NUMOD,N,NF,NS,0,1,DPR,
     &              ZKF,ZMF,AP,D)
      RETURN
      END
C
C*  ORTRED PERFORMS ORTHOGONAL REDUCTION ON MATRIX V
C
      SUBROUTINE ORTRED(V,M1V,N1V,MV,NV,OMI,NS,ICODE4,OM,OMSUB,
     &                   IRANK,ICODE3,ZKNU1,WL,IO,WF)
      INTEGER IQ(10),ICODE1(10)
      DOUBLE PRECISION
     &           V(M1V,N1V),OMI(N1V,N1V),OM(N1V,N1V),OMSUB(N1V,N1V),
     &           VK(20),W(20),WL(20),WF(20,10),WTVK,ZKNU1,EPS,TOL,VKEPS
      DATA IO,EPS/8,1.D-4/
      ICNT=0
      IRANK=0
      ICODE2=0
      ICODE3=0                                                          FDF05510
          WRITE(IO,205)                                                 FDF05520
205       FORMAT('ENTERING ORTRED')                                     FDF05530
      DO 5 I=1,NS                                                       FDF05540
        IQ(I)=0                                                         FDF05550
        ICODE1(I)=0                                                     FDF05560
    5 CONTINUE                                                          FDF05570
      DO 10 I=1,MV                                                      FDF05580
        DO 10 J=1,NV                                                    FDF05590
          OM(I,J)=OMI(I,J)                                              FDF05600
   10 CONTINUE                                                          FDF05610
      DO 95 K=1,MV                                                      FDF05620
          TOL=0.                                                        FDF05630
        DO 20 J=1,NV                                                    FDF05640
          VK(J)=V(K,J)                                                  FDF05650
          VKEPS=VK(J)*EPS                                               FDF05660
          TOL=DABS(DMAX1(TOL,VKEPS))                                    FDF05670
   20   CONTINUE                                                        FDF05680
        IF(TOL.LT.EPS) TOL=EPS                                          FDF05690
        ICODE2=ICODE2+1                                                 FDF05700
        IARG=ICODE2-((ICODE2-1)/NS)*NS                                  FDF05710
        IF(ICODE1(IARG).NE.0) GO TO 95                                  FDF05720
        CALL MATVEC(OM,VK,N1V,N1V,NV,NV,W)                              FDF05730
        NZERO=0                                                         FDF05740
        DO 25 I=1,NV                                                    FDF05750
          IF(DABS(W(I)).LT.TOL) NZERO=NZERO+1                           FDF05760
   25   CONTINUE                                                        FDF05770
        WTVK=0.                                                         FDF05780
        DO 29 I=1,NV                                                    FDF05790
          WTVK=WTVK+W(I)*VK(I)                                          FDF05800
   29   CONTINUE                                                        FDF05810
        IF(NZERO.NE.NV) GO TO 28                                        FDF05820
        ICODE1(IARG)=1                                                  FDF05830
        GO TO 95                                                        FDF05840
   28   IF(DABS(WTVK).GT.TOL) GO TO 30                                  FDF05850
        ICODE1(IARG)=1                                                  FDF05860
        GO TO 95                                                        FDF05870
   30   IF(ICODE3.NE.0) GO TO 35                                        FDF05880
