An Investigation Into the
Stability of Redundant Controller
Voting Schemes

by

D. James Feltner

Project submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment for the degree of
MASTER OF SCIENCE
in
Electrical Engineering

APPROVED:

L. L. Grigsby, Chairman

C. E. Nunnally

H. F. VanLandingham

May, 1984
Blacksburg, Virginia

c7

LD
5655
V851
1990

F467
c.2

An Investigation Into the Stability of Redundant Controller Voting Schemes

by

D. James Feltner

(ABSTRACT)

Redundant controllers are being designed to provide improved system reliability to meet the needs of various applications. In some implementations of these systems unacceptable system transients have occured when a single-controller failure occurs. The problem has been treated using case-by-case correction which is not sufficiently general in application to meet the needs of systems which may be user-programmed.

Redundant control systems generalized as ideal and nonideal using two specific voting schemes are examined to gain insight into theoretical characterization of their performance under single-point failure modes and the limitations of applicability for the characterizations. State-model computer simulation is used to demonstrate the characterizations. A suggested direction for a general solution to the problem based on the chatacterizations is described.

TABLE OF CONTENTS

INTRODUCTION

PRESENTATION OF PROBLEM

A significant number of industrial applications require digital controllers with extremely high running reliability. The required reliability usually exceeds the achievable levels of mean time between failure (MTBF) for single controllers. One fairly common solution to the problem is to use multiple identical controllers in a "two-out-of three" voting scheme in which any two controllers which agree on the control signal required are capable of overriding the third, which is assumed to have failed and to be producing an erroneous control signal.

There have been a wide variety of proposed and implemented schemes for operating three parallel redundant controllers [8], [9]. The schemes fall into two major classifications: hardware implemented fault tolerance (HIFT) and software implemented fault tolerance (SIFT). A modified SIFT scheme has been proposed for a new product in which three digital controllers will use software-implemented internal state voting and mixed hardware/software output voting to implement a fault-tolerant controller with loose synchronism.

It has been observed in previous implementations of fault-tolerant controllers using 2/3 voting that under some disturbance conditions (e.g. single controller failure), system transients occur which occasionally are of sufficient magnitude to cause system parameters to exceed allowable levels. It has been hypothesized that the proposed voting scheme may exacerbate this problem by introducing nonlinear terms into an otherwise stable system, and that these terms may in fact destabilize the control system.

The purpose of this project is to model a simple redundant digital controller and linear plant, and investigate the performance of the system with different voting schemes under disturbance conditions. The project will attempt to analytically characterize the effect of the

voting scheme, predict the effect of the scheme, and confirm the result with simulation. The project will attempt to characterize the performance of an ideal averaging voter and to predict the effect of non ideal realizations of a mid-value select voting scheme.

STATEMENT OF THE PROBLEM

There are probably an infinite number of possible topologies for control networks with voting. This paper will consider in detail only one of these, forward-path voting of the outputs of three controllers to produce the control input to the plant. A second topology, forward-path and feedback-path voting, will be briefly considered.

Consider the general system shown in Figure 1 with three identical digital controllers CTL-K, CTL-L, and CTL-M with a common input vector $\underline{R}$, state vectors $\underline{X}k$, $\underline{X}l$, and $\underline{X}m$, feedback vectors $\underline{U}k$, $\underline{U}l$, and $\underline{U}m$ respectively and outputs $\underline{Y}k$, $\underline{Y}l$, and $\underline{Y}m$. The individual outputs are operated on by the voter to produce $\underline{Y}$, the control output. This control output is applied to the plant. Each controller is defined by the linear model

$$\frac{d}{dt}\underline{X}j = \underline{A}j\underline{X}j + \underline{B}j\underline{U}j + \underline{D}j\underline{R}$$

$$\underline{Y}j = \underline{C}j\underline{X}j \qquad (1)$$

where j = k, l, or m and $\underline{A}$, $\underline{B}$, $\underline{C}$, and $\underline{D}$ are real, constant-coefficient matrices. Since the controllers are identical $\underline{A}k = \underline{A}l = \underline{A}m$ as are the $\underline{B}$'s, $\underline{C}$'s, and $\underline{D}$'s. If the feedback $\underline{U}j$ is for the moment considered to be zero, $\underline{A}$ can be redefined to include the reference in a homogeneous controller model under the assumption that the reference $\underline{R}$ can be modeled by a homogeneous differential equation. This assumption is justified since all references which will be considered are included in that class. Now the controllers are defined by the linear model

$$\frac{d}{dt}\underline{X}j = \underline{A}j\underline{X}j + \underline{B}j\underline{U}j$$

$$\underline{Y}j = \underline{C}j\underline{X}j \qquad (2)$$

where $\underline{A}$ is the augmented coefficient matrix and $\underline{X}$ is the augmented state vector for the homogeneous controller-reference state model. For notational simplicity the three controllers can be combined as follows:

$$\underline{X}c = \begin{matrix}\underline{X}k \\ \underline{X}l \\ \underline{X}m\end{matrix}, \quad \frac{d}{dt}\underline{X}c = \begin{matrix}\frac{d}{dt}\underline{X}k \\ \frac{d}{dt}\underline{X}l \\ \frac{d}{dt}\underline{X}m\end{matrix} \qquad (3)$$

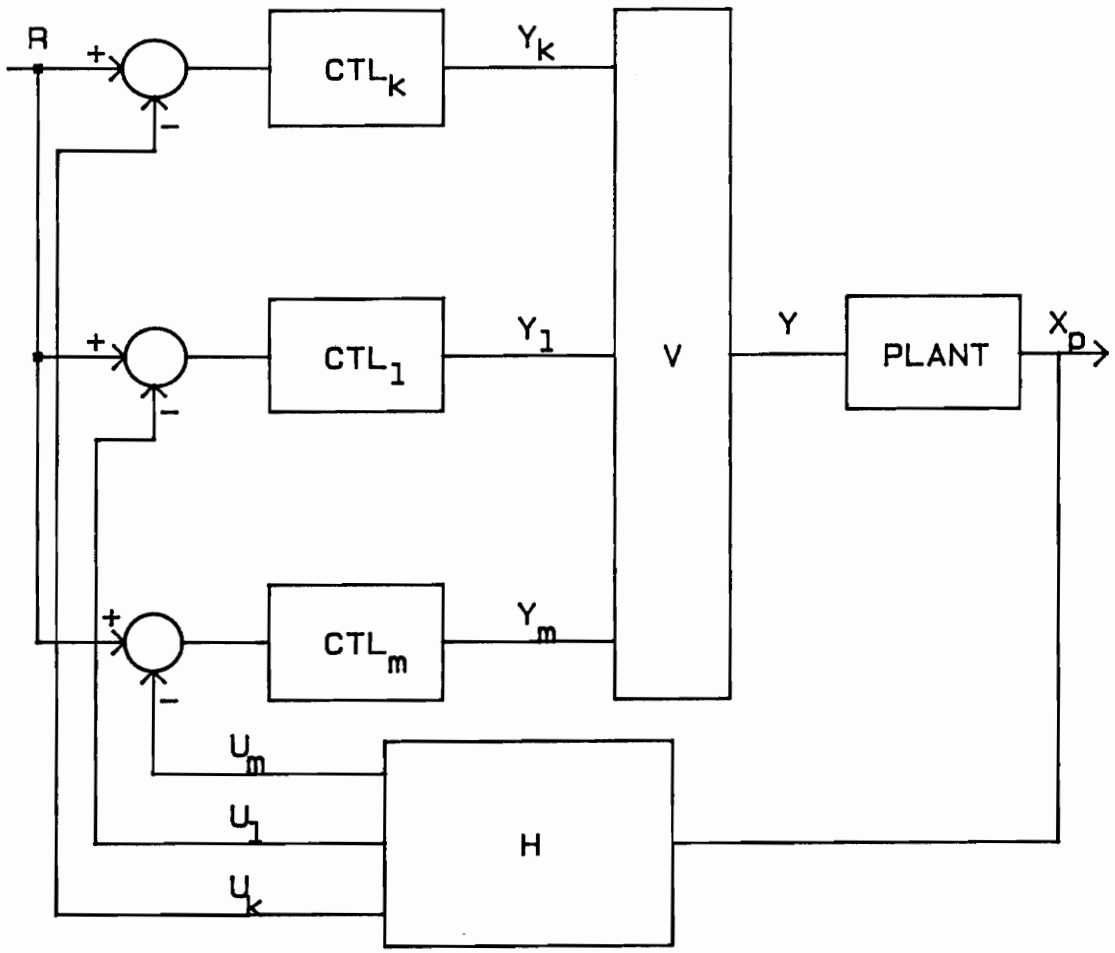

Figure 1

General Redundant Controller Block Diagram

$$\underline{U}c = \begin{matrix} \underline{U}k \\ \underline{U}l \\ \underline{U}m \end{matrix} \qquad (4)$$

$$\underline{A}c = \begin{matrix} \underline{A}k & \underline{0} & \underline{0} \\ \underline{0} & \underline{A}l & \underline{0} \\ \underline{0} & \underline{0} & \underline{A}m \end{matrix} \qquad (5)$$

$$\underline{B}c = \begin{matrix} \underline{B}k & \underline{0} & \underline{0} \\ \underline{0} & \underline{B}l & \underline{0} \\ \underline{0} & \underline{0} & \underline{B}m \end{matrix} \qquad (6)$$

$$\underline{C}c = \begin{matrix} \underline{C}k & \underline{0} & \underline{0} \\ \underline{0} & \underline{C}l & \underline{0} \\ \underline{0} & \underline{0} & \underline{C}m \end{matrix} \qquad (7)$$

$$\underline{Y}c = \begin{matrix} \underline{Y}k \\ \underline{Y}l \\ \underline{Y}m \end{matrix} \qquad (8)$$

and thus the composite controller becomes

$$\begin{aligned} d/dt\ \underline{X}c &= \underline{A}c\underline{X}c + \underline{B}c\underline{U}c \\ \underline{Y}c &= \underline{C}c\underline{X}c \end{aligned} \qquad (9)$$

The plant is defined by the linear model

$$\begin{aligned} d/dt\ \underline{X}p &= \underline{A}p\underline{X}p + \underline{B}\underline{U}p \\ \underline{Y}p &= \underline{C}p\underline{X}p \end{aligned} \qquad (10)$$

where $\underline{A}p$, $\underline{B}p$, and $\underline{C}p$ are real, constant-coefficient matrices.

The input to the plant $\underline{U}p$ is the output of the voter $\underline{Y} = \underline{V}(\underline{Y}k, \underline{Y}l, \underline{Y}m)$. The voter is modeled by the system (not necessarily linear)

$$\frac{d}{dt}\underline{X}v = \underline{A}v\underline{X}v + \underline{B}v\underline{U}v$$

$$\underline{Y}v = \underline{C}v\underline{X}v \qquad (11)$$

For the non-voted-feedback closed loop system, the controller input $\underline{U}c$ is defined by

$$\underline{U}c = \underline{H}\underline{X}p \qquad (12)$$

The complete system model is the set of equations

$$\frac{d}{dt}\underline{X}c = \underline{A}c\underline{X}c + \underline{B}c\underline{U}c \qquad (13a)$$

$$\frac{d}{dt}\underline{X}p = \underline{A}p\underline{X}p + \underline{B}p\underline{U}p \qquad (13b)$$

$$\frac{d}{dt}\underline{X}v = \underline{A}v\underline{X}v + \underline{B}v\underline{U}v \qquad (13c)$$

$$\underline{U}c = \underline{H}\underline{X}p \qquad (13d)$$

$$\underline{U}p = \underline{C}v\underline{X}v \qquad (13e)$$

$$\underline{U}v = \underline{C}c\underline{X}c \qquad (13f)$$

By defining $\underline{X}$ as transpose [$\underline{X}c$ $\underline{X}v$ $\underline{X}p$] the equations (13a) - (13f) for the system may be combined to yield the single homogeneous equation

$$\frac{d}{dt}\underline{X} = \begin{matrix} \underline{A}c & \underline{0} & \underline{B}c\underline{H} \\ \underline{B}v\underline{C}c & \underline{A}v & \underline{0} \\ \underline{0} & \underline{B}p\underline{C}v & \underline{A}p \end{matrix} \underline{X} \qquad (14)$$

For a plant of order n, a voter of order r, and a controller of order s the resulting system is of order 3s+r+n.

TEST CASE SYSTEM MODEL

A model suggested by an actual system [3] will be used to test the results predicted by analysis. This system is the fuel servo control subsystem of the General Electric SPEEDTRONIC (TM) Mark IV Turbine Control System shown in Figure 2. This subsystem, shown in Figure 3 in nonredundant form, provides position control of a two-stage hydraulic servo valve which in turn controls the fuel flow to the combustion system of an industrial gas turbine, in response to a fuel flow reference signal. The turbine drives a synchronous generator which is connected to power grid which is assumed to be large (stiff) with respect to the output of the generator. The actual system is a non linear hybrid digital-analog-hydraulic system. The digital portion of the system computes a fuel flow reference using a complex nonlinear algorithm, and is a redundant (three-controller) system. This algorithm is for the purposes of this analysis considered to be very slow with respect to the fuel control system and the resulting fuel flow reference will be considered a constant in the analysis. The three digital controller outputs are fed to three analog servo regulators, which output a servo coil current signal using the fuel flow reference input, and fuel flow and servo coil current feedback. The first-stage electrohydraulic servo valve provides the voting and first-stage power amplification, while the second hydraulic stage provides power amplification only.

The primary nonlinear elements in the actual system occur in two places. The actual fuel flow reference signal is operated on by a limit function to hold maximum and minimum fuel flow reference within pre-established limits which maintain minimum fuel required to prevent flameout and limit maximum fuel to prevent overfueling during transient conditions which occur primarily during startup and acceleration of the turbine. Since the reference is considered constant for this analysis, this nonlinearity is eliminated. The second major nonlinearity occurs since the fuel flow reference is a multiplicative function of the shaft speed N of the turbine. In this analysis the synchronous generator

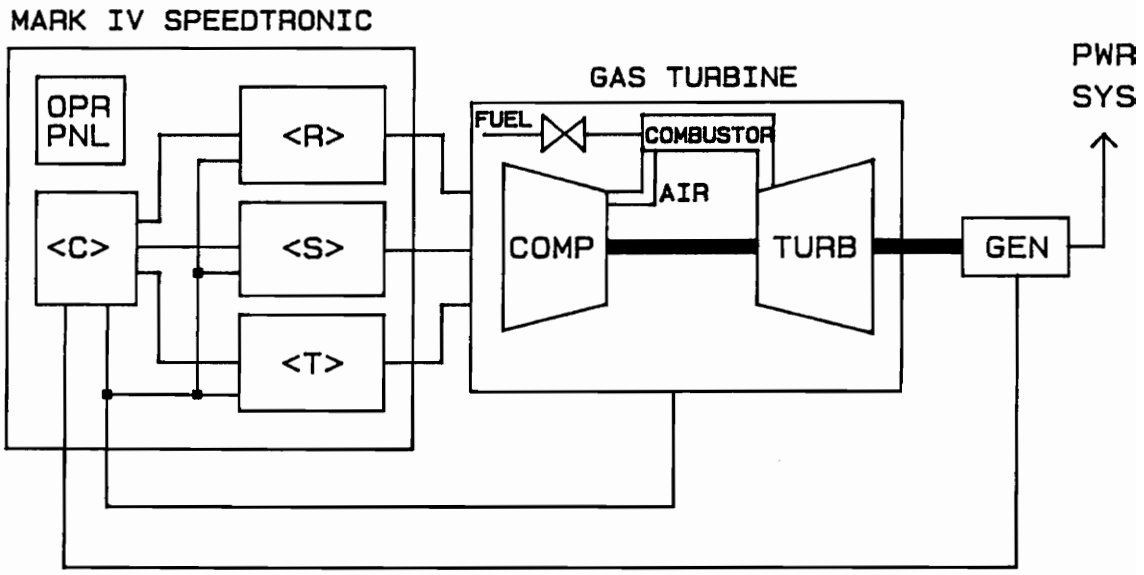

Figure 2
System Block Diagram

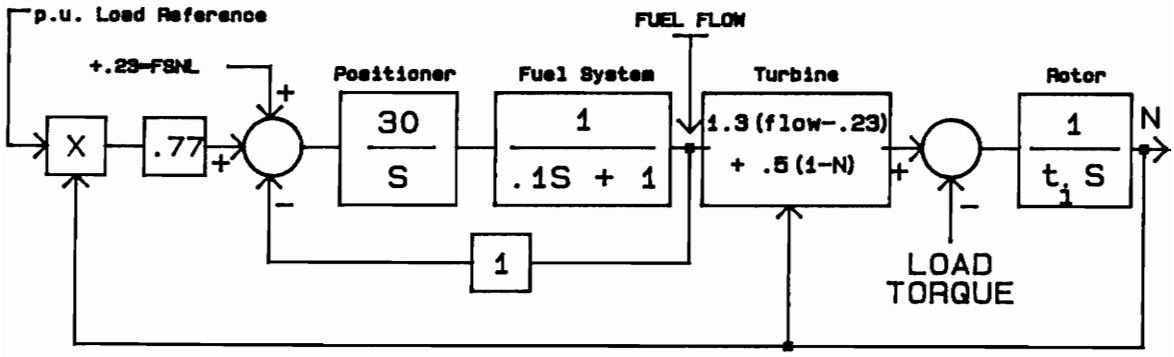

Figure 3
System Model

which is rigidly coupled to the turbine shaft is connected to the power system, which is assumed to be sufficiently stiff to hold turbine speed essentially constant. With constant speed of 1 p.u. the nonlinearity thus is removed, becoming part of a gain term in the model's forward path. The assumption of constant speed also simplifies the function which models the turbine itself, and eliminates effects of the rotor inertia time constant.

Using the linearizations described above and appropriate values for the various parameters [3], the nonredundant system model reduces to the model shown in Figure 4. From this model it is apparent that the output torque of the turbine, and thus the output power of the synchronous generator to the power system, is a linear function of fuel flow. Thus the criteria for performance evaluation becomes the magnitude of the deviation which occurs in the fuel flow in response to disturbances introduced in the system. Since the intent is to investigate the effect of the disturbances and not to design a controller, the systems will be compared based on relative deviation in the regulated fuel flow and not against an absolute criteria, except that steady-state negative power output (fuel flow less than 0.23 p.u.) is considered unacceptable, since reverse power flow in the generator can result in a system trip on an inverse time characteristic, interrupting output to the served power system [3].