        ICODE3=K                                                        FDF05890
   35   DO 40 I=1,NV                                                    FDF05900
          WL(I)=W(I)                                                    FDF05910
   40   CONTINUE                                                        FDF05920
        IF(ICODE4.NE.1) GO TO 80                                        FDF05930
        IQ(IARG)=IQ(IARG)+1                                             FDF05940
        DO 60 I=1,NV                                                    FDF05950
          WF(I,IARG)=W(I)                                               FDF05960
   60   CONTINUE                                                        FDF05970
   80   IRANK=IRANK+1                                                   FDF05980
        WTVK=0.                                                         FDF05990
        DO 85 I=1,NV                                                    FDF06000
          WTVK=WTVK+W(I)*VK(I)                                          FDF06010
   85   CONTINUE                                                        FDF06020
        ZKNU1=WTVK                                                      FDF06030
        DO 90 I=1,NV                                                    FDF06040
          DO 90 J=1,NV                                                  FDF06050
          OMSUB(I,J)=W(I)*W(J)/WTVK
   90   CONTINUE
        CALL MATSUB(OM,OMSUB,N1V,N1V,NV,NV,OM)
   95 CONTINUE
      RETURN
      END
C
C*  MATGEN GENERATES LARGER MATRIX C FROM A AND B
C
      SUBROUTINE MATGEN(A,B,M1A,N1A,M1C,MA,NA,N,CSUB,CSUB2,C)
      DOUBLE PRECISION A(M1A,N1A),B(N1A,N1A),C(M1C,N1A),
     &          CSUB(M1A,N1A),CSUB2(M1A,N1A),EPS
      DATA EPS/1.D-5/
      DO 5 I=1,MA
        DO 5 J=1,NA
          IF(DABS(A(I,J)).LT.EPS) A(I,J)=0.D0
          C(I,J)=A(I,J)
          CSUB(I,J)=C(I,J)
    5 CONTINUE
      IF(N.EQ.1) RETURN
      NM1=N-1
      DO 15 K=1,NM1
        CALL VMULFF(CSUB,B,MA,NA,NA,M1A,N1A,CSUB2,M1A,IER)
        DO 10 I=1,MA
          DO 10 J=1,NA
            CSUB(I,J)=CSUB2(I,J)
            C(MA*K+I,J)=CSUB(I,J)
   10   CONTINUE
   15 CONTINUE
      RETURN
      END
C
C*  EVAS DETERMINES COEFFICIENTS P IN CHARACTERISTIC EQUATION
C
      SUBROUTINE EVAS(N1,N,EV,P)
      REAL PDUM(10)
      DOUBLE PRECISION P(N1),EV
      EVSNGL=SNGL(EV)
      DO 5 K=1,N
        I=N-K+1
        BETA=(FLOAT(IFACT(N))/(FLOAT(IFACT(N-I))*FLOAT(IFACT(I))))
     &        *(EVSNGL**I)
        PDUM(K)=((-1.0)**I)*BETA
        P(K)=DBLE(PDUM(K))
    5 CONTINUE
      RETURN
      END
C
C*  MULTIPLIES MATRIX BY VECTOR
C
      SUBROUTINE MATVEC(ARRAY,V,M1,N1,M,N,RET)
      DOUBLE PRECISION ARRAY(M1,N1),V(N1),RET(M1)
      DO 10 I=1,M
        RET(I)=0.D0
        DO 10 J=1,N
```

```
          RET(I)=RET(I)+ARRAY(I,J)*V(J)
   10 CONTINUE
      RETURN
      END
C
C*  ADDS MATRIX B TO MATRIX A
C
      SUBROUTINE MATADD(A,B,M1,N1,M,N,RET)
      DOUBLE PRECISION A(M1,N1),B(M1,N1),RET(M1,N1)
      DO 10 I=1,M
        DO 10 J=1,N
          RET(I,J)=A(I,J)+B(I,J)
   10 CONTINUE
      RETURN
      END
C
C*  SUBTRACTS MATRIX B FROM A
C
      SUBROUTINE MATSUB(A,B,M1,N1,M,N,RET)
      DOUBLE PRECISION A(M1,N1),B(M1,N1),RET(M1,N1)
      DO 10 I=1,M
        DO 10 J=1,N
          RET(I,J)=A(I,J)-B(I,J)
   10 CONTINUE
      RETURN
      END
C
C*  GENERATES THE IDENTITY MATRIX
C
      SUBROUTINE MATID(A,N1,N)
      DOUBLE PRECISION A(N1,N1)
      DO 20 I=1,N
        DO 10 J=1,N
          A(I,J)=0.D0
   10   CONTINUE
        A(I,I)=1.0D0
   20 CONTINUE
      RETURN
      END
C
C*  RAISES MATRIX A TO POWER K
C
      SUBROUTINE MATPOW(A,N1,N,K,AK2,AK)