In the system modeled by the redundant controller shown in Figure 5 the actual voting is done by summing magnetic flux in a spring-biased three-coil hydraulic servo valve. This valve operates as follows: A supply of high-pressure oil is applied to a movable jet, which aims the oil at two outlet ports, one of which causes the main fuel pump bypass valve to open and the other causes it to close by operating a pilot spool valve. Closing the bypass valve increases fuel flow to the turbine, while opening it reduces flow to the turbine by recirculating the fuel through the main fuel pump. The mechanical (spring) bias on the jet is such that in the absence of current in the coils the jet will be positioned to open the bypass valve. Increasing current in the servo coils points the jet toward the port which closes the bypass valve. The

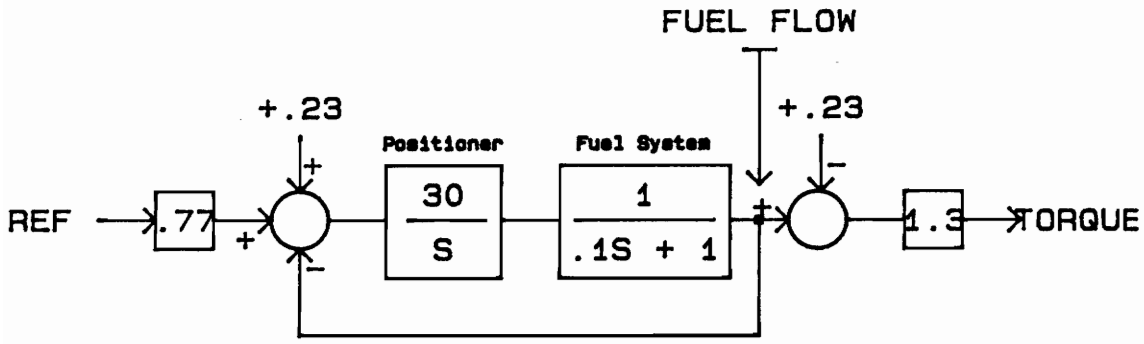

Figure 4
Constant-speed System Model

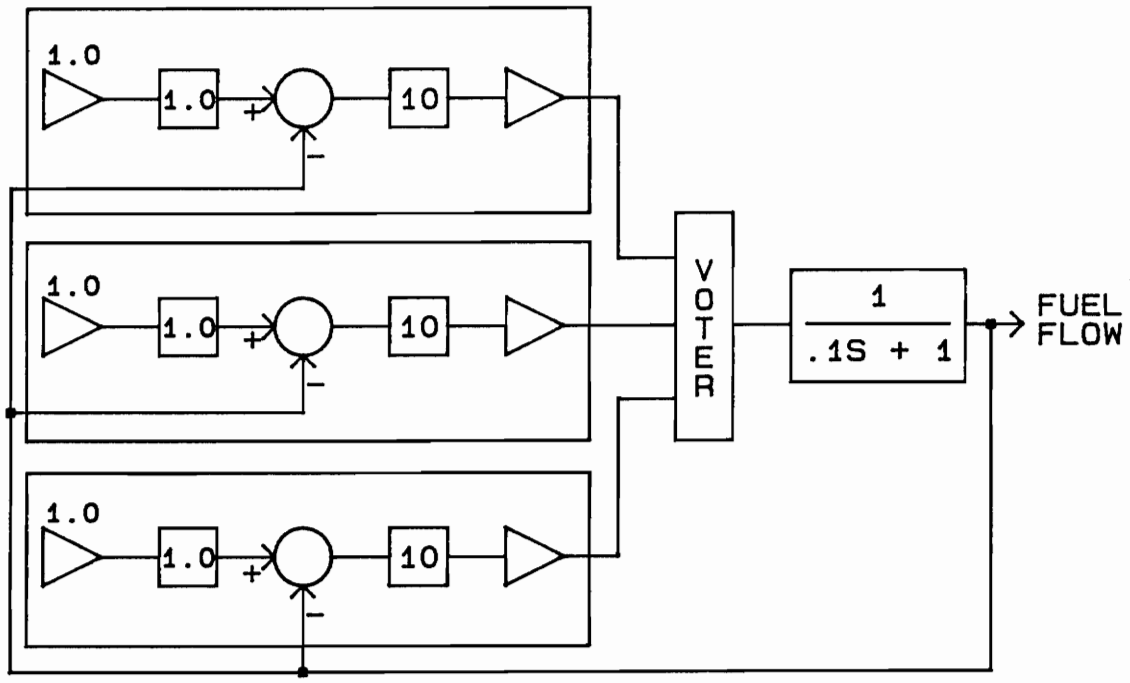

Figure 5
Redundant Controller Model

relative gain of an individual coil is such that the flux contribution of one coil is required to overcome the bias spring and close the valve, thus in the case of a failure of a single coil or driver circuit the remaining two are able to maintain authority over the position of the valve. During steady-state conditions the jet is positioned such that the flow is equally divided between the two ports, the pilot valve spool is centered, and the position of the bypass valve does not change. Since this jet has very low inertia and the time constant of the coils and driving electronics is very small with respect to the mechanical time constants of the fuel system, the complete valve may be modeled as an integrator, with the voter averaging the input currents [3], [5].

SIMULATION OF THE SYSTEM MODEL

The system model has been derived to be amenable to simulation using state-variable techniques, in particular the state variable simulation program STATEVAR (Appendix A), the details of which will be discussed here only to the extent necessary to explain its application to the problem at hand. This program has been used by the author as a general-purpose state-variable-simulation tool for about three years. It is basically an enhanced version of a program called SAMP [6], and operates using a homogeneous state-variable transition matrix simulation of the linear continuous portion of the system. Nonlinearities and functions which are not able to be linearly approximated are simulated by clamping the state variables at appropriate intervals and using a separate mechanism to compute the state variables at t+ from the state variables at time t. This technique, which was originally developed to simulate sampling in a digital control system, is well-suited for handling the effect of a voter in the system. By writing appropriate subroutines and linking them to the main program virtually any voting scheme may be represented. If the simulation is run at sufficiently high speed, with the effect of the voter applied to every state transition calculation, the effect of a continuous voter is effectively simulated using an essentially discrete technique. This is the same technique which has been applied in previous work [6] to nonlinearities. The subroutines for the various voting schemes will be individually referenced as they are encountered in the analysis.

ANALYSIS

IDEAL SYSTEM WITH AVERAGING VOTER

Some immediate observations about the system with ideal components may be made from examination of eq. (14):

1. If the voting function is linear and the plant, voter, and controllers are inherently stable (det [s$\underline{I}$ - $\underline{A}$p], det [s$\underline{I}$ - $\underline{A}$v], det [s$\underline{I}$ - $\underline{A}$c] having poles only in the left half-plane) then the open-loop system ($\underline{H}$ = $\underline{0}$) is also stable in the bounded-input, bounded-output sense.

2. If the voting function is linear then the transient response of the open-loop system is totally determined by the characteristic equations of the plant, voter, and the controller.

Both of these results follow from the fact that with $\underline{H}$ = $\underline{0}$ the coefficient matrix of eq. (14) is lower block diagonal with the subsystem coefficient matrices on the main diagonal and thus the eigenvalues of the open-loop system are the union of the eigenvalues of the plant, voter, and the controller.

From Eq. (14) it can also be concluded that, for the special case of linear voting in the forward path of the system with all controllers operational, the closed-loop system can be analyzed using all of the normal (classical or modern) techniques for system analysis, and further that conventional linear design techniques apply and will yield useful results. The inclusion of the redundant controllers and the voter increase the complexity of the design or analysis problem only in that they increase the order of the system which must be modeled and studied.

Unfortunately very few useful voting functions are linear. A simple averaging voter or an averaging voter with time constants, although linear, usually produces acceptable performance only when all three inputs (controller outputs) are present. This conclusion is

supported by the following intuitive argument: If a single controller fails then the erroneous output is averaged with the correct outputs and the steady-state reference input to the plant changes, with corresponding change in plant operating point. In an open-loop system the resulting steady-state output will also change, by 1/n of the change in the output of the failed controller, where n is the number of controllers. For example if the output of the three controllers is 1 p.u. each before failure and the voting is simple average then before failure the reference to the plant is also 1 p.u. If one controller fails and it's output goes to zero then after the failure the reference applied to the plant is 2/3 p.u. and the open-loop output will be 1/3 less than the output before the failure.

One alternative to this (normally) unacceptablly poor regulation is to increase the number of controllers supplying input to the voter until the resulting performance is acceptable. It is not, however, economically feasible to use ten controllers to provide only 10% regulation.

The regulation of a closed-loop system using a simple averaging voter would be expected to improve. The transient response at first examination should be similar to a simple (nonredundant) system with a disturbance equal to the change in voter output applied to the plant at the point of the voter output. Figure 6 shows the homogeneous state model used for the nonredundant system simulation. The transfer function for this system is

$$\frac{C}{R} = \frac{300}{s^2 + 10s + 300} \quad , \qquad (15)$$

and has a calculated damping ratio of 0.288675 and natural frequency of 17.3205 rad/second corresponding to 38.782 percent overshoot and 0.1845 seconds time-to-peak.

The STATEVAR program is first run with the disturbance model X4 equal to zero to establish the steady-state conditions, and then by setting the initial condition on X4 to -1/3 and using the final conditions from the remaining state variables for the initial conditions, the second run produces the system response to the

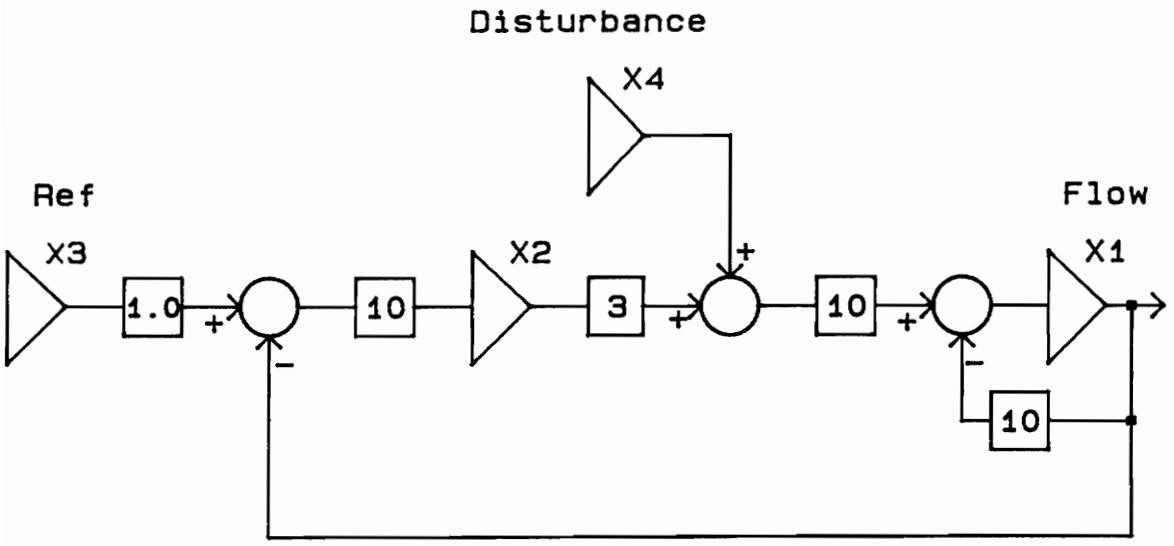

Figure 6
Non-redundant System State Model

disturbance. Figure 7 shows the response of the baseline (nonredundant) system to a -1/3 p.u. step disturbance applied to the summing junction where the redundant voting will be performed, and characterizes the expected system response.

Figure 8 shows the homogeneous state model used for the redundant system simulation. Note that in this model the integration is performed in the controllers and not in the plant, as is the case in the actual system. The linear averaging voter is constructed without additional state variables by reducing the controller output gains (X2, X4, and X6 in Figure 8) to unity and summing the individual outputs. Again the STATEVAR program is run once to establish the steady-state values of the state variables, and using these as initial conditions the disturbance run is constructed by disconnecting X6 from the voter input. It is also necessary to disable the inputs feeding d/dt X6 to prevent the program from terminating on arithmetic overflow. Figure 9 shows the response of the voting system to a -1 p.u. step disturbance in the output of one controller, with that controller rendered non-functional by opening its forward transfer function. The (identical) response of the two remaining controllers is also shown in Figure 9.

Comparison of the plant output for the baseline case and the redundant case shows that the system response is not identical in the two cases. The baseline system has -13.07% overshoot, with time-to-peak of 0.08 seconds and settling to within 2% of final value in 0.34 seconds. The voting system, on the other hand, has -14.9% overshoot with time-to-peak of 0.09 seconds and settling time of 0.43 seconds.

On closer examination one observes that the effective forward gain of the control system is reduced by the lack of control output from the non-functional controller. In the baseline system the control output has an effective gain of 1 when applied to the plant input, while in the redundant case the averaging voter causes the plant input gain to be 2/3 (two control inputs each with a relative strength of 1/3). This example leads to the conclusion that if the gain of the controller output applied to the plant input is K then for the triple-redundant case the transient response of the system is characterized by a nonredundant