      DOUBLE PRECISION A(N1,N1),AK(N1,N1),AK2(N1,N1)
      CALL MATID(AK,N1,N)
      IF(K.EQ.0) RETURN
      DO 10 IDUM=1,K
        CALL VMULFF(A,AK,N,N,N,N1,N1,AK2,N1,IER)
        DO 5 I=1,N
          DO 5 J=1,N
            AK(I,J)=AK2(I,J)
    5   CONTINUE
   10 CONTINUE
      RETURN
      END
```

```
C                                                                       FDF07160
C*   FINDS FACTORIAL OF N                                               FDF07170
C                                                                       FDF07180
      INTEGER FUNCTION IFACT(N)                                         FDF07190
      IFACT=1                                                           FDF07200
      IF(N.LE.1) RETURN                                                 FDF07210
      DO 10 J=2,N                                                       FDF07220
        IFACT=IFACT*J                                                   FDF07230
   10 CONTINUE                                                          FDF07240
      RETURN                                                            FDF07250
      END                                                               FDF07260
```

```
                   +++++
                +++++++++++
              +++++++++++++++
             +++++++++++++++++
 ---------------------------------------------
 I---MASSACHUSETTS INSTITUTE OF TECHNOLOGY---I
 ---------------------------------------------
  III       III       III       III       III
  III       III       III       III       III
  III       III       III       III       III
  III       III       III       III       III
  III       III       III       III       III
  III       III       III       III       III
 =============================================
 =============================================
 ========================================= IPC
```

```
    10     3     3     0     1
          -15.0E0
0.         1.0        0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         1.0        0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         1.0        0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         -130.38    0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         1.0
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         -983.45    0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         1.0        0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
-3752.6    0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         1.0        0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         -10181.    0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         1.0        0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         -22556.    0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         0.         1.0
0.         0.         0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.         -43683.    0.
1.00       0.         -1.59      0.         -1.60      0.         1.321      0.
1.04       0.         -0.753     0.         -0.465     0.         -0.181     0.
1.00       0.         -1.01      0.         -0.123     0.         -0.876     0.
-1.30      0.         1.09       0.         0.356      0.         -0.553     0.
1.00       0.         0.00       0.         1.21       0.         0.00       0.
1.41       0.         0.00       0.         1.40       0.         0.00       0.
0.
0.
0.
0.
0.
0.
0.
0.
0.
0.
0.         0.         0.
1.992      1.992      1.992
0.         0.         0.
-3.167     -2.012     0.000
0.         0.         0.
-3.187     -0.2450    2.410
0.         0.         0.
2.631      -1.745     0.000
0.         0.         0.
2.072      -2.590     2.809
0.         0.         0.
-1.500     2.171      0.000
0.         0.         0.
-0.9263    0.7092     2.789
0.         0.         0.
-0.3606    -1.102     0.000
```

APPENDIX B

```
C***********************************************************************
C
C                PROGRAM FDSIM SIMULATES CONTINUOUS
C                    SYSTEM AND FILTER DYNAMICS
C             FOR FAILURE DETECTION AND IDENTIFICATION
C
C***********************************************************************
C
C  INPUT:  NSYS - NUMBER OF SYSTEM STATES
C          NFIL - NUMBER OF FILTER STATES
C          NS - NUMBER OF SENSORS
C          NA - NUMBER OF ACTUATORS
C          NF - NUMBER OF FAILURE EVENTS
C          IAS - (=1) FOR ACTUATOR FAILURES
C                (=2) FOR SENSOR FAILURES
C          OMU - ACTUATOR INPUT FREQUENCY
C          ISR - SAMPLING RATE (HZ)
C          NTSEC - INTEGRATION FREQUENCY (HZ)
C          IFAIL - # OF FAILED ACTUATOR OR SENSOR
C          TFAIL - TIME OF FAILURE (SEC)
C          TLAST - TIME OF SIMULATION
C          A - SYSTEM MATRIX
C          BS - CONTROL EFFECTIVENESS MATRIX FOR SYSTEM
C          CS - MEASUREMENT MATRIX FOR SYSTEM
C          CF1 - MATRIX OF OUTPUT ERROR DIRECTIONS FOR
C                ACTUATOR FAILURES (COLUMNS ARE VECTORS)
C          D - DETECTOR GAIN
C          X - INITIAL STATES
C
C  OUTPUT:  RE - TRANSFORMED OUTPUT ERROR VECTOR
C
C***********************************************************************
C
C
      DIMENSION A(20,20),BS(20,10),BF(20,10),CS(10,20),U(10),
     &          D(20,10),CF1(10,10),X(40),XDOT(40),E(10),S(10,10),
     &          R(10,10),CFTCF(10,10),CFTCFI(10,10),CFTTI(10,10),
     &          WK(40),WK1(40,9),RE(10),C1A(24),CF(10,20),
     &          XS(20),XF(20),YS(10),YF(10),
     &          TIME(5000),RE1(5000),RE2(5000),RE3(5000)
      EXTERNAL FCN
      DATA IN,IO,IND,TOL,EPS,NW/9,10,1,0.001,1.E-4,40/
      COMMON A,BS,BF,CS,CF,D,E,U,OMU,NSYS,NFIL,NA,NS
      CALL PLOTS(IDUM,IDUM,18)
      READ(IN,5) NSYS,NFIL,NS,NA,NF,IAS,OMU,ISR,NTSEC,IFAIL,TFAIL,TLAST,
     &           ((A(I,J),J=1,16),I=1,16),
     &           ((BS(I,J),J=1,8),I=1,16),
     &           ((CS(I,J),J=1,16),I=1,8),
     &           ((CF1(I,J),J=1,3),I=1,8),
     &           ((D(I,J),J=1,3),I=1,16),
     &           (X(I),I=1,32)
    5 FORMAT(6I3/F10.4/I3/2I3,2F10.4/32(8F10.4/),16(8F10.4/),
     &       16(8F10.4/),8(3E16.6/),16(3E16.6/),4(8F10.4/))
      WRITE(IO,7) NSYS,NFIL,NS,NA,NF,NTSEC,IFAIL,TFAIL,TLAST,
     &            ((CF1(I,J),J=1,4),I=1,4),
     &           ((D(I,J),J=1,4),I=1,4)
    7 FORMAT('NSYS=',I2,'NFIL=',I2,5X,'NS=',I2,5X,'NA=',I2,5X,'NF=',I2/
     &         'NTSEC=',I3,5X,'IFAIL=',I2,5X,'TFAIL=',F8.3,5X,'TLAST=',
     &         F8.3/'CF1='/4(4E14.4/)/'D='/4(4E14.4/)/)
      NE=NSYS+NFIL
      DO 10 I=1,NFIL
        DO 10 J=1,NA
          BF(I,J)=BS(I,J)
   10 CONTINUE
      DO 12 I=1,NS
        DO 12 J=1,NFIL
          CF(I,J)=CS(I,J)
   12 CONTINUE
      DO 20 I=1,NS
        DO 15 J=1,NF
          S(I,J)=0.
   15   CONTINUE
        S(I,I)=1.0
   20 CONTINUE
      IF(IAS.EQ.2) GO TO 21
      CALL VMULFM(CF1,CF1,NS,NF,NF,10,10,CFTCF,10,IER)
      CALL LINV1F(CFTCF,NF,10,CFTCFI,0,WK,IER)
      CALL VMULFP(CFTCFI,CF1,NF,NF,NS,10,10,CFTTI,10,IER)
      CALL VMULFF(S,CFTTI,NS,NF,NS,10,10,R,10,IER)
      GO TO 24
   21 COSGAM=CF1(2,1)/SQRT(CF1(2,1)**2+CF1(3,1)**2)
      SINGAM=CF1(3,1)/SQRT(CF1(2,1)**2+CF1(3,1)**2)
      R(1,1)=1.0
      R(1,2)=0.
      R(1,3)=0.
      R(2,1)=0.
      R(2,2)=COSGAM
      R(2,3)=SINGAM
      R(3,1)=0.