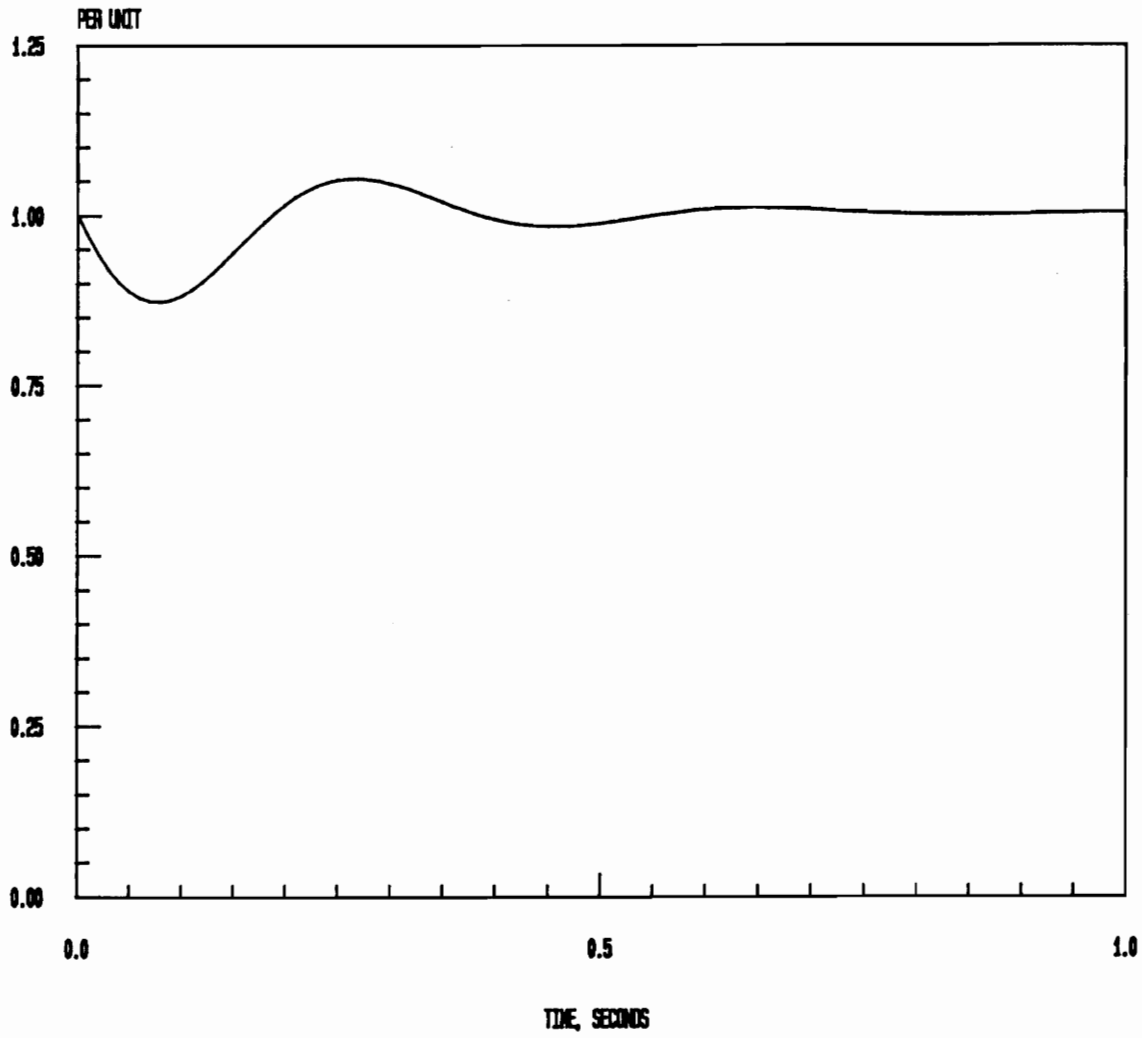

Figure 7
Non-redundant System Response

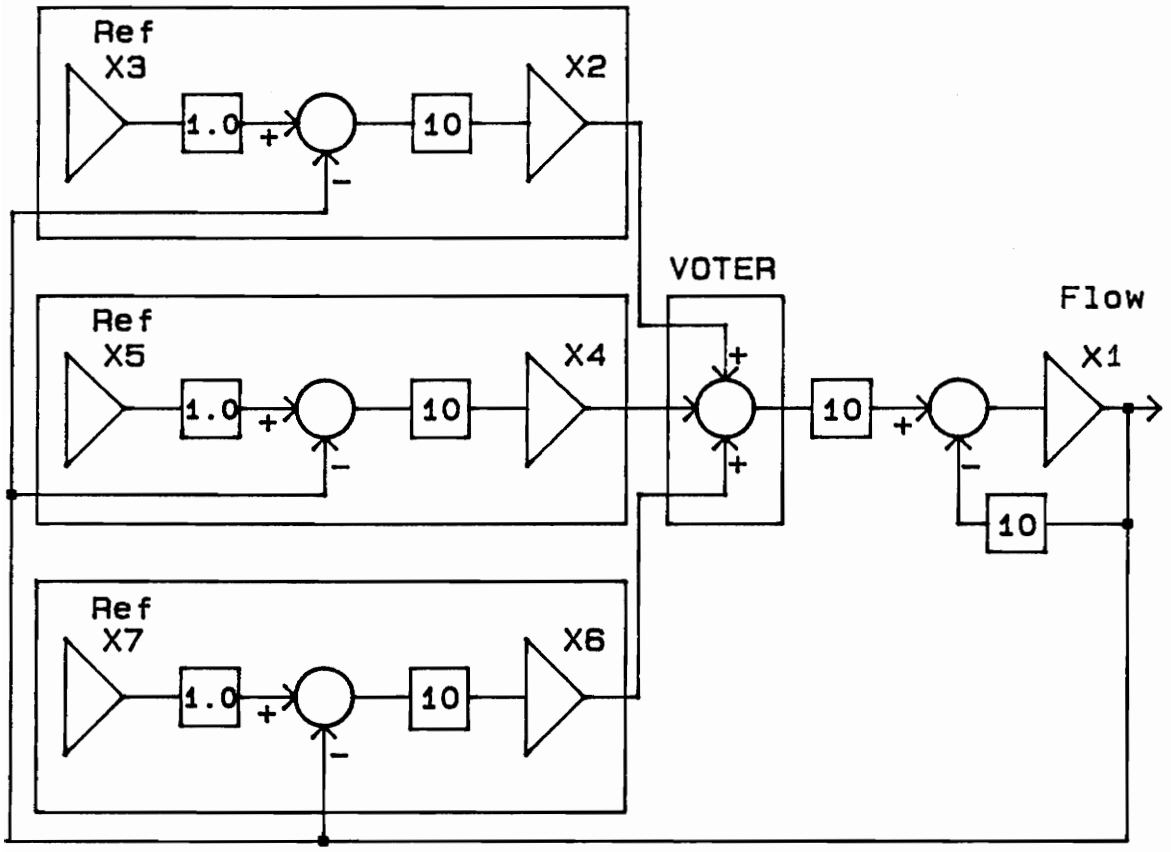

Figure 8
Redundant System State Model

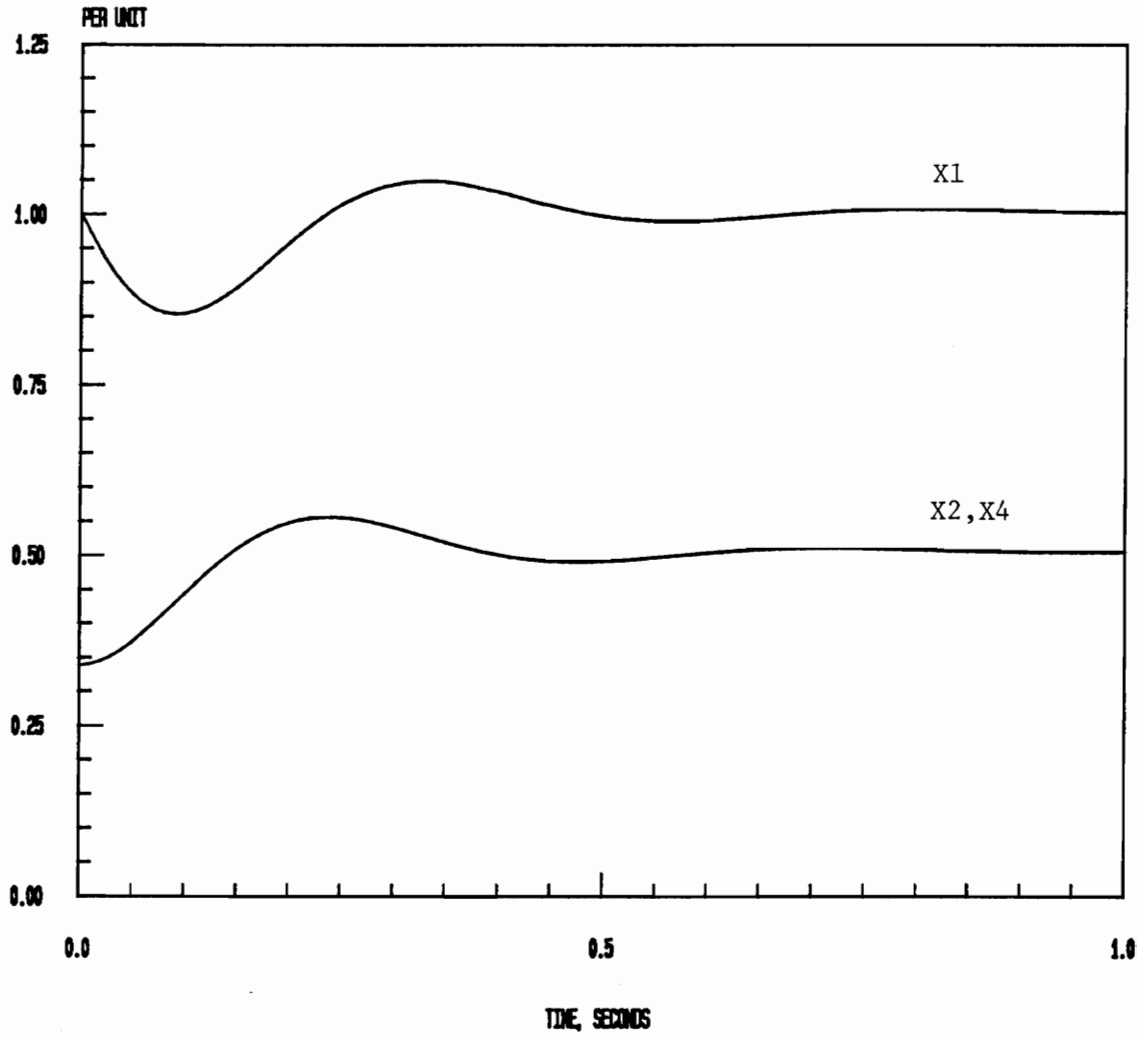

Figure 9
Redundant System Response

system with the corresponding gain of 2K/3. This conclusion is confirmed by Figure 10 which shows the output of the baseline system for a -1/3 p.u. step disturbance and controller output to plant input gain of 2/3. The response in this case is identical to the voting controller with one failed controller. This may be generalized by saying that a system with n redundant controllers, one of which has failed, is characterized by the response of the corresponding nonredundant system with forward gain (at the point of introduction of the voter) reduced by a factor of (n-1)/n.

The preceding conclusion is further extended as follows: If a system with n redundant controllers is constructed which requires a minimum gain in the forward path at the point the voter will be introduced to be stable, and a (n-1)/n reduction of that gain results in an unstable (right half-plane) pole in the system, then the introduction of redundant controllers will cause the system to be unstable in the event of a single-controller failure. An example of such a (non redundant) system occurs in [1] 4.14 pp. 118 - 119 for a multiple-loop system, and examples of general forms of some such transfer functions are given in [2] pp. 296 - 297.

This unfortunate variable-gain characteristic of averaging voters complicates the design process for a redundant system controller, which by definition must function with one controller of the three out of service, and should survive the transient which occurs when one controller fails. This characteristic also changes the complexion of the problem, since the voter must now be characterized as piecewise-linear at best, with a discontinuity (nonlinearity) when a controller fails. Two separate analysis are thus required, one for the normal (all controllers functioning) system and a second modeling the system with two controllers functioning, and a disturbance substituted for the third controller. An exhaustive analysis could thus be made by substituting appropriate failure-behavior models for the third controller, or by using root locus techniques to predict the behavior of second-order systems, using gain at the voter as the parameter. Under the piecewise-linear model an appropriate nonredundant system may be used to predict

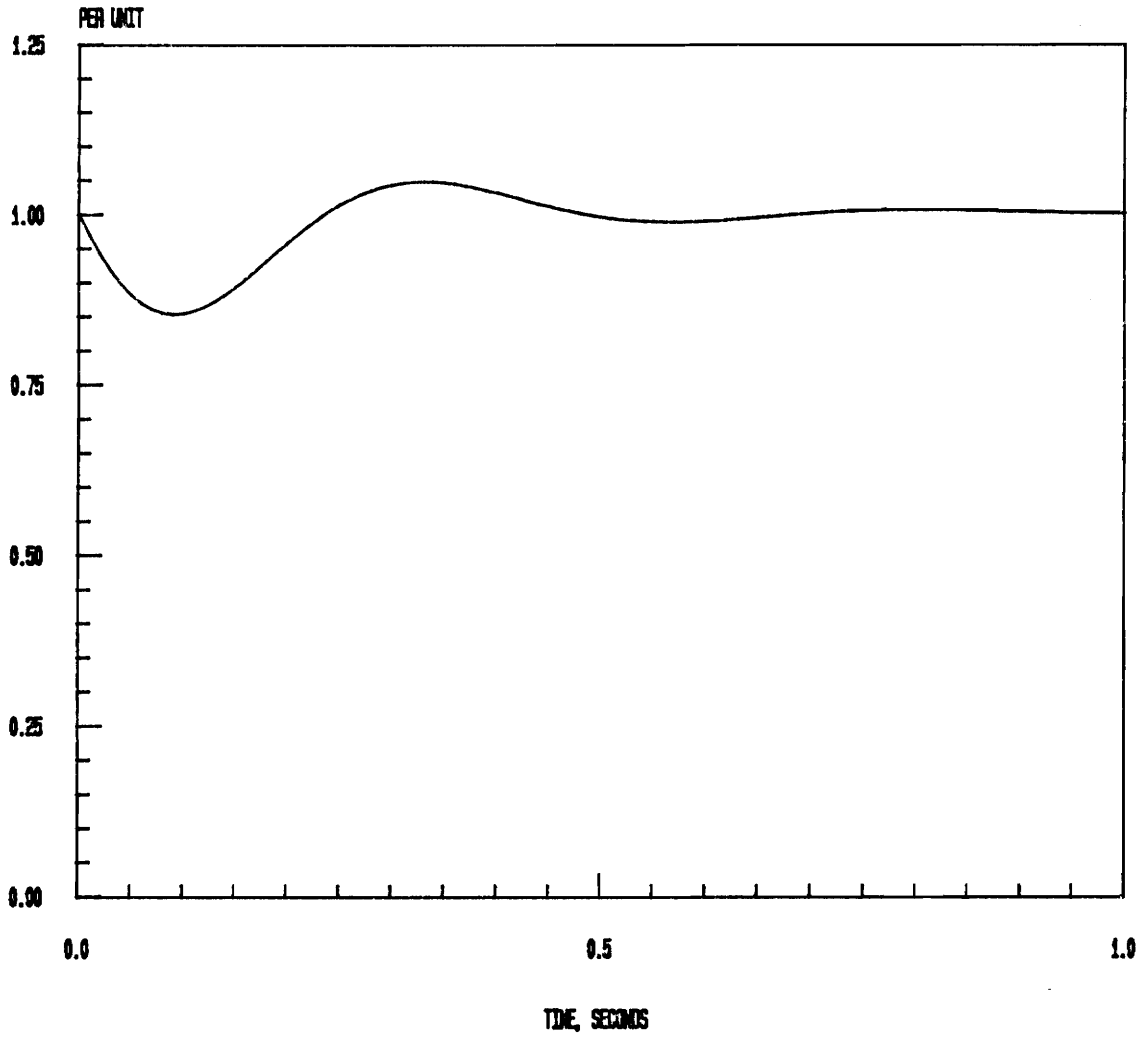

Figure 10
Reduced-gain Non-redundant System Response

the redundant-controller system by making the adjustment in forward gain described above, allowing use of a reduced-order model.

It is not necessary to have a system with model characteristics referenced above to be unable to produce an acceptable design. If the plant is sufficiently sensitive to gain at the point of the voter introduction into the system, it may be impossible to choose a value of gain such that operation is acceptable in both cases (i.e. either two or three controllers of the set operational). It is not conceptually difficult to hypothesize a system in which the choice of the gain at the point of the voter will not tolerate the expected range of operation (i.e. 2/3 p.u. to 1 p.u.) and still produce acceptable operation, since the minimum value of gain required to reduce the transient which occurs on controller failure to acceptable values may be such that with all three controllers operating the system may exhibit the excessive overshoot usually associated with high gain, or may no longer be stable.

The introduction of additional controllers reduces the effect of a single failure, since the range of the gain is in general (n-1)/n to 1. This is seldom a useful technique, both for economic and practical reasons: The number of points in a system where a voter can be introduced is limited, and construction of a voter with more than three inputs is difficult. Voting is, of course, possible in a variety of places. Useful voting i.e. voting which improves overall system reliability, however, is much more limited. For this particular system, introduction of voting further "upstream" of the servo valve coils is of little value since the coil and its associated driver are the least reliable elements of the system, and to significantly improve the reliability of the system they must be redundant. This situation is not unique to this particular system, as the actual input/output (I/O) hardware including sensors is usually the least reliable element of the overall control. It is also the area in which it is most difficult to implement voting.

NONIDEAL SYSTEM WITH AVERAGING VOTER

In the analysis of the preceding system the model used was ideal in the sense that the redundant controllers are mathematically identical. This is seldom the case in real systems, since the non-digital portions of the system exhibit small variations in the parameters which characterize them. It has proven impossible to construct systems in which the performance is mathematically identical. In this section we introduce this reality by slightly altering the parameters in the feedback gain of the controller to simulate the effect of non-identical analog components in the system. It is not the intent of this analysis to characterize the effect a variations in a given parameter but rather to demonstrate the result on system performance. For this demonstration it is sufficient to perturb a convenient parameter and look at the result on system performance.

Using the model of Figure 8, a 0.1% variation in the controller feedback gain is used to illustrate the problem. The feedback gain blocks feeding d/dt X2, d/dt X4, and d/dt X6 are set to .999, 1.000, and 1.001 respectively while the forward (reference) gain remains at 1.000. A limit function, described in detail in Appendix B, must be introduced on the state variables X2, X4, and X6 to prevent overflow in the calculation. This is not an artificial constraint, since the presence of the limit is implicit in the physical system. The value of 1.0 is used for the limit.

Figure 11a shows the transient response of the system to a 1 p.u. step reference applied to all three controllers, resulting in 38.8 percent overshoot and 0.19 seconds time-to-peak, which implies a damping ratio of 0.2885 and a natural frequency of 17.27 rad/sec. This transient response is identical to the theoretical response of the non redundant system shown in Figure 6. Figure 11b shows the behavior of the controller outputs as they go to steady-state. As might be expected, X2 ramps to positive saturation (limit), X6 to negative saturation, and X4 assumes the value 1.0 p.u. required to control the system. Figure 12 shows the response of this system, using as initial

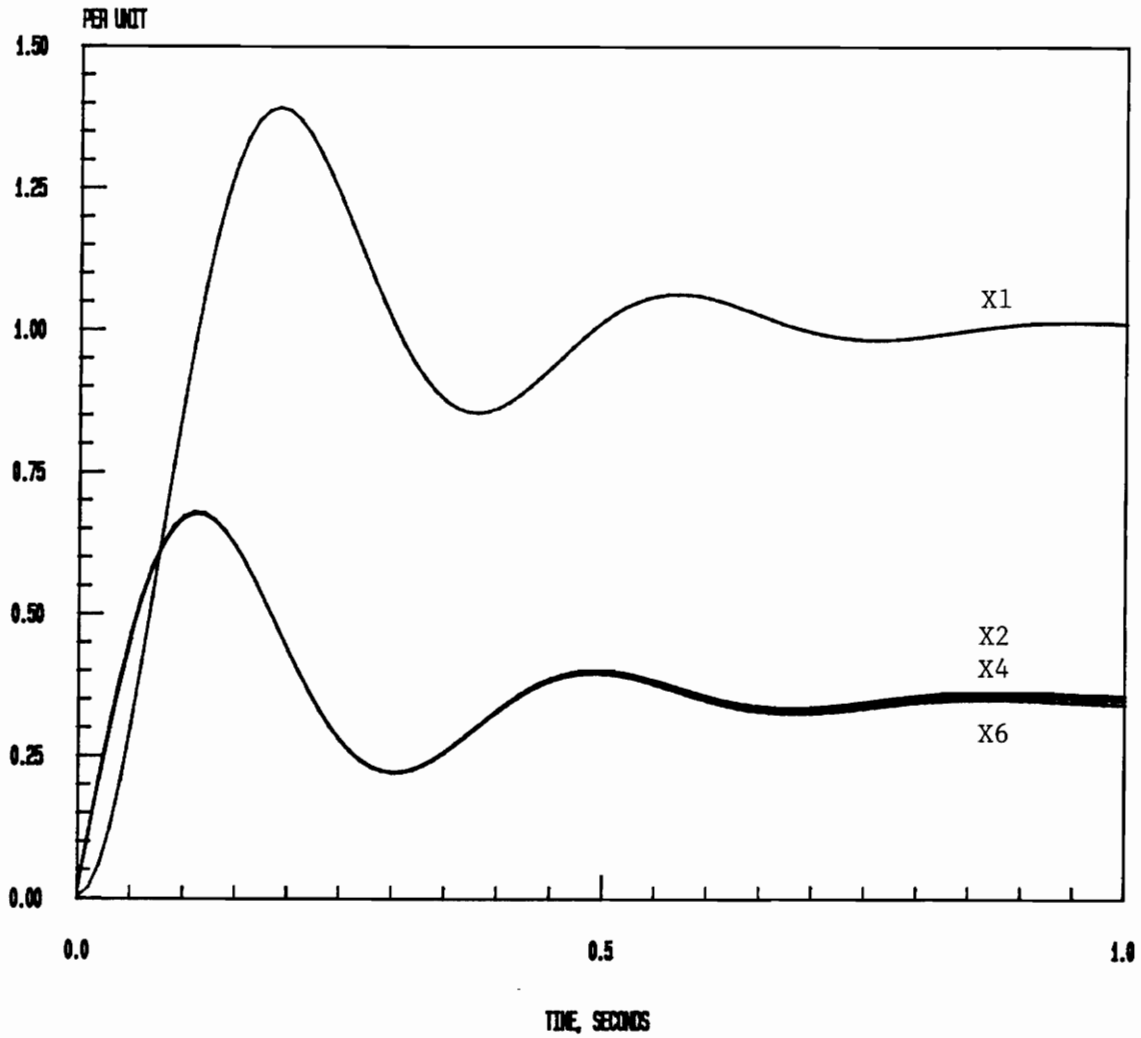

Figure 11a
Non-ideal Redundant System Response

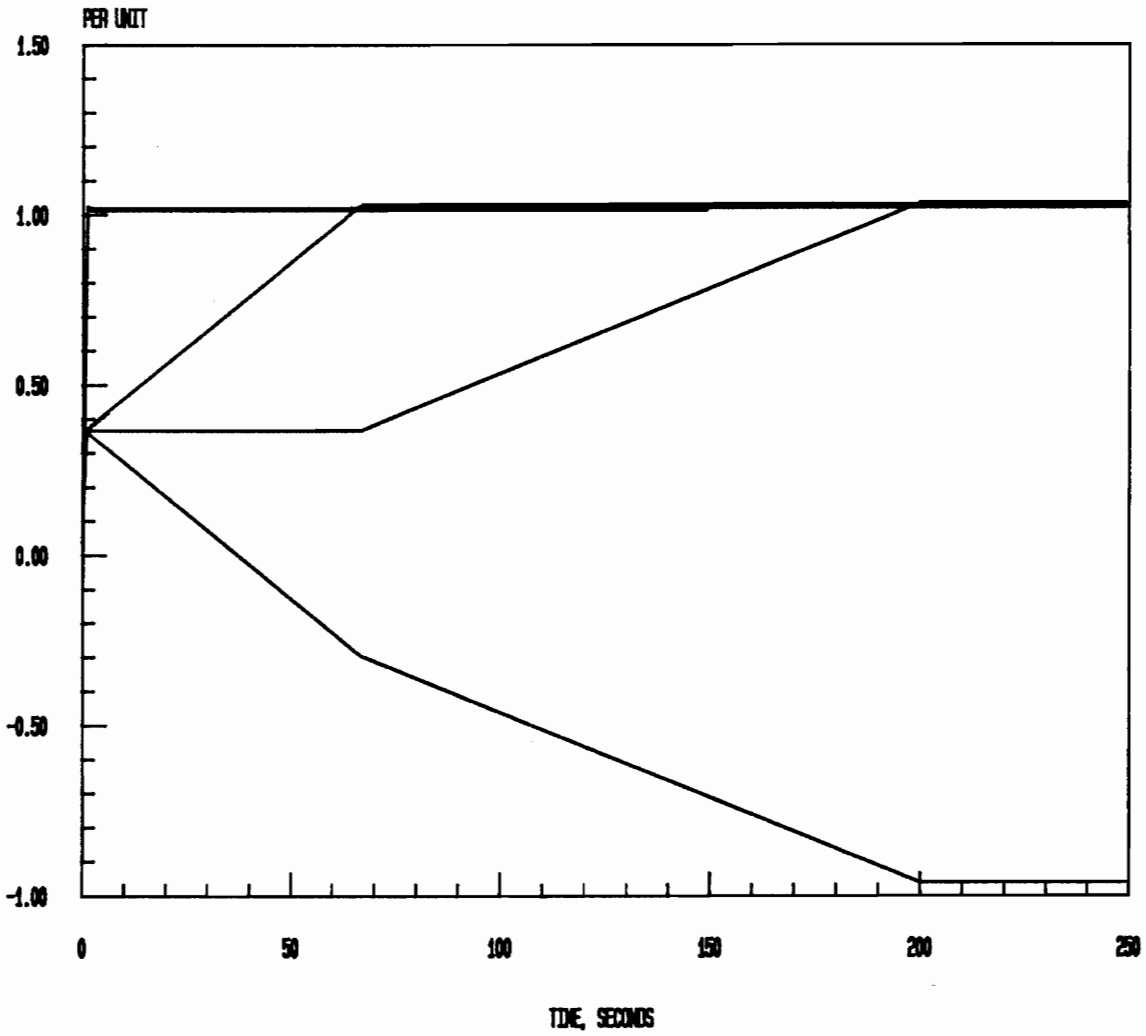

Figure 11b
Non-ideal Redundant System Steady-state Response

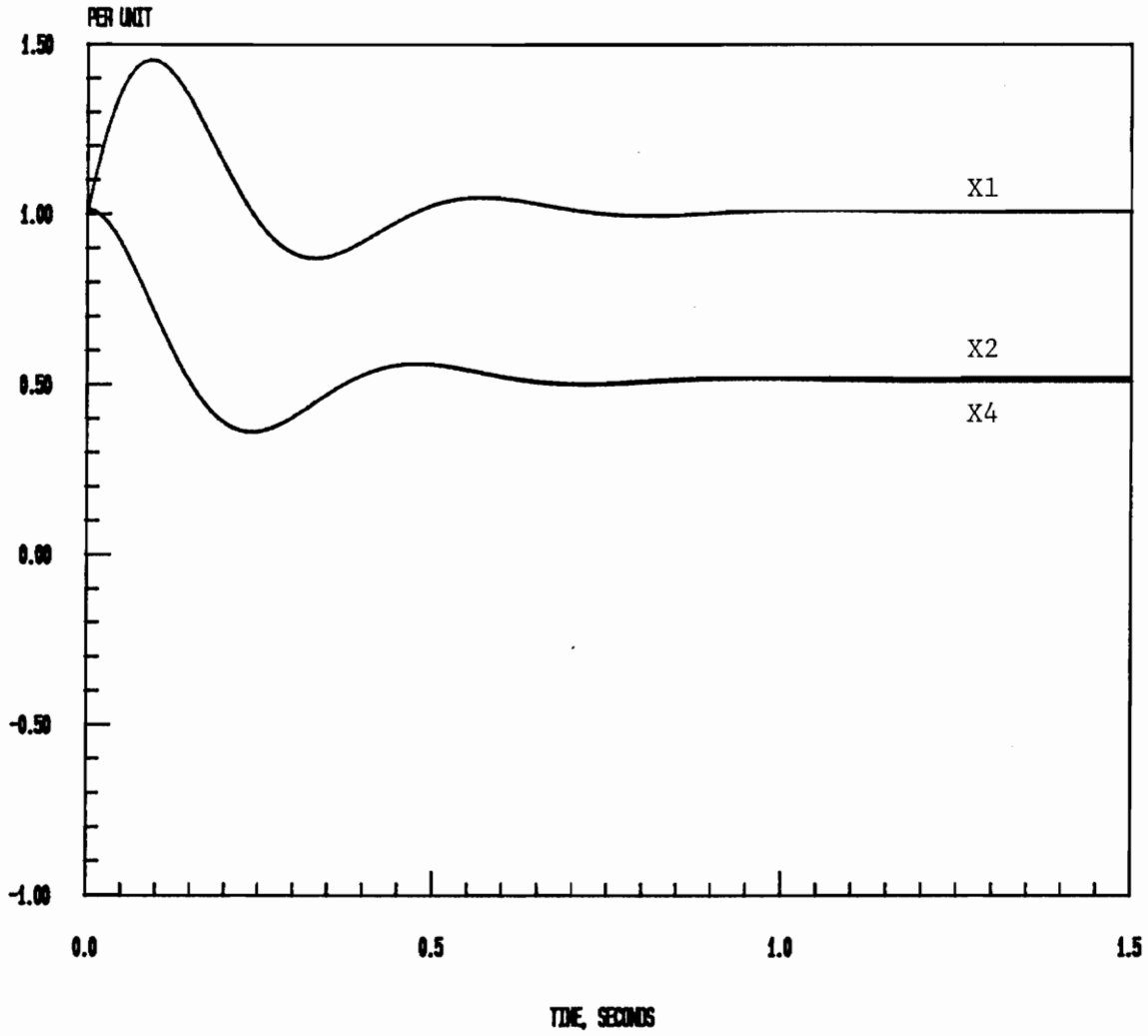

Figure 12
Non-ideal Redundant System Disturbance Response
(Loss of X6)

values the final (steady-state) values from the simulation of Figure 11b, to a step disturbance applied to the voter summing junction (which models the servo coil). This disturbance is created by disconnecting the X6 input from the summing junction to simulate controller failure. The system responds to this disturbance by increasing fuel flow, reaching a peak value of 1.448 p.u. in 0.08 seconds. When the new steady-state is reached approximately 100 seconds after the disturbance the two functional controller outputs X2 and X4 have again reached their expected values of positive limit for X2 and zero for X4 required for a net input to the plant of 1.0 p.u.

The system disturbance which results from a controller failure in this nonideal system is considerably worse than the resulting disturbance in the ideal system shown in Figure 9. This is largely due to the magnitude of the step which is introduced by the controller failure. Figure 13 shows the output response when the X2 controller fails, and Figure 14 the response when the X4 controller fails. Comparing the three outputs we see that the magnitude of the output disturbance varies depending on which controller fails. In Figure 12 the total input applied to the plant is (X2 + X4 + X6). Immediately before the simulated controller failure the sum is (+1) + (+1) + (-1) = +1. Immediately after the failure the sum is (+1) + (+1) = +2, for a step input of magnitude +1. In Figures 13 and 14 the plant input immediately after failure is (+1) + (-1) = 0 resulting in a step input of magnitude -1. If the limit function does not interfere with the transient response in this model, the output of the redundant system should be predicted using the reduced-gain baseline model similar to the one used to produce Figure 10. In Figure 15 the output which results from a step disturbance of +1 applied to the reduced-gain nonredundant model shows response identical to the response in Figure 12 as predicted. In Figure 16, however, the output resulting from a step disturbance of -1 applied to the reduced-gain baseline model is significantly different from the response in Figures 13 and 14. On closer examination we see in Figures 13 and 14 that only one of the controllers begins to respond immediately to the disturbance, while the other remains at limit until the output changes sufficiently for the

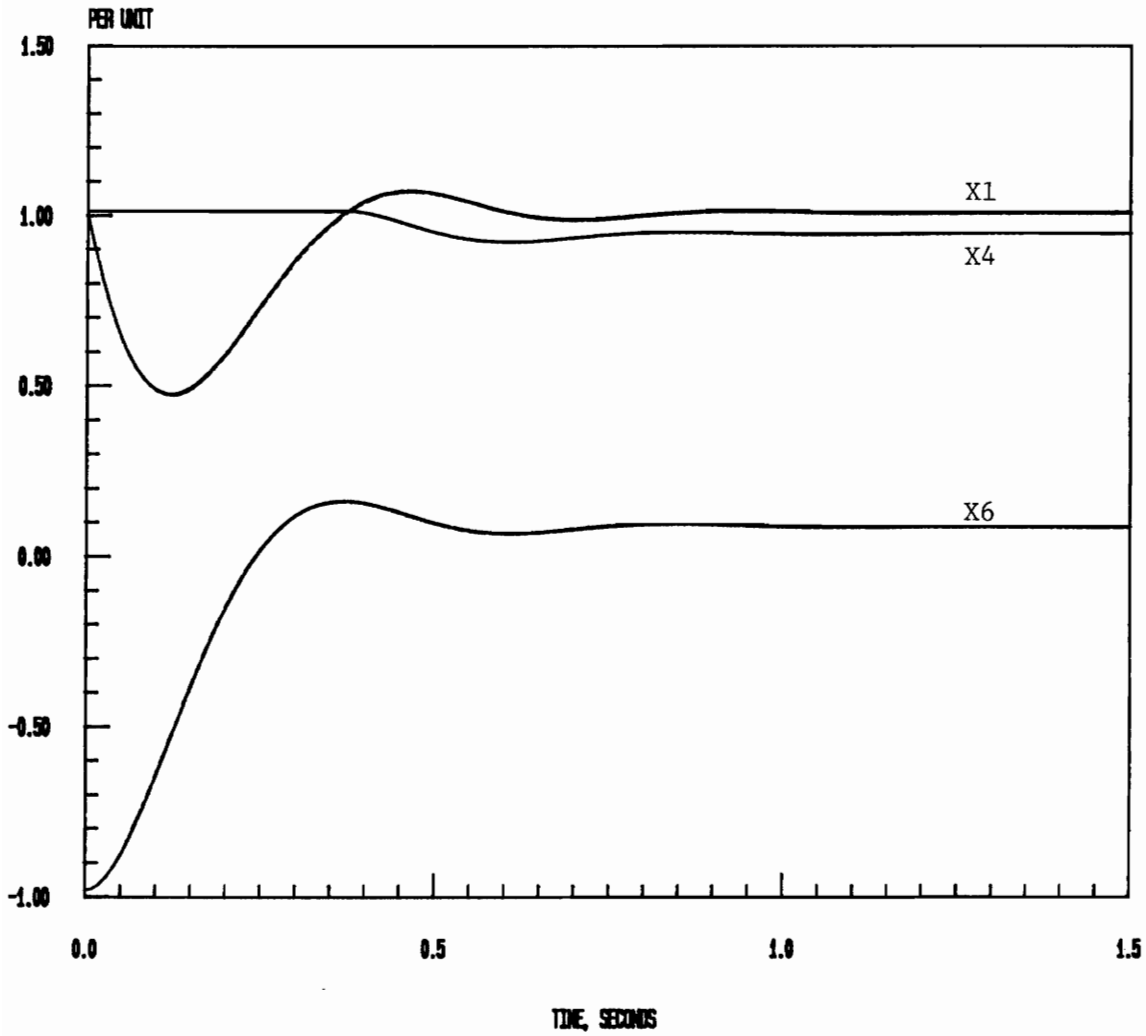

Figure 13
Non-ideal Redundant System Disturbance Response
(Loss of X2)

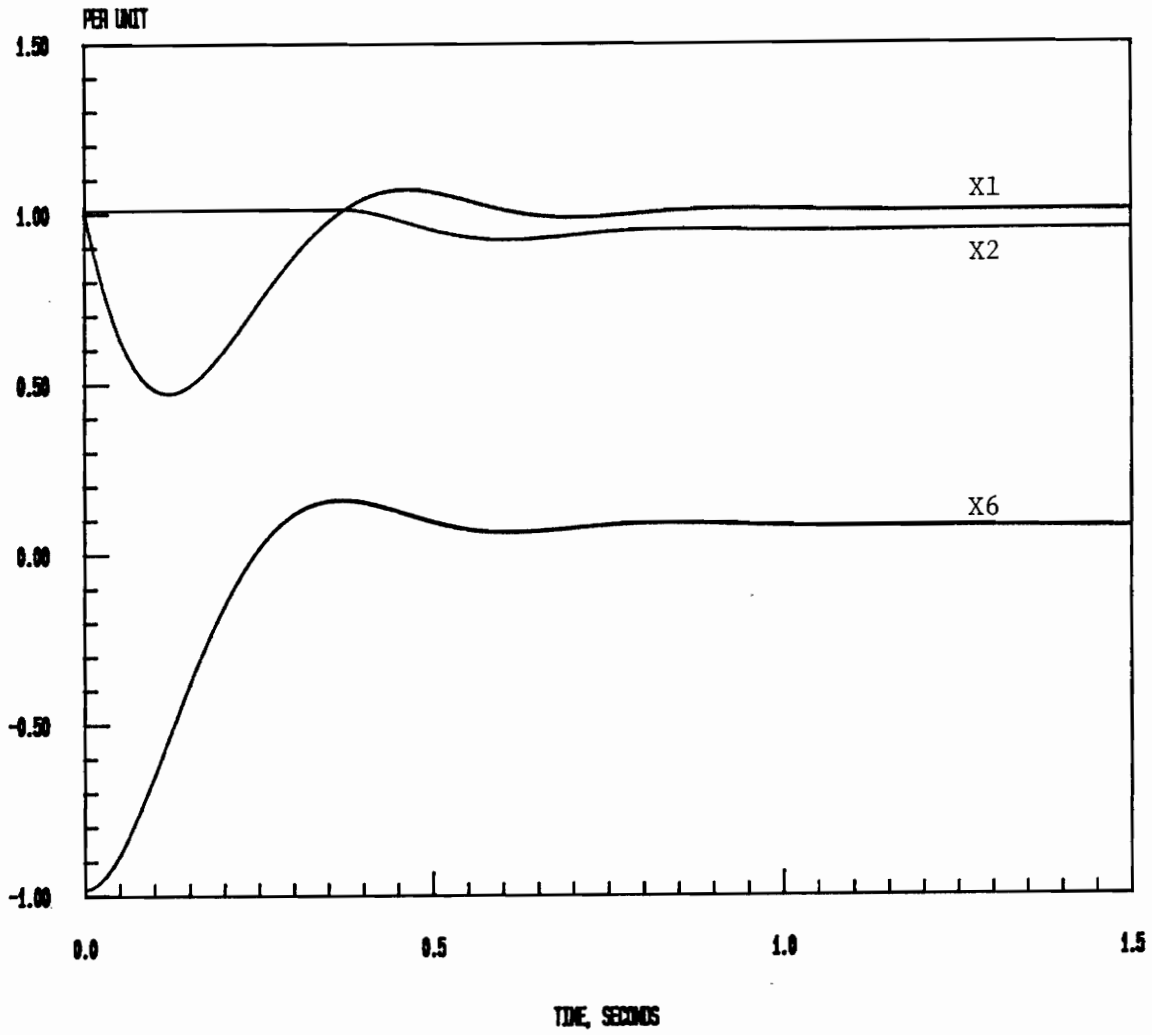

Figure 14
Non-ideal Redundant System Disturbance Response
(Loss of X4)

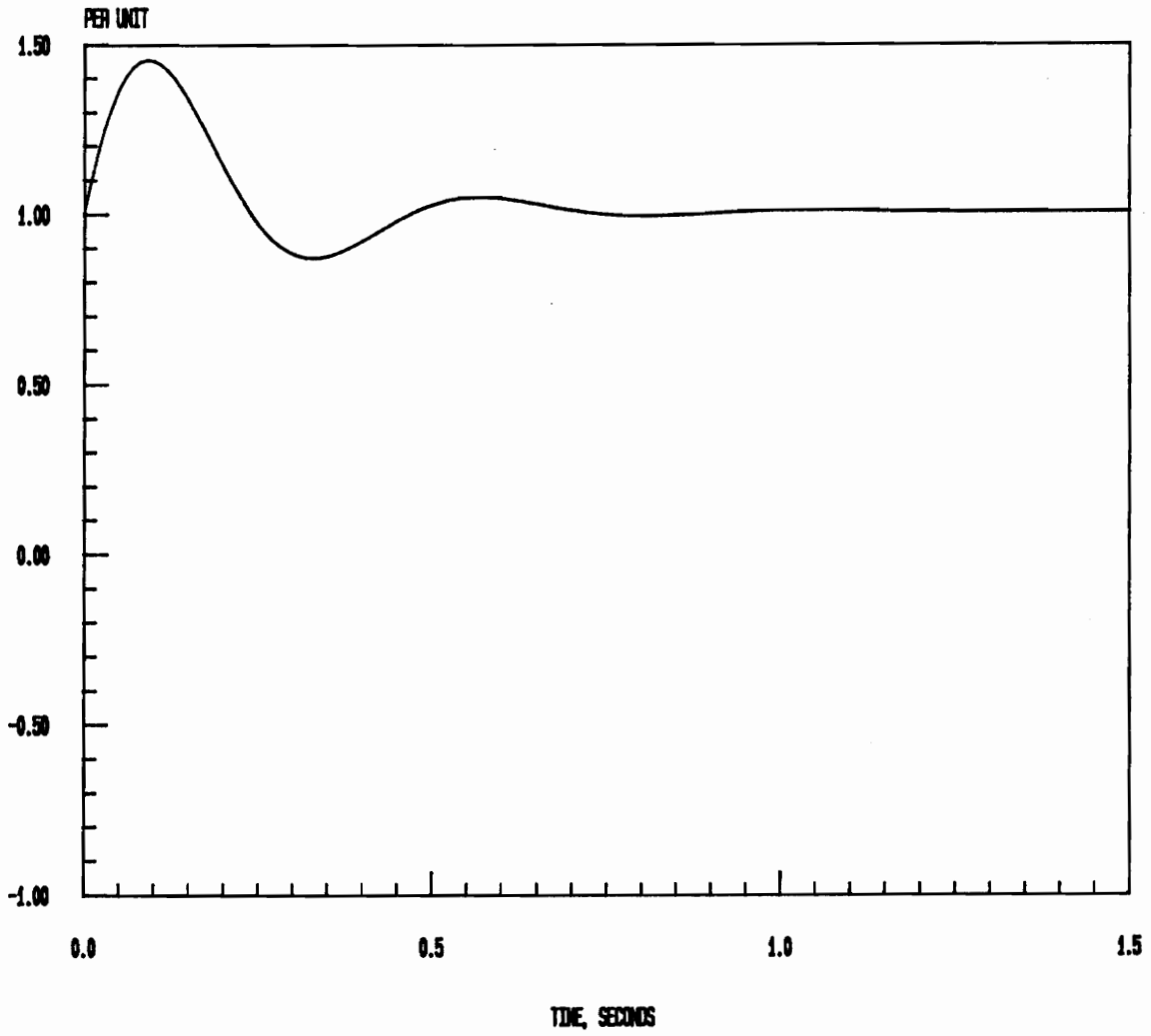

Figure 15
Non-redundant System Disturbance Response
(+1 p.u. disturbance, G = 20)

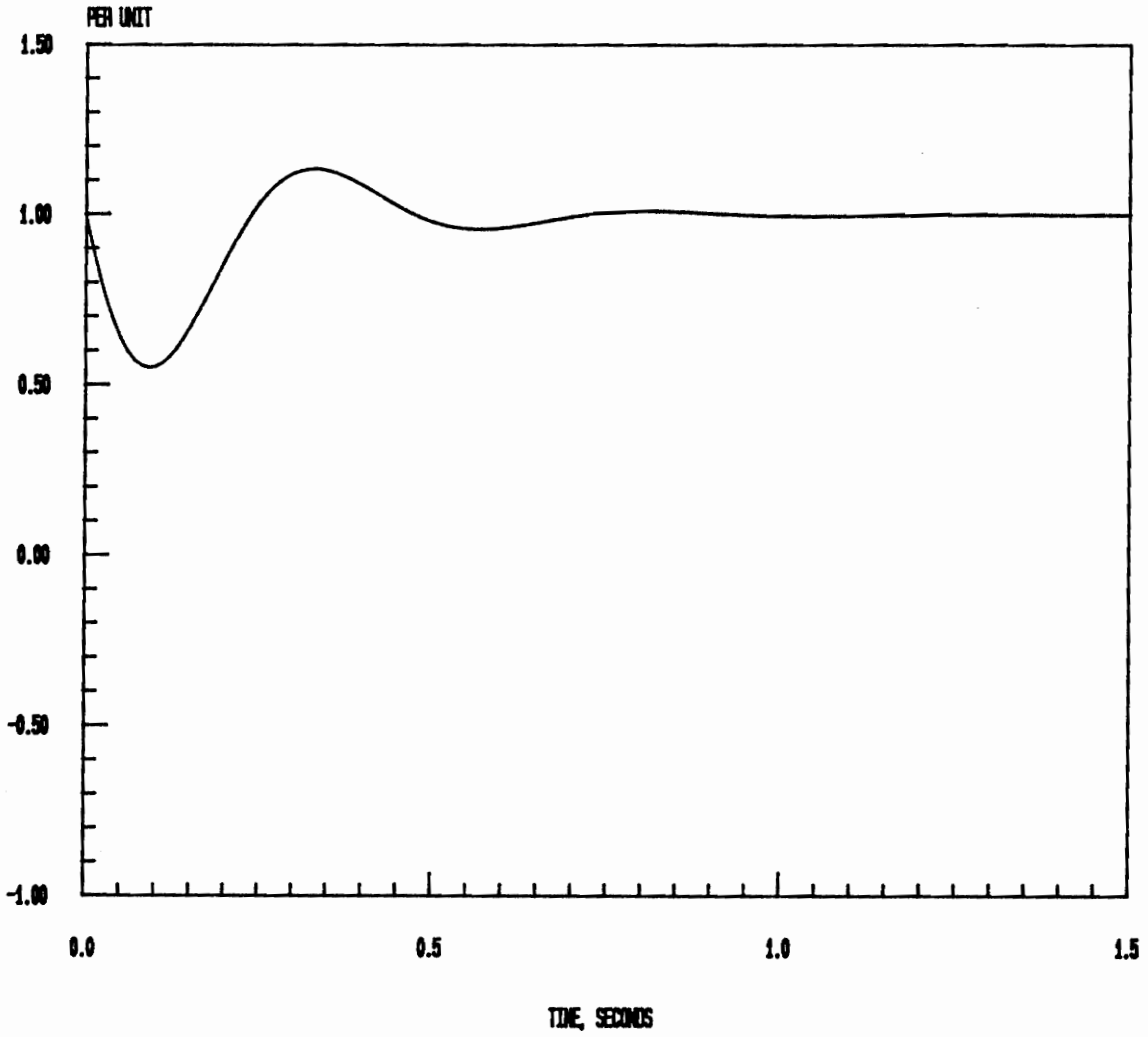

Figure 16
Non-redundant System Disturbance Response
(-1 p.u. disturbance, G = 20)

error signal applied to the input of the integrator to change sign and begin to bring the integrator out of limit. The result is slower response with more overshoot in the redundant system since for a short period of time the effective forward gain of the control system is 1/3 instead of the 2/3 gain of the nonredundant model. When the second controller comes out of limit (about .38 seconds after the disturbance is applied) the effective forward gain again becomes 2/3, but with different initial conditions from the situation in Figure 12. The reduced-gain nonredundant model has -44.8 percent overshoot and 0.09 seconds time-to-peak, while the redundant controller model has -53.4 percent overshoot and 0.12 seconds time-to-peak.

From the preceding analysis we can conclude that the behavior of nonideal redundant controllers differs significantly from ideal redundant controllers:

1. The magnitude of the disturbance which must be applied to simulate performance when a controller fails is different and must be calculated differently depending on which controller fails.

2. The output transient which results is larger than for the ideal redundant controller, since the disturbance input is larger, and thus the performance of the system is degraded.

3. The forward path gain is not constant, and may assume three discrete values with a range of 3:1 - when all three controllers are in the active range the gain is 1 p.u., when two controllers are in the active range it is 2/3 p.u., and when only one controller is in the active range it is 1/3 p.u. The third case is the model which should be used to model small-signal disturbances other than controller failure, since unless and until the error is large enough to bring the other two out of limit only one controller will be contributing control effort.

Because of these characteristics, the full (nonlinear redundant) model should be used to simulate system performance.

Note that when only two controllers are operational a steady-state error may exist in the output due to the error in the net feedback gain. The steady-state equation describing the operating point of the system is different for each controller, and only one of the equations may be satisfied. The other controller will go to one limit or the other, and the plant output will be determined by the feedback gain of the remaining controller. If the gain of that controller is not exact then the plant output will exhibit steady-state error.

NONIDEAL SYSTEM WITH MID-VALUE SELECT OUTPUT VOTING

The nonideal redundant controller with averaging voter has two quite undesireable characteristics - the large output transient which occurs on controller failure due to the magnitude of the disturbance applied to the input of the plant, and the nonlinear gain characteristic which makes prediction of performance difficult. The first problem is perhaps more serious in actual application, since transients of +/- 50% are normally considered quite large. They may not affect the overall power system if it is sufficiently stiff since they are quite fast relative to system time constants, but they do represent a torque transient in the output shaft and conceivably could excite a torsional vibration mode in the mechanical system. The second problem is largely theoretical in the case of the gas turbine fuel system since this system is relatively insensitive to changes in the gain parameter at this point, but it can (and does) require that a large number of separate analysis be performed in order to predict turbine performance under a variety of different system operating conditions i.e. which controller fails and the exact failure mode which occurs.

One solution to the basic problem is to artificially force the three controllers to hold the same operating point in their active range. This tends to reduce both of the problems of the nonideal system by reducing the magnitude of the applied disturbance to approximately the same as in the ideal case and by allowing both remaining controllers to contribute to reducing the transient i.e. maintain the forward-path gain at 2/3 instead of allowing operation for some time at a gain of 1/3. This can be accomplished in several ways.

The method currently used is the addition of a "trim" input at appropriate locations in the controller to drive the three controllers to the same operating point. The trim is determined by ad hoc calculations in which each controller looks at data on the operating point of the other two and provides an appropriate trim signal to cause the three controllers to converge. This method works but is difficult to implement and generally must be altered with each change to the

control algorithms. Proof of acceptable performance is generally experimental and difficult to predict in advance, since it is difficult to model and nearly impossible to analyze. It also requires accurate, timely, and reliable data exchange between the three controllers, which with current hardware architecture places a heavy burden on communications between the three controllers and the <C> communications processor. The <C> communicator is required to be functional for the scheme to work.

A second method under consideration for the next generation of the system uses "state voting" in which a separate fault-tolerant communications system provides mid-value select voting of the complete state of the three controllers. This system would for each controller during each control interval acquire the complete state of the other two controllers, perform the mid-value select voting, and make the results available to the host controller in time for the next control calculation. Potentially the system can vote all inputs, outputs, and internal states to "reconverge" the three controllers at each control interval by eliminating the effect of parameter variations in the individual controllers.

Before considering the controller convergence approach, simple mid-value select voting of the outputs only should be considered. This section will consider a simplified redundant controller with mid-value select voting, as shown in Figure 17. An additional state variable X8 is assigned to the voter output and the inputs X2, X4, and X6 to the voter are considered to be sampled when the input for the STATEVAR program is prepared. The state variables X2, X4, and X6 are limited as in the previous section. The voter algorithm, FUNC5 listed in Appendix C, is

$$X8 = 3 * (MID(X2, X4, X6))$$

where MID describes the process of selecting the middle value from the set X2, X4, and X6. This simulation does not explicitly show the summation of the three servo coils. The simplification is made largely to reduce the order of the model, and the implications of the simplification will be discussed later.

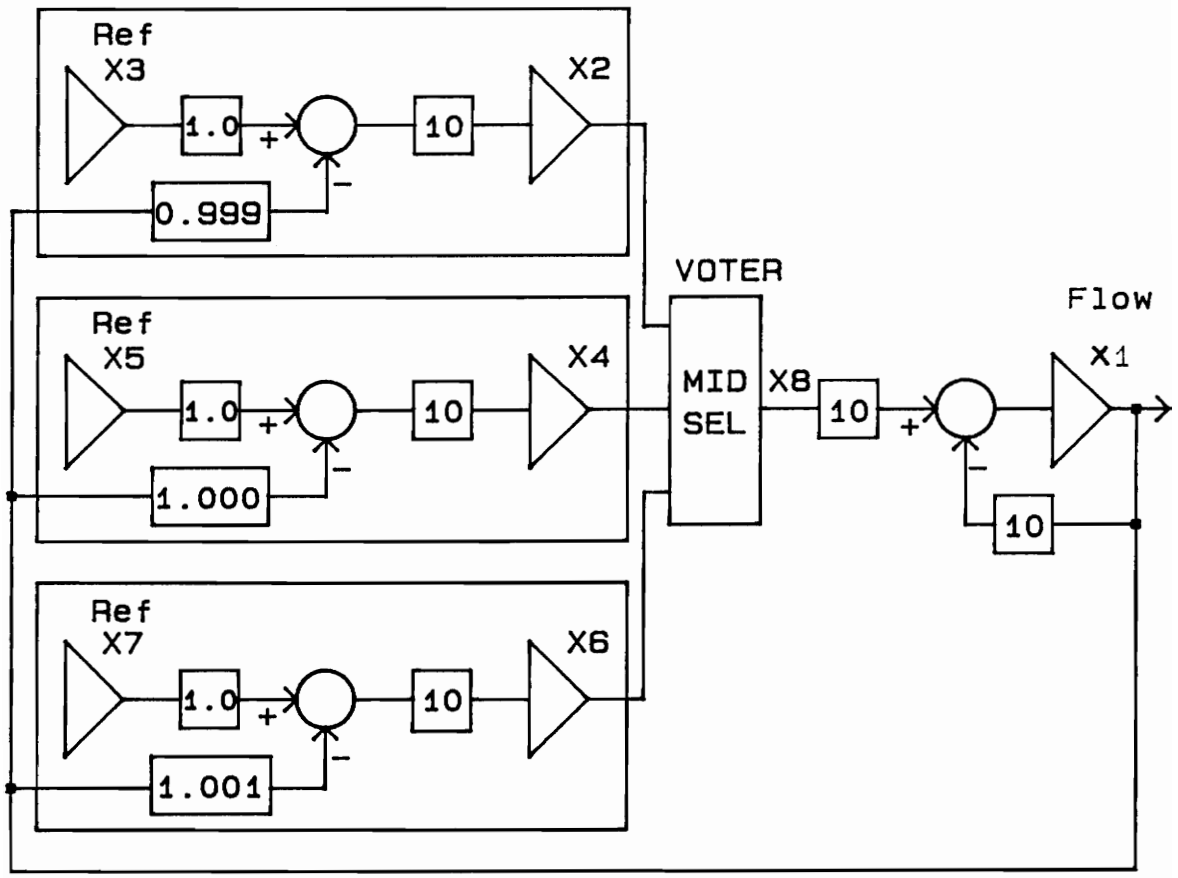

Figure 17
Non-ideal Redundant System with Mid-value Select Voter

For this simulation the model of Figure 17 and the voter described above and in Appendix C were used in the STATEVAR program. Again a run was first made to establish the steady-state conditions: In steady-state X2 is in positive limit (+1), X6 is in negative limit (-1) and X4 is controlling with a steady-state value of 0.333 (due to the gain of 3 in the voter). A second run was made using those initial conditions and with the X4 state variable disconnected from the voter to simulate controller failure. Due to the nature of the voter subroutine and the operation of the STATEVAR program this was accomplished by setting the inputs to d/dt X4 to zero and the inital condition on X4 to zero. If the voter is considered as the summing junction of the previous model then the effect of this technique is similar to the method used in previous simulations.

The performance of the model in the case of single-controller failure is shown in Figure 18. The peak overshoot is -85 percent with time-to-peak of 0.20 seconds for a step disturbance of -1/3 p.u., the worst system performance of any scheme considered so far! This simulation demonstrates the problem with the mid-value select voting of outputs only: If a controller fails in such a manner that the its output assumes the mid-value the voter will follow that signal until one or the other of the remaining two controllers can become the middle value, with higher gain than before. Again the failure mode must be carefully defined before performance can be predicted. If the failure mode is such that the failed controller assumes the mid-value a large disturbance is likely to result. In this simulation a relatively small disturbance caused a large transient in the output, since the failed controller remained the mid-value for considerable time (approximately 0.19 seconds).

Other failure modes which at first may look severe could in fact produce smaller disturbances. For example Figure 19 shows the transient resulting from the same initial conditions as Figure 18, but with the controller whose output is X6 failing by stepping from -1.0 p.u. to +1.0 p.u. In this case the mid-value switches from X4 at 1/3 p.u. to 1.0 p.u. (either X6 or X4) for a step disturbance of +2/3 compared with the -1/3

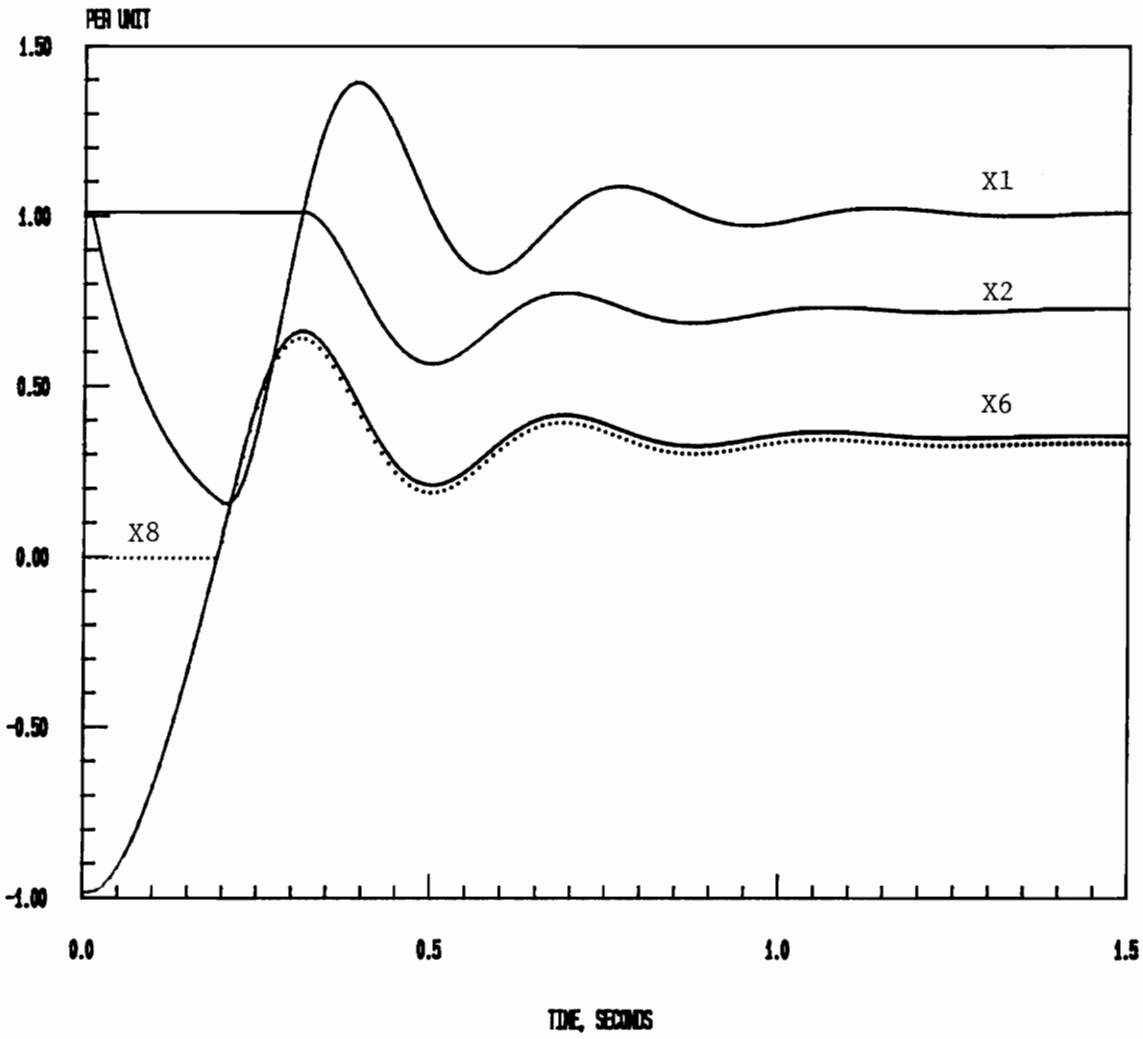

Figure 18
Disturbance Response of Mid-value Select Voter System
(Loss of X4)

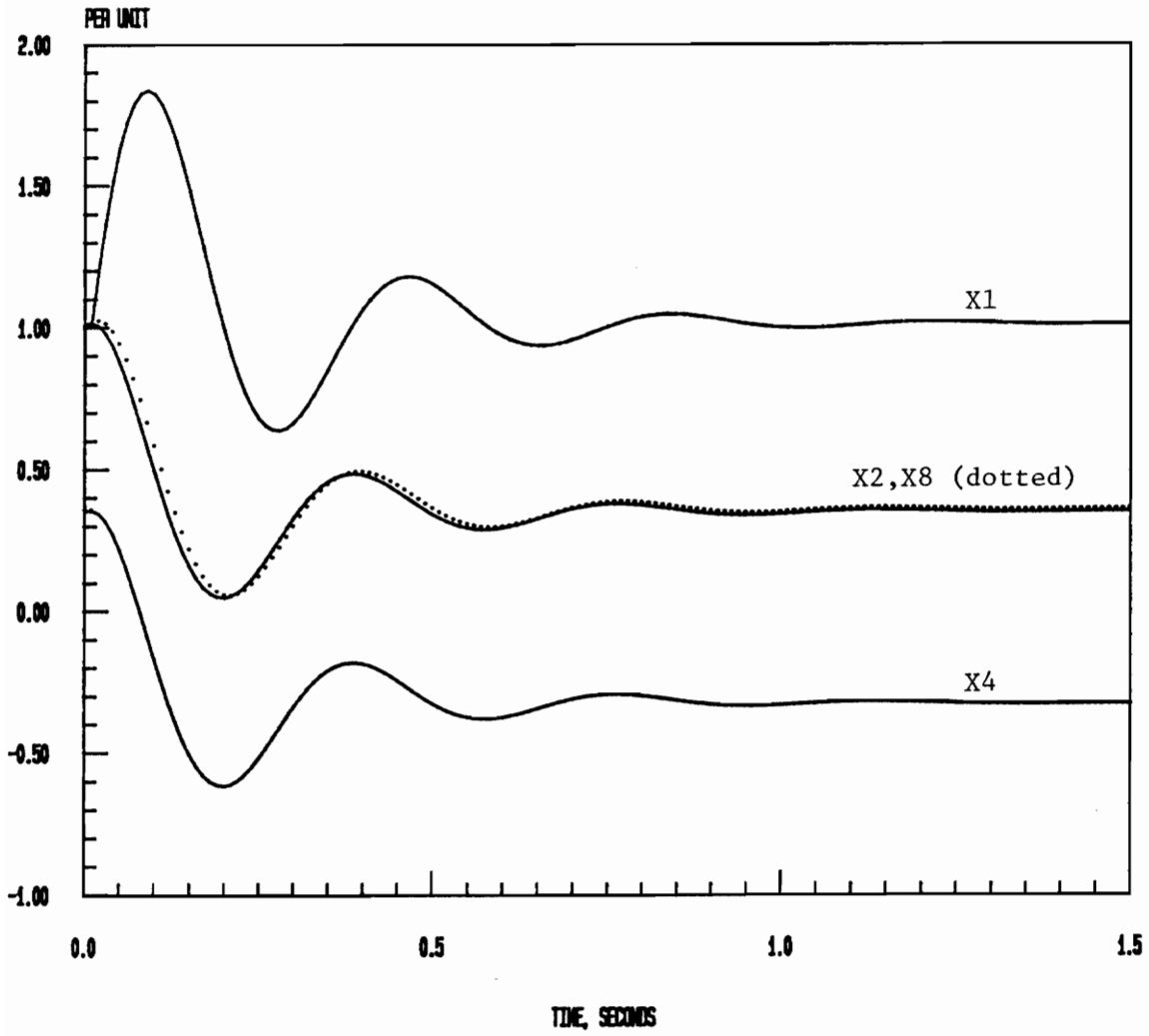

Figure 19
Disturbance Response of Mid-value Select Voter System
(Step of X6 from -1 to +1 p.u.)

step disturbance of Figure 18. In this mode, however, the failed controller remains the min-value for only a very short period, with X2 quickly assuming the mid-value. The result is a transient with 82.7 percent peak overshoot and time-to-peak of 0.09 seconds - a smaller transient for a disturbance of twice the magnitude. Although it was not immediately apparent, the problem with this output voting scheme is a form of windup - the delay between the need for action and a controller getting into the active control region.

This model was designed to test controller failure, and appears as a single nonredundant servo coil in the model. If in fact the voter had three outputs, one of which failed, the situation would be nearly identical to the ideal controller considered earlier: The coil failure would simultaneously introduce a -1/3 p.u. step disturbance and reduce forward gain by 1/3. Provided that the dynamics of the controllers were such that the controller which was at the mid-value when the disturbance occured remained the mid-value for the duration of the transient, due to the presence of the mid-value selector the redundant controllers would appear as a single controller with the same dynamics as the two ideal controllers. If, however, the dynamics were such that during the period of the transient the mid-value selector chose different controllers during different portions of the interval then a nonlinearity would be introduced, with corresponding difficulty in analysis. Simulation would provide results for a given set of initial conditions but the number of possible sets of initial conditions can be quite large.

NONIDEAL SYSTEM WITH MID-VALUE SELECT FEEDBACK VOTING

Figure 20 is the state model for a system with mid-value select voting in the feedback path instead of the forward path. Since mid-value select voting in the forward path has been shown to be undesirable with non-synchronized controllers and of marginal benefit with ideal (synchronized) controllers, an averaging voter is used in the forward path. The system is considered to have three sensors modeled as feedbacks of slightly different gains. The output of the gain is modeled as a sample-and-hold state variable, which is reasonable for a digital mid-value select voter. The voter itself uses the same algorithm as the preceding example with the appropriate state variables as inputs (similar to the routine shown in Appendix C). Since all three controllers receive the same feedback (middle value from the three sensors) they remain synchronized as in the ideal case; in fact the performance for a loss-of-controller failure is identical to Figure 9.

The results of the analysis of a nonideal controller with mid-value select voting of the outputs suggest examining the result of a loss-of-feedback failure. Figure 21 shows the response of the system to the loss of the X10 feedback signal, simulated by using the steady-state conditions as initial conditions and setting the gain in the X10 path to zero. The resulting transient is on the order of 0.1%, and would be considered insignificant in the application.

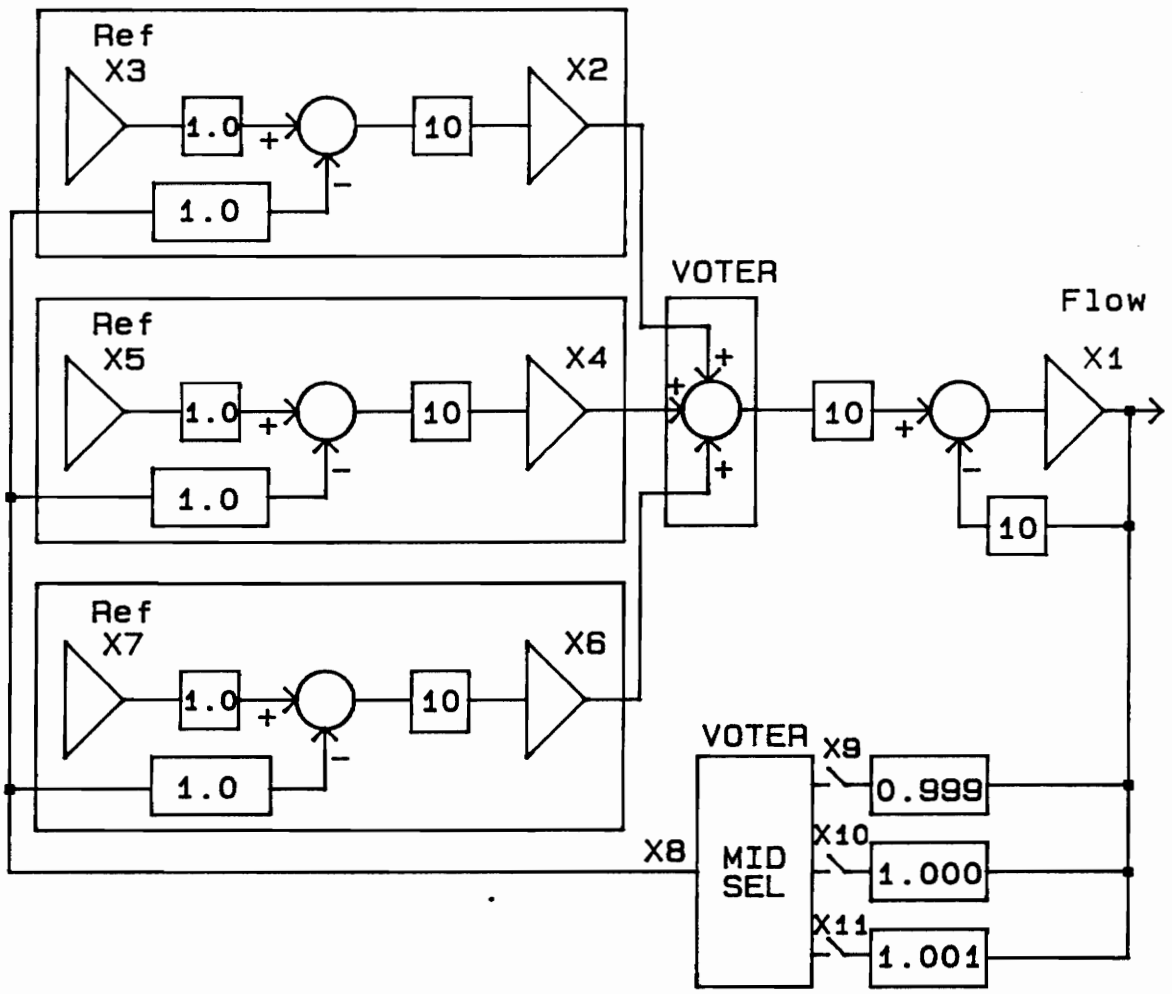

Figure 20
Controller with Voting Feedback

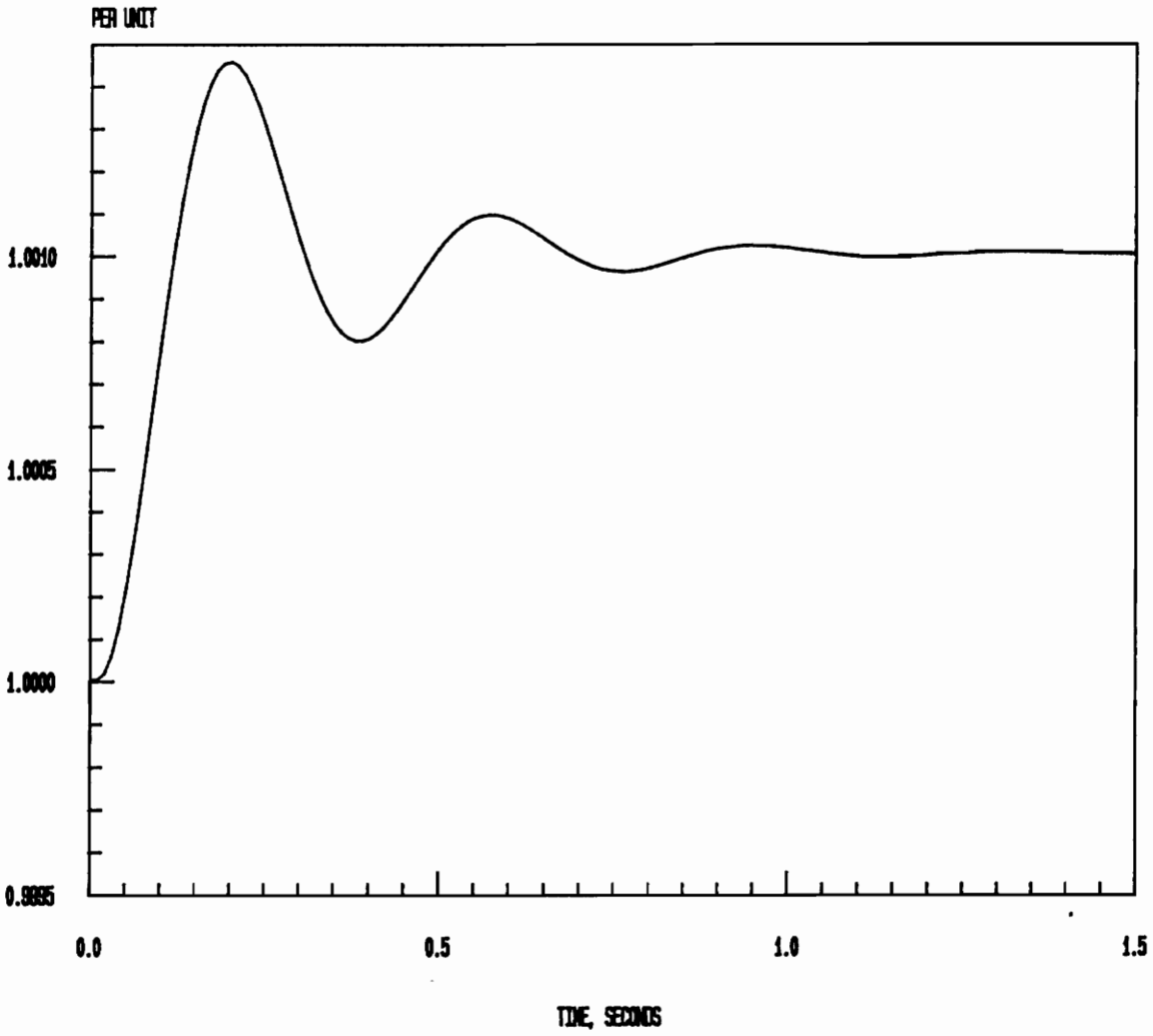

Figure 21
Feedback Failure with Feedback Voting

RESULTS

APPLICABILITY OF THEORETICAL ANALYSIS

Control system with redundant controllers under certain restrictions are amenable to analysis with conventional theoretical techniques such as root locus, etc. Unfortunately the restriction is that the voting function must be linear, which is difficult to acomplish under most common failure modes since most useful voters exhibit non linearities when one input fails. Piecewise-linear analysis, with separate linear analysis for different conditions, may be used but the number of possible sets of conditions which must be considered can become large.

It is possible to characterize certain voters as a variable gain plus a step disturbance. The range of variation of the gain may then be used as the parameter of variation for a root locus analysis which can predict the range of performance of the system. By designing the system to be relatively insensitive to gain at the point of voter introduction the effect of the variable gain is minimized and conventional techniques can be used to predict the effect use to the disturbance. If the system is excessively sensitive to gain at that point conventional techniques can be used to furnish bounds on the range of operation. Predictions from such analysis techniques should be confirmed by nonlinear simulation.

Other types of voters cannot be modeled by anything approaching a linear approximation. In systems using such voters linear analysis techniques are of little use and misapplication of the techniques may lead to severly erroneous conclusions. Nonlinear simulation may be used to predict system performance, but a separate simulation is required for each failure mode which can occur. Since it is not feasible to run simulations for all such modes, a subset of representative test cases must be chosen. The selection of test cases requires thought since some seemingly innocuous situations can yield suprisingly different results due to subtleties in voter behavior.

VOTER PERFORMANCE

The averaging voter was found to fall in the variable gain plus disturbance modeling class. As such its performance in systems is relatively predictable, although the result from some types of failures may be unacceptable in terms of the effect on the system. The averaging voter is perhaps most useful in achieving the desired goal of increased system reliability, since it is possible to implement an averaging voter at locations in the system where the voter has marked effect on reliability. It may also appear in systems using other types of voting in other places in the system. For single point failure analysis the averaging voter exhibits a gain range of (n-1)/n : 1 and a step disturbance of 1/n p.u. where n is the number of inputs which are being averaged, and thus the range of parameter variation may be reduced by applying additional inputs if practical in the actual system. In applications where system sensitivity is such that the variable-gain characteristic of this voter can be tolerated system performance under various conditions can be readily bounded by considering the extremes of the gain variations and step disturbances which result from single point failures using classical analysis techniques.

The averaging voter has one major problem in that the disturbance introduced when a single-point failure occurs can be quite large, and the resulting transient may drive other portions of the control system into nonlinear operation, changing the complextion of the problem. This is a secondary effect which must be considered before relying entirely on theoretical analysis to characterize system performance.

The mid-value select voter was found to fall into the totally non linear class, and is difficult to analyze using classical techniques. Each mode of operation must be considered separately since the parameters of the voter are dependant on the trajectories of the inputs to the voter, and when the voter selects a different input during the control interval the performance of the system changes. The mid-value select voter does have constant gain but the actual performance consists of a series of sub-intervals each with a different set of initial conditions. The transitions from one sub-interval to another are a

function of the dynamics of the controller and the system. If the dynamics and mode of operation are such that the selected input to the voter does not change during the control interval then (and only then) classical techniques will accurately predict system performance, since the system model reduces to the nonredundant model.

The constant-gain characteristic of the mid-value select voter performs best with ideal controllers. In this situation a single controller failure has no effect on the system. Implementation of a mid-value select voter in a manner which is useful in increasing system performance, however, is difficult.

CONTROLLER CONVERGENCE

Regardless of the type of voter used it is desireable to create a control system in which the controllers are "ideal". If the individual controllers each assume an equal share of the control effort at all times then the disturbance introduced by controller failure is minimized, and system performance is improved. Ideal controllers present some difficulty in implementation since very small parameter variations can cause divergence in individual outputs. If the controller integrates the error then the steady-state operation requires a limit function, either explicit or implicit, to maintain operation. Controllers which do not integrate the error will exhibit finite, predictable steady-state difference in operating points. The amount of variation between the operating point of the controllers partially determines the magnitude of the disturbance applied to the system when a failure occurs. With an averaging voter this can be the predominant factor in determining the magnitude of the resulting transient, and with a mid-value select voter determining which control output to apply to the plant divergence in the operating point actually makes the situation worse than it at first may appear, if the voter continues to track the failed controller for a significant portion of the control interval.

In controllers such as the Mark IV SPEEDTRONIC used as the basis for the simulations in this paper the control is a mixed analog-digital hybrid. The digital portion is readily made ideal, since the algorithms and constants used can be made mathematically identical and thus will yield identical results for a given set of inputs. Very small variations in the analog parameters such as analog to digital converter gain factors in the feedback loop have been demonstrated to produce divergence with corresponding degradation in system performance. The mid-value select voter, however, has been shown to be quite effective in eliminating the effect of the analog parameter variations with significant gains in overall system performance. Application of such voting represents a significant communications task since the controllers must exchange sufficient state information to perform the voting for each control interval and for typical systems tens and often

hundreds of states must be voted. The communications system must also be quite reliable and fault-tolerant since failure in the voting process can have unpredictable results on system operation.

DISCUSSION

This paper has investigated only four of a perhaps infinite number of topologies for redundant controller implementation. The topologies were chosen to be representative of control schemes being considered for application and in some cases actually in use. From this set of investigations three generalizations appear to emerge:

1. Maintaining convergence of controller outputs is a major factor in overall system performance regardless of the actual voting scheme involved. It would appear that effort to design practical systems need to concentrate on providing reliable convergence under as wide a spectrum of failure modes as is practical. Feedback voting after the last analog stage was shown to be quite effective in maintaining convergence in the face of single-point feedback failure.

2. Mid-value select output voting is very effective when controllers are converged, but average voting works better if convergence cannot be maintained.

3. Great care must be exercised before theoretical results are applied to redundant systems, since the voters are unavoidably nonlinear. The averaging voter is more amenable to piecewise-linear modeling than the mid-value select voter.

This investigation has suggested two areas in which further investigation is indicated. This investigation concentrated on the effect on a system when something failed. Since the broad goal of redundant controllers is to reduce the down time of a system, it is desireable (in fact typically a requirement) that after repair the failed portion of the system can be brought back into operation without shutting down the entire system i.e. this paper addresses what happens when something goes off-line but not what happens when you bring it back on-line. A specific question is whether or not mid-value select voting of feedbacks alone will re-converge a single controller without upseting the system. It appears that full state voting should effect the re-convergence but this remains to be proven.

A second area is how to improve the performance of averaging voters. In the specific system considered, the three-coil fuel control servo valve is considered to be mandatory to achieve reliability gains. Thus when the most common failure mode (loss of a single coil by open-circuit) occurs the disturbance and variable-gain characteristic must be considered regardless of how the rest of the system works. It appears that little can be done to change the magnitude of the disturbance which results but by reducing the duration of the disturbance the magnitude of the resulting transient in the system. The problem is (a) to design an inner loop (in the voter itself?) to adjust the remaining two outputs and (b) characterize the performance and reliability of the resulting system.

REFERENCES

1. Savant, C.J. Jr., Basic Feedback Control System Design, McGraw-Hill, 1958.

2. Dorf, R.C., Modern Control Systems, 3rd ed., Addison-Wesley, 1980.

3. Rowen, W.I. "Simplified Mathematical Representations of Heavy-Duty Gas Turbines", Transactions of the ASME, Journal of Engineering for Power, 1983.

4. Gilbert, K.E. "ECDL Digital Simulation of MS6001", unpublished engineering report of the General Electric Co., 1983.

5. Gilbert, K.E. and others, "HSAA-1H1C Circuit Analysis", unpublished engineering report of the General Electric Co., 1982.

6. Kuelz, E. "SAMP: A State Analysis Modeling Program", M. S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1970.

7. Brogan, W.L., Modern Control Theory, Quantum Publishers, 1974.

8. Digest of Papers for the 13th International Symposium on Fault-Tolerant Computing, Milan, Italy, June 28-30 1983, Institute of Electical and Electronics Engineers.

9. Digest of Papers for the 11th International Symposium on Fault-Tolerant Computing, Portland, Maine, June 24-26 1981, Institute of Electical and Electronics Engineers.

Appendix A

STATEVAR Simulation Program

```
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!                  STATEVAR
! SYSTEM SIMULATION USING STATE VARIABLE TECHNIQUES
! INCLUDES NON-LINEAR ELEMENT HANDLING
!
!
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!        REVISION HISTORY
!
! VERSION 1.0     16 JAN 81       DJ FELTNER
!         1.1     13 JAN 81       DJ FELTNER  ADD SAVE,UNSAVE;MODIFY EDIT
! VERSION 2.0     01 OCT 82       DJ FELTNER  MODIFY FOR VAX/VMS 3.0
! VERSION 3.0     13 OCT 82       DJ FELTNER  UNFORMATTED SAVE FILES
! VERSION 3.1     10-NOV-82       DJ FELTNER  REDUCE TO 15 S.V.'S, ADD
!                                             X(0) PLOT POINT
! VERSION 4.0     16-DEC-82       DJ FELTNER  ELIMINATE RECOMPUTE FOR
!                                                 MULTIPLE PLOTS
! VERSION 5.0     25-JAN-84       DJ FELTNER  CHANGE PLOT INTERVAL TO
!                                              ELIMINATE DEPENDANCE ON
!                                              CLAMP UPDATE RATE
!                                              (ENABLE INTERSAMPLE
!                                              RESPONSE PLOTTING)
! VERSION 5.1     08-FEB-84       DJ FELTNER  ADD EXTERNAL PLOTTER (HP)
!                                              OUTPUT FILE, MISC CHANGES
! VERSION 5.2     10-APR-84       DJ FELTNER  ADD SAVE OF FINAL VALUES,
!                                              UPPERCASE CONVERT OF INPUT
! VERSION 5.3     25-APR-84       DJ FELTNER  FIX ASSIGNED FUNCTION PRINT
! VERSION 5.4     30-APR-84       DJ FELTNER  FIX INITIAL PLOTTED VALUE
!
!
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!
!        DESCRIPTION OF OPERATION
!
!
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
! THIS PROGRAM USES THE STATE TRANSISITION MATRIX AND A ZERO-ORDER
! HOLD MATRIX TO CALCULATE THE STATE OF A SYSTEM AT ANY TIME GREATER
! THAN ZERO USING RECURSION EQUATIONS.
!
! FOR THE SYSTEM WITH THE STATE VECTOR X DESCRIBED BY THE EQUATION
!         X' = A * X         (HOMOGENEOUS EQUATION)
! THE PROGRAM CALCULATES A STATE TRANSITION MATRIX PHI(DELTA)
!         PHI(DELTA) = EXP(A*DELTA)
! USING A POWER SERIES WHICH TRUNCATES AT 1% ERROR.
!
! THE MATRIX B IS DEFINED AS THE MATRIX WHICH RELATES THE STATE OF THE
! SYSTEM AT T = N*DELTA TO THE STATE AT T = (N*DELTA)+ OR IMMEDIATELY
! FOLLOWING THE IMPULSE OF THE SAMPLER, WHERE N*DELTA IS THE TIME BETWEEN
! SAMPLES.  THE STATE OF THE SYSTEM IS 'UPDATED' BY
!         X((N*DELTA)+) = B * X(N*DELTA)
!
! THE STATE OF THE SYSTEM IS THEN EVALUATED BY
!         X((N+1)*DELTA) = PHI(DELTA) * X(N*DELTA)
!
! THE PROGRAM IS DESIGNED TO COMPUTE THE STATE OF THE SYSTEM AT EACH
! INCREMENT OF DELTA, UPDATE THE CLAMP AT EACH INCREMENT N*DELTA, AND
! PLOT THE OUTPUT AT EACH POINT M*DELTA TO A FILE DESIGNED TO PRINT
! ON A LINE PRINTER.  AN OPTIONAL PLOT FILE CONTAINS DATA POINTS
! DESIGNATED BY THE OPERATOR AND EXTRACTED AT FIVE TIMES THE LINE-PRINTER
```

```
! PLOT RATE.  THIS OPTIONAL PLOT FILE IS DESIGNED TO BE INPUT TO AN
! EXTERNAL PLOTTER PROGRAM, SUCH AS THE HP2647A 'AUTOPLOT'.
!
! THE PROGRAM IS DESIGNED TO HANDLE NON-LINEAR ELEMENTS BY DEFINING THEM
! AS PART OF THE CLAMP FUNCTION OF THE ZERO-ORDER HOLD BLOCK.  THIS
! METHOD, KNOWN LOCALLY AS THE 'CONRAD METHOD', OPERATES BY PREFIXING
! EACH NON-LINEAR ELEMENT BY A CLAMP, AND DEFINING A STATE VARIABLE
! AS THE OUTPUT OF THE NON-LINEAR BLOCK.  AFTER EACH CLAMP UPDATE,
! THE STATE OF EACH STATE VARIABLE WHICH REPRESENTS A NON-LINEARITY
! IS USED AS THE INPUT TO A FUNCTION SUB-PROGRAM WHICH PROVIDES THE
! VALUE TO UPDATE THAT STATE VARIABLE.  THIS METHOD AVOIDS RECOMPUTING
! PHI FOR EACH SAMPLING INTERVAL, AND REPLACES A TIME-CONSUMING
! MATRIX POWER SERIES EXPANSION WITH A SIMPLE LOOK-UP OR COMPUTATION.
!
! USING THIS METHOD, THE A-MATRIX IS WRITTEN AS USUAL WITH THE SAMPLER(S)
! OPEN AND THUS THE NON-LINEARITYS DO NOT APPEAR IN THAT MATRIX.
! THE B-MATRIX IS WRITTEN ASSUMING EACH NON-LINEAR BLOCK AS UNITY GAIN.
!
! THE BASIC PROGRAM FLOW IS:
!         INPUT VARIABLES
!               !
!         CALCULATE PHI(DELTA) = EXP(A*DELTA)
!               !
!         START RECURSION AT INTERVAL DELTA
!               !
!                 X(TIME +) = B * X(TIME)          !UPDATE THE CLAMP
!                 UPDATE NON-LINEAR STATE VARIABLES
!                 X(NEXT) = PHI * X(TIME+)         !CALC NEXT STATE
!                 IF (TIME MOD PRINT INTERVAL = 0) SAVE THIS VALUE
!                 EXITIF TIME > MAX
!               !
!         CALCULATE MAX OF ALL STATES
!         PLOT STATE VARIABLE
!
! THE NON-LINEARITES ARE HANDLED BY NAMING FIVE FUNCTION SUBPROGRAMS
! FUNC1 THRU FUNC5 WHICH ARE WRITTEN BY THE USER AND LINKED INTO
! THE MAIN PROGRAM.  AT RUN TIME THE USER IDENTIFIES THE STATE VARIABLES
! WHICH ARE TO BE ASSOCIATED WITH THE FUNCTIONS.  SINCE THE FUNCTIONS
! (AND THUS THE CLAMPS) MUST BE INCLUDED WHETHER THERE ARE NON-LINEARITIES
! OR NOT, IN SYSTEMS WITHOUT NON-LINEARITIES THE SAMPLING RATE
! MUST BE SET WELL ABOVE THE FASTEST SYSTEM TIME CONSTANT (10:1 MINIMUM)
! TO AVOID INFLUENCE DUE TO THE CLAMPS.  THIS CAN BE ACOMPLISHED WITH
! A CLAMP UPDATE MULTIPLIER OF 1 AND AN APPROPRIATE TIME INTERVAL DELTA
! THE FUNCTION SUBPROGRAMS ARE MERELY OF THE FORM:
!         REAL FUNCTION FUNC1(X)
!         REAL X
!         FUNC1 = X
!         RETURN
! THE PENALTY PAID IS RUN TIME.
!
! THE OPERATOR INTERFACE IS DESCRIBED IN THE FILE "STATEVAR.HLP", WHICH
! CONTAINS THE OPERTING INSTRUCTIONS FOR THE USER OF THE PROGRAM.
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!
!         VARIABLE DECLARATIONS
!
!
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  IMPLICIT NONE                    ! AVOID UNWANTED ALIASES
!
!
  PARAMETER       MSZ = 15         !DEFINE MAXIMUM ORDER OF THE SYSTEM
                                   !NOTE: WHEN THIS PARAMETER IS CHANGED
                                   !   THE CORRESPONDING PARAMETERS IN
                                   !   "SAVE" AND "UNSAVE" MUST BE CHANGED
                                   !   TO MATCH!!!!!!!!!!!!!!!!!!
!
!
  REAL    T                !TIME TAG FOR PLOTTED POINTS
```

```
      REAL    X(MSZ)              !STATE VARIABLE MATRIX
      REAL    XO(MSZ)             !INITIAL STATE VARIABLE MATRIX
      REAL    XTP(MSZ)            !CLAMP OUTPUT MATRIX
      REAL    A(MSZ,MSZ)          !COEFFICIENT MATRIX
      REAL    PHI(MSZ,MSZ)        !STATE TRANSITION MATRIX
      REAL    B(MSZ,MSZ)          !ZERO-ORDER HOLD MATRIX
      REAL    OUT(MSZ+1,0:999) !OUTPUT VARIABLE MATRIX (TIMETAG, VALUE(I))
      REAL    TEMP1(MSZ,MSZ)   !TEMPORARY MATRIX FOR EXPAT
      REAL    TEMP2(MSZ,MSZ)   !    "       "     "    "
      REAL    TEMP3(MSZ,MSZ)   !    "       "     "    "
      REAL    DELTA               !INCREMENT FOR PHI(DELTA)
      REAL    MIN                 !MINIMUM VALUE OF OUTPUT
      REAL    MAX                 !MAXIMUM VALUE OF OUTPUT
      REAL    FUNC1,FUNC2,FUNC3,FUNC4,FUNC5   !EXTERNAL NON-LINEAR PROCEDURES
!
      INTEGER*2       I,J,K,L !GENERAL PURPOSE INDEXES
      INTEGER*2       FLAG    !SUBROUTINE ERROR INDICATOR
      INTEGER*2       MAXSIZ  !MAXIMUM NUMBER OF ELEMENTS/DIMENSION (ORDER)
      INTEGER*2       NC      !READ STATEMENT CHARACTER COUNTER
      INTEGER*2       NV      !ORDER OF SYSTEM (<= MSZ)
      INTEGER*2       TI      !LUN FOR INPUT TERMINAL
      INTEGER*2       TO      !LUN FOR OUTPUT TO TERMINAL
      INTEGER*2       PF      !LUN FOR EXTERNAL PLOTTER FILE          !+R5.1
      INTEGER*2       CLAMP   !MULTIPLE OF DELTA TO UPDATE CLAMP  (>=1)
      INTEGER*2       PLOTI   !MULTIPLE OF DELTA TO PLOT AT (>=1)
      INTEGER*2       EPLOTI  !MULTIPLE OF DELTA FOR EXT PLOT (>=1)   !+R5.1
      INTEGER*2       SVOUT   !OUTPUT STATE VARIABLE SUBSCRIPT (1<=SVOUT<=NV)
      INTEGER*2       SVF(5)  !SUBSCRIPTS OF STATE VARIABLE FOR FUNC(I)
      INTEGER*2       NPTS    !TOTAL NUMBER OF POINT TO PLOT (<= 1000)
      INTEGER*2       PTCNT   !NUMBER OF POINTS PLOTTED
      INTEGER*2       MARK    !PROGRESS TAG FOR DIAGNOSTIC MESSAGES
      INTEGER*4       BIGI    !LARGE INTEGER
      INTEGER*2       PLOTV(5)!SV'S FOR EXTERNAL PLOTTER              !+R5.1
!
!
      BYTE    SVEFIL(35)          !SAVE FILE NAME
      BYTE    OUTFIL(35)          !OUTPUT FILE NAME
      BYTE    EPFFIL(35)          !EXTERNAL PLOTTER OUTPUT FILE           !+R5.1
      BYTE    TME(8)              !CURRENT TIME-OF-DAY
      BYTE    GTITLE(50)          !GRAPH TITLE
      BYTE    XTITLE(25)          !X-AXIS (TIME) TITLE
      BYTE    YTITLE(25)          !Y-AXIS (STATE VARIABLE) TITLE
      BYTE    TEST(1)             !CHARACTER TEST BYTE
      BYTE    TEMP(80)            !INPUT EDITING BUFFER
      BYTE    BEL(1)              !CNTL-G
      BYTE    SP(1)               !SPACE
!
!
      LOGICAL LOAD                !
      LOGICAL EXTPLT              !.TRUE. ==> ENABLE EXT PLOTFILE         !+R5.1
!
!
!           COMMON FOR SAVE/UNSAVE SUBROUTINES
      COMMON /DSAVE/MAXSIZ,NV,A,B,XO,DELTA,CLAMP,PLOTI,SVOUT,SVF,PLOTV,NPTS !R5.1
!
!           COMMON FOR EXTERNAL FUNCTION SUBROUTINES
      COMMON /STATEVECTOR/X                                       ! +R5.2
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!
!           FORMATS
!
!
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
1     FORMAT(Q:,80A1)
2     FORMAT(I6)                !DECODE   STATE VAR, CLAMP, DELTA, ETC
3     FORMAT(F12.6)             !DECODE DELTA, ETC.
4     FORMAT(' ENTER "Y" FOR INSTRUCTIONS, ELSE CR: ',$)      !HELP PROMPT
5     FORMAT(' NUMBER OF STATE VARIABLES [INTEGER]: ',$)
6     FORMAT(' ENTER NAME OF SAVE FILE: ',$)  !SAVE FILENAME PROMPT
```

```
7 FORMAT(' ALL EDITING MODES EXCEPT MIEDT MATRIX EDITOR:'/
  1'      INPUT OK OR NO INPUT: HIT CR ("RETURN")'/
  1'      CHANGE DISPLAYED VALUE: TYPE NEW VALUE, HIT CR'/
  1'      BACKUP: TYPE "<", CR'/
  1'      NEXT ITEM: TYPE ">", CR'/
  1'      PRINT CURRENT VALUE: TYPE "=", CR'/
  1'      COMMAND SUMMARY: TYPE CTRL-H, CR'/
  1'      TERMINATE PROGRAM: HIT CTRL-Z.'/
  1'      YES/NO QUESTIONS: TYPE "Y" OR "N", HIT CR'/
  1' MATRIX EDITOR (PROMPT "MIEDT>"):'/
  1'      END EDITOR: CTRL-Z'/
  1'      PRINT: CR, RESPOND "Y"'/
  1'      BACKUP: CR, CR, "Y"'/
  1' TRY OUT THE COMMANDS - YOU CAN FIX ALMOST ANYTHING.')
8 FORMAT(' B-MATRIX: INITIALLY IDENTITY MATRIX.  ENTER
  1 CHANGES NOW WITH MIEDT "I,J,B(I,J)" [INT,INT,REAL]')
9 FORMAT(' X0: INITIALLY ZERO. ENTER CHANGES WITH MIEDT "I,X0(I)"
  1 [INT,REAL]')
10        FORMAT(' SUBSCRIPT OF STATE VARIABLE TO PLOT [INT]:',$)
11        FORMAT(' FOR PHI(DELTA), ENTER DELTA IN SECONDS [REAL]:',$)
12        FORMAT(' ENTER MULTIPLE OF DELTA FOR SAMPLES (CLAMP) [INTEGER]
  1:',$)
13        FORMAT(' ENTER OUTPUT FILENAME: ',$)
14        FORMAT(' ENTER MULTIPLE OF DELTA FOR PLOT [INTEGER]:',$)        !V5.0
15        FORMAT(' E TO EDIT, CTRL-Z TO TERMINATE RUN')                   !R5.1
16        FORMAT(' ENTER SUBSCRIPT OF STATE VARIABLE CORRESPONDING
  1 TO USER FUNCTION')
17        FORMAT(' FUNC',I1,':',$)
18        FORMAT(' ENTER NUMBER OF POINTS TO PLOT [INTEGER]:',$)
19        FORMAT(' DO YOU HAVE NON-LINEARITIES? ',$)
20        FORMAT('  SVR - BEGIN PHI(DELTA)=EXP(A*DELTA) AT ',8A1)
21        FORMAT('  SVR - END PHI(DELTA) AT ',8A1/)
22        FORMAT('  SVR - BEGIN PLOT AT ',8A1)
23        FORMAT('  SVR - DONE AT ',8A1)
24        FORMAT('  * * * SVR - ERROR ',I2,' AT ',I2,', T = ',1PE10.3)
25        FORMAT('1','          CALCULATED USING THE FOLLOWING PARAMETERS:'/
  1'0TIME INCREMENTS IN SECONDS:'/
  2' DELTA = ',1PE10.3,
  3' CLAMP = ',1PE10.3,
  4' PLOT = ',1PE10.3)
26        FORMAT('0          A-MATRIX (COEFFICIENTS OF SYSTEM)')
27        FORMAT('0          B-MATRIX (COEFFICIENTS OF CLAMPS)')
28        FORMAT('0          X(0) (INITIAL CONDITIONS)')
29        FORMAT('0          PHI(DELTA) MATRIX')
30        FORMAT('+',1PE12.6,' >',$)
31        FORMAT('+',I4,' >',$)
32        FORMAT(' ENTER FILENAME TO SAVE DATA, ELSE CR: ',$)
33        FORMAT('0VARIABLES ASSOCIATED WITH NON-LINEARITIES:'/
  1'    ',5(' FUNC',I1,': ',I2))
34        FORMAT(' A-MATRIX: INITIALLY ZERO. CHANGE WITH MIEDT')
35        FORMAT(' RELOAD FROM SAVE FILE? [Y/N] ',$)
36        FORMAT('+ SVR - COMPUTING STATE ',I12,' AT ',8A1)
37        FORMAT(' NEXT PLOT VARIABLE: [INT] ',$)
38        FORMAT(' ',6(1PE10.3,'  '))                                     !+R5.1
39        FORMAT(' ENTER FILENAME OF EXT PLOT FILE, ELSE CR: ',$)         !+R5.1
40        FORMAT(' PLOT',I1,':',$)                                        !+R5.1
41        FORMAT('+',I4,' >',$)                                           !+R5.1
42        FORMAT('    TIME    ',5('    X(',I2,')   '))                    !+R5.1
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!         INITIALIZE THE VARIABLES
!
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  MAXSIZ = MSZ              !PARAMETER FOR CHANGING SIZE OF MAXIMUM MATRIX
                            !IT IS THE DECLARED NUMBER OF ELEMENTS PER
                            !DIMENSION FOR THE MATRICES IN THE PROGRAM.
                            !IT MUST BE ADJUSTED WHEN THE DECLARED SIZE
                            !CHANGES, AND VICE-VERSA.
                            !THIS IS A 'SYSGEN' TYPE PARAMETER - CHANGE
                            !ONLY WHEN CHANGING THE SIZE OF THE SYSTEM.
```

```
      TI = 5                    !TI IS THE LOGICAL UNIT NUMBER FOR THE
                                !INPUT TERMINAL.
      TO = 6                    !
      PF = 7                    !PLOTTER FILE                              !+R5.1
      MARK = 0                  !INITIAL LOCATION
      SP(1) = "40               !SPACE
      BEL(1) =  7               !CNTL-G
                          !MATRICES AND VECTORS
      DO 90 I=1,MSZ
      DO 91 J=1,MSZ
      A(I,J) = 0                !COEF MATRIX = ZERO
      B(I,J) = 0                !CLAMP MATRIX = IDENTITY
      IF (I.EQ.J) B(I,J)=1.
      PHI(I,J) = 0              !STATE TRANS MATRIX = ZERO
      TEMP1(I,J) = 0
      TEMP2(I,J) = 0
      TEMP3(I,J) = 0
91          CONTINUE
      X0(I) = 0                 !INITIAL CONDITIONS = ZERO
      XTP(I) = 0                !CLAMP OUTPUT = ZERO
      X(I) = 0                  !STATE VARIABLES = ZERO
90          CONTINUE
                          !OUTPUT ARRAY
      DO 92 I=0,999
      DO 92 J=1,MSZ+1
      OUT(J,I) = 0.
92          CONTINUE
!
!           GTITLE: BLANK
      DO 93 I=1,50
93          GTITLE(I)="40
!
!           XTITLE: 'TIME, SECONDS'
      XTITLE(1) = "124
      XTITLE(2) = "111
      XTITLE(3) = "115
      XTITLE(4) = "105
      XTITLE(5) = "054
      XTITLE(6) = "040
      XTITLE(7) = "123
      XTITLE(8) = "105
      XTITLE(9) = "103
      XTITLE(10) = "117
      XTITLE(11) = "116
      XTITLE(12) = "104
      XTITLE(13) = "123
      DO 94 I=14,15
94          XTITLE(I) = "040
!
!           YTITLE: 'OUTPUT VARIABLE X(   )'
      YTITLE(1) = "117
      YTITLE(2) = "125
      YTITLE(3) = "124
      YTITLE(4) = "120
      YTITLE(5) = "125
      YTITLE(6) = "124
      YTITLE(7) = "040
      YTITLE(8) = "130
      YTITLE(9) = "050
      YTITLE(10) = "040
      YTITLE(12) = "051
      DO 95 I = 13,25
95          YTITLE(I) = "040
      LOAD = .FALSE.
      EXTPLT = .FALSE.
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!
!           INPUT DATA FROM THE TERMINAL OR A SAVE FILE
!
```

```
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  WRITE(TI,*)' STATEVAR     VER 5.4        30 APR 84'
96        CONTINUE
  WRITE(TI,4)                   !HELP?
  READ(TI,1,END=999)NC
  IF (NC.EQ.0) GOTO 97
  OPEN(UNIT=2,NAME='[FELTNER.EXE]STATEVAR.HLP',TYPE='OLD',
  1 READONLY,ERR=97)
  CALL HELP(2,TI)                                  !LIST HELP FILE ON TERMINAL
  CLOSE(UNIT=2)
!
! * * * UNSAVE FEATURE  * * *
97        CONTINUE
  WRITE(TI,35)                  !RELOAD?
  READ(TI,1,END=999,ERR=97)NC,(TEMP(L),L=1,NC)
  IF ((NC.EQ.0).OR.((TEMP(1).NE.'Y').AND.(TEMP(1).NE.'y'))) GOTO 99 ! R5.2
98        CONTINUE                          !ELSE ENTER NAME OF SAVE FILE
  WRITE(TI,6)
  READ(TI,1,END=999)NC,SVEFIL
  IF(NC.EQ.0) GOTO 99
  SVEFIL(NC+1) = 0
  OPEN(UNIT=2,NAME=SVEFIL,TYPE='OLD',READONLY,ERR=98,FORM='UNFORMATTED',
  1 CARRIAGECONTROL='NONE',ACCESS='SEQUENTIAL',RECORDTYPE='VARIABLE')
  CALL UNSAVE(2,TI)                                !ENTER DATA FROM SAVE FILE
  CLOSE(UNIT=2)
  LOAD = .TRUE.
! * * * END UNSAVE * * *
!
!
99        WRITE(TI,7)                       !SUMMARY OF COMMANDS
  !
  !
  !               BEGIN DATA INPUT AND EDIT
100       WRITE(TI,5)                       !NUMBER OF STATE VARIABLES
  WRITE(TI,31)NV                !DISPLAY CURRENT VALUE
  READ(TI,1,END=999,ERR=100)NC,(TEMP(L),L=1,NC)
  CALL EDTR(TI,NC,TEMP,FLAG)
  GOTO (98,100,110),FLAG                !BACKUP,REPRINT,NEXT
  DECODE(NC,2,TEMP,ERR=100)NV           !ELSE GET NV
  IF ((NV.GE.1).AND.(NV.LE.MSZ)) GOTO 110
  WRITE(TI,*)'* * * MUST BE BETWEEN 1 AND ',MAXSIZ                  !R5.1
  GOTO 100
!
  !
  !               READ AND EDIT A-MATRIX
110       WRITE(TI,34)                      !INSTRUCTIONS FOR A-MATRIX
  WRITE(TI,15)                  !WAIT FOR CR OR EXIT
  READ(TI,1,END=999,ERR=110)NC,(TEMP(L),L=1,NC)                     !R5.1
  CALL EDTR(TI,NC,TEMP,FLAG)                                        !+R5.1
  GOTO (100,115,120),FLAG                                           !+R5.1
115       CALL MIEDIT(A,MSZ,NV,2,TI,FLAG)           !INPUT AND/OR EDIT A
  GOTO (110,100),FLAG                          !ERR = 110, BACKUP=100
  !
  !
  !               READ B-MATRIX
120       WRITE(TI,8)                       !INSTRUCTIONS FOR B-MATRIX
  WRITE(TI,15)                  !WAIT FOR CR OR EXIT
  READ(TI,1,END=999,ERR=120)NC,(TEMP(L),L=1,NC)                     !R5.1
  CALL EDTR(TI,NC,TEMP,FLAG)                                        !+R5.1
  GOTO (110,125,130),FLAG                                           !+R5.1
125       CALL MIEDIT(B,MSZ,NV,2,TI,FLAG)          !INPUT AND/OR EDIT B
  GOTO (120,110),FLAG                          !ERR = 120, BACKUP=110
  !
  !
  !               READ X0 MATRIX
130       WRITE(TI,9)
  WRITE(TI,15)                  !WAIT FOR CR OR EXIT
  READ(TI,1,END=999,ERR=130)NC,(TEMP(L),L=1,NC)                     !R5.1
  CALL EDTR(TI,NC,TEMP,FLAG)                                        !+R5.1
  GOTO (120,135,140),FLAG                                           !+R5.1
```

```
135       CALL MIEDIT(XO,MSZ,NV,1,TI,FLAG)        !INPUT AND/OR EDIT XO
  GOTO (130,120),FLAG                      !ERR=130, BACKUP=120
  !
  !   PRINT/VERIFY/MODIFY REST OF PARAMETERS
  !
140       CONTINUE          !GET SVOUT
  WRITE(TI,10)
  WRITE(TI,31)SVOUT
  READ(TI,1,END=999,ERR=140)NC,(TEMP(L),L=1,NC)
  CALL EDTR(TI,NC,TEMP,FLAG)
  GOTO (130,140,144),FLAG          !BACKUP,REPRINT,NEXT
  DECODE(NC,2,TEMP,ERR=140)SVOUT   !ELSE GET SVOUT
144       IF (SVOUT.LE.9) GOTO 145                !SKIP 'TENS' DIGIT
  YTITLE(10) = "060 + SVOUT/10             !CONVERT 'TENS' DIGIT TO ASCII
145       YTITLE(11) = "060 + MOD(SVOUT,10)       !   "    'ONES'   "    "   "
  IF ((SVOUT.GT.0).AND.(SVOUT.LE.NV)) GOTO 150
  WRITE(TI,*)' * * * REQUESTED VARIABLE NOT IN SYSTEM * * *'
  GOTO 140
  !
150       CONTINUE          !GET DELTA
  WRITE(TI,11)
  WRITE(TI,30)DELTA
  READ(TI,1,END=999,ERR=150)NC,(TEMP(L),L=1,NC)
  CALL EDTR(TI,NC,TEMP,FLAG)
  GOTO (140,150,160),FLAG          !BACKUP,REPRINT,NEXT
  DECODE(NC,3,TEMP,ERR=150)DELTA   !ELSE GET DELTA
  !
160       CONTINUE          !GET CLAMP
  WRITE(TI,12)
  WRITE(TI,31)CLAMP
  READ(TI,1,END=999,ERR=160)NC,(TEMP(L),L=1,NC)
  CALL EDTR(TI,NC,TEMP,FLAG)
  GOTO (150,160,170),FLAG          !BACKUP,REPRINT,NEXT
  DECODE(NC,2,TEMP,ERR=160)CLAMP   !ELSE GET CLAMP
  IF (CLAMP.GT.0) GOTO 170
  WRITE(TI,*)' * * * CLAMP MUST BE > ZERO * * *'
  GOTO 160
  !
170       CONTINUE          !GET PLOTI
  WRITE(TI,14)
  WRITE(TI,31)PLOTI
  READ(TI,1,END=999,ERR=170)NC,(TEMP(L),L=1,NC)
  CALL EDTR(TI,NC,TEMP,FLAG)
  GOTO (160,170,180),FLAG          !BACKUP,REPRINT,NEXT
  DECODE(NC,2,TEMP,ERR=170)PLOTI   !ELSE GET PLOTI
  IF (PLOTI.GT.0) GOTO 180
  WRITE(TI,*)' * * * PLOT MULTIPLE MUST BE > 0 * * *'
  GOTO 170
  !
180       CONTINUE          !GET NPTS
  EPLOTI = PLOTI/5                 !EXTERNAL PLOTTER INTERVAL      !+R5.1
  IF (EPLOTI.LT.1) EPLOTI = 1                                      !+R5.1
  WRITE(TI,18)
  WRITE(TI,31)NPTS
  READ(TI,1,END=999,ERR=140)NC,(TEMP(L),L=1,NC)
  CALL EDTR(TI,NC,TEMP,FLAG)
  GOTO (170,180,190),FLAG          !BACKUP,REPRINT,NEXT
  DECODE(NC,2,TEMP,ERR=180)NPTS    !ELSE GET NPTS
  IF ((NPTS.GT.0).AND.(NPTS.LE.1000)) GOTO 190
  WRITE(TI,*)' * * * MAXIMUM NUMBER OF POINTS = 1000 * * *'
  GOTO 180
  !
190       CONTINUE          !GET OUTPUT FILENAME
  WRITE(TI,13)
  READ(TI,1,END=999,ERR=190)NC,(OUTFIL(I),I=1,NC)
  IF(NC.EQ.0) GOTO 180
  OUTFIL(NC+1) = 0
  !       IF(INHSY(OUTFIL,NC+1,TI)) GOTO 190      !---V2.0---
  !
195       CONTINUE          !GET EXT PLOTTER FILENAME                  !+R5.1
```

```
  WRITE(TI,39)                                                          !+R5.1
  READ(TI,1,END=999,ERR=195)NC,(EPFFIL(I),I=1,NC