      R(3,2)=-SINGAM
      R(3,3)=COSGAM
   24 T=0.
      CALL FCN(NE,T,X,XDOT)
      WRITE(IO,25)
   25 FORMAT(2X,'TIME',16X,'E1',12X,'E2',12X,'E3'/)
      WRITE(IO,30) T,(RE(I),I=1,3)
   30 FORMAT(2X,F6.3,10X,3E14.4)
      ILAST=INT(TLAST)*NTSEC
      IPLOT=ILAST
      NFE=NTSEC/ISR
      INDEXE=0
      IWRT=0
      DO 50 K=1,ILAST
        TEND=FLOAT(K)/FLOAT(NTSEC)
C*   NEXT STATEMENTS FOR DATA-SAMPLING ONLY
C       INDEXE=INDEXE+1
C       IF(INDEXE.NE.NFE) GO TO 38
C       NP1=NSYS+1
C       DO 32 I=1,NSYS
C         XS(I)=X(I)
```

```
                                                                        FDSO1110
C  32   CONTINUE                                                        FDSO1120
C       DO 34 I=NP1,NE                                                  FDSO1130
C         XF(I-NSYS)=X(I)                                               FDSO1140
C  34   CONTINUE                                                        FDSO1150
C       CALL MATVEC(CS,XS,10,20,NS,NSYS,YS)                             FDSO1160
C       CALL MATVEC(CF,XF,10,20,NS,NFIL,YF)                             FDSO1170
C       CALL VECS(YS,YF,10,NS,E)                                        FDSO1180
C       INDEXE=0                                                        FDSO1190
   38   IF(T.LT.TFAIL) GO TO 41                                         FDSO1200
        IF(IFLAG.EQ.1) GO TO 41                                         FDSO1210
        IFLAG=1                                                         FDSO1220
        IF(IAS.EQ.2) GO TO 415                                          FDSO1230
        DO 40 I=1,NSYS                                                  FDSO1240
          BS(I,IFAIL)=0.                                                FDSO1250
   40   CONTINUE                                                        FDSO1260
        GO TO 41                                                        FDSO1270
  415   DO 416 J=1,NSYS                                                 FDSO1280
          CS(IFAIL,J)=0.                                                FDSO1290
  416   CONTINUE                                                        FDSO1300
   41   CALL DVERK(NE,FCN,T,X,TEND,TOL,IND,C1A,NW,WK1,IER)              FDSO1310
        IF(IND.LT.0 .OR. IER.GT.0) GO TO 62                             FDSO1320
        CALL MATVEC(R,E,10,10,NS,NS,RE)                                 FDSO1330
        TIME(K)=T                                                       FDSO1340
        RE1(K)=RE(1)                                                    FDSO1350
        RE2(K)=RE(2)                                                    FDSO1360
        RE3(K)=RE(3)                                                    FDSO1370
        IWRT=IWRT+1                                                     FDSO1380
        IF(IWRT.NE.5) GO TO 50                                          FDSO1390
        WRITE(IO,45) T,(RE(I),I=1,3)                                    FDSO1400
   45   FORMAT(2X,F6.3,10X,3E14.4)                                      FDSO1410
        IWRT=0                                                          FDSO1420
   50 CONTINUE                                                          FDSO1430
      CALL PICTUR(8.0,3.0,'TIME(SEC)',9,'ERROR(FT)',9,                  FDSO1440
     &            TIME,RE1,IPLOT,0.,0,                                  FDSO1450
     &            TIME,RE2,IPLOT,0.,1,                                  FDSO1460
     &            TIME,RE3,IPLOT,0.,2)                                  FDSO1470
      CALL ENDPLT(12.0,0.0,999)                                         FDSO1480
      GO TO 100                                                         FDSO1490
   62 WRITE(IO,75)                                                      FDSO1500
   75 FORMAT('IND<0 OR IER>0')                                          FDSO1510
  100 STOP                                                              FDSO1520
      END                                                               FDSO1530
C                                                                       FDSO1540
C                                                                       FDSO1550
C                                                                       FDSO1560
      SUBROUTINE FCN(NE,T,X,XDOT)                                       FDSO1570
      DIMENSION X(NE),XDOT(NE),U(10),BS(20,10),BF(20,10),CS(10,20),     FDSO1580
     &          XS(20),XSDOT(20),XF(20),XFDOT(20),YS(10),YF(10),        FDSO1590
     &          E(10),D(20,10),A(20,20),AXS(20),AXF(20),BSU(20),        FDSO1600
     &          BFU(20),DE(20),AXFBFU(20),CF(10,20)                     FDSO1610
      COMMON A,BS,BF,CS,CF,D,E,U,OMU,NSYS,NFIL,NA,NS                    FDSO1620
      U(1)=EXP(-0.2*T)*SIN(OMU*T)                                       FDSO1630
      U(2)=EXP(-0.2*T)*SIN(OMU*T)                                       FDSO1640
      U(3)=EXP(-0.2*T)*SIN(OMU*T)                                       FDSO1650
      NP1=NSYS+1
      DO 5 I=1,NSYS
        XS(I)=X(I)
        XSDOT(I)=XDOT(I)
    5 CONTINUE
      DO 10 I=NP1,NE
        XF(I-NSYS)=X(I)
        XFDOT(I-NSYS)=XDOT(I)
   10 CONTINUE
      CALL MATVEC(A,XS,20,20,NSYS,NSYS,AXS)
      CALL MATVEC(BS,U,20,10,NSYS,NA,BSU)
      CALL VECP(AXS,BSU,20,NSYS,XSDOT)
      CALL MATVEC(A,XF,20,20,NFIL,NFIL,AXF)
      CALL MATVEC(CS,XS,10,20,NS,NSYS,YS)
      CALL MATVEC(CF,XF,10,20,NS,NFIL,YF)
      CALL VECS(YS,YF,10,NS,E)
      CALL MATVEC(D,E,20,10,NFIL,NS,DE)
      CALL MATVEC(BF,U,20,10,NFIL,NA,BFU)
      CALL VECP(AXF,BFU,20,NFIL,AXFBFU)
      CALL VECP(AXFBFU,DE,20,NFIL,XFDOT)
      DO 15 I=1,NSYS
        XDOT(I)=XSDOT(I)
   15 CONTINUE
      DO 20 I=NP1,NE
        XDOT(I)=XFDOT(I-NSYS)
   20 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE MATVEC(ARRAY,V,M1,N1,M,N,RET)
      DIMENSION ARRAY(M1,N1),V(N1),RET(M1)
      DO 10 I=1,M
        RET(I)=0.
        DO 10 J=1,N
          RET(I)=RET(I)+ARRAY(I,J)*V(J)
   10 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE VECP(V1,V2,N1,N,VADD)
      DIMENSION V1(N1),V2(N1),VADD(N1)
      DO 5 I=1,N
        VADD(I)=V1(I)+V2(I)
    5 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE VECS(V1,V2,N1,N,VSUB)
      DIMENSION V1(N1),V2(N1),VSUB(N1)
      DO 5 I=1,N
        VSUB(I)=V1(I)-V2(I)
    5 CONTINUE
      RETURN
      END
```

```
 16 10  3  3  3  1
20.0
 64
192  2      1.0       2.01
0.          1.0         0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          1.0         0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          1.0         0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          -130.38     0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          1.0
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          -983.45     0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          1.0         0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
-3752.6     0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          1.0         0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          -10181.     0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          1.0         0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          -22556.     0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          0.          1.0
0.          0.          0.          0.          0.          0.          0.          0.
0.          0.          0.          0.          0.          0.          -43683.     0.
0.          0.          0.
1.992       1.992       1.992
0.          0.          0.
-3.167      -2.012      0.000
0.          0.          0.
-3.187      -0.2450     2.410
0.          0.          0.
2.631       -1.745      0.000
0.          0.          0.
2.072       -2.590      2.809
0.          0.          0.
-1.500      2.171       0.000
0.          0.          0.
-0.9263     0.7092      2.789
0.          0.          0.
-0.3606     -1.102      0.000
1.00        0.          -1.59       0.          -1.60       0.          1.321       0.
1.04        0.          -0.753      0.          -0.465      0.          -0.181      0.
1.00        0.          -1.01       0.          -0.123      0.          -0.876      0.
```

```
-1.30     O.        1.09      O.        0.356     O.        -0.553    O.
1.00      O.        0.00      O.        1.21      O.        0.00      O.
1.41      O.        0.00      O.        1.40      O.        0.00      O.
O.
O.
O.
O.
O.
O.
O.
O.
O.
O.
    0.177572E+02    0.584335E+00    0.105736E+01
    0.584315E+00    0.894988E+01   -0.195613E+01
    0.105725E+01   -0.195635E+01    0.886879E+01
O.
O.
O.
O.
O.
    0.214951E+01    0.981505E+01    0.149971E+02
    0.193940E+02    0.627022E+02    0.635415E+02
   -0.560015E+01   -0.487336E+01    0.688610E+01
   -0.383867E+02   -0.486003E+02   -0.240117E+01
   -0.586796E+01    0.119337E+01    0.746371E+01
   -0.207824E+02    0.113480E+02    0.575569E+02
    0.575753E+01   -0.985363E+01   -0.165995E+02
   -0.884495E+02    0.587169E+02   -0.434314E+03
    0.294047E-01    0.362257E+01    0.562981E+02
   -0.484272E+03    0.117701E+04    0.569748E+03
    0.0             0.0             0.0
    0.0             0.0             0.0
    0.0             0.0             0.0
    0.0             0.0             0.0
    0.0             0.0             0.0
    0.0             0.0             0.0
O.        O.        O.        O.        O.        O.        O.        O.
O.        O.        O.        O.        O.        O.        O.        O.
O.        O.        O.        O.        O.        O.        O.        O.
O.        O.        O.        O.        O.        O.        O.        O.
O.
```

References

1. Beard, R.V., "Failure Accommodation in Linear Systems Through Self-Reorganization," Ph.D. Thesis, Dept. of Aeronautics and Astronautics, M.I.T. Cambridge MA, Feb. 1971.

2. Jones, H.L., "Failure Detection in Linear Systems," Ph.D. Thesis, Dept. of Aeronautics and Astronautics, M.I.T. Cambridge MA, Aug. 1973.

3. Meserole, J.S., "Detection Filters for Fault-Tolerant Control of Turbofan Engines," Ph.D. Thesis, Dept. of Aeronautics and Astronautics, M.I.T., Cambridge MA, June 1981.

4. Cole, S.R., Horner, G., and Montgomery, R., "Finite Element Analysis of Experimental Beam," NASA Langley Research Center, 1980.

5. Major, C.S., "A Demonstration of the Use of Generalized Parity Relations for Detection and Identification of Instrument Failures on a Free-free Beam," M.I.T. Space Systems Lab., Report #28-81, Sept. 1981.

6. VanderVelde, W.E., and Carignan, C.R., "A Dynamic Measure of Controllability and Observability for the Placement of Actuators and Sensors on Large Space Structures," M.I.T. Space Systems Lab., Report #2-82, Jan. 1982.

7. Balas, M.J., "Active Control of Flexible Systems," Proc. of Dynamics and Control of Large Flexible Spacecraft Symposium, Blacksburg VA, June 1977.

8. VanderVelde, W.E., "Application of Failure Detection Filter Theory to Reliable Longitudinal Control of Guideway Vehicles," Report for U.S. Dept. of Transportation Contract No. DOT-TSC-1445, M.I.T. Cambridge MA, March 1979.

9. Gerard, J.P., "Application of Failure Detection Theory to Longitudinal Control of Guideway Vehicles," M.S. Thesis, Dept. of Aeronautics and Astronautics, M.I.T., Cambridge MA, June 1978.

10. Gustafson, D.E., and Speyer, J.L., "Linear Minimum Variance Filters Applied to Carrier Tracking," Trans. on Auto. Control, Vol. AC-21, No. 1, Feb. 1976.

11. VanderVelde, W.E., "Reliability Issues in Active Control of Flexible Space Structures," Workshop on Structural Dynamics and Control of Large Space Structures, NASA Langley Research Center, Oct. 1980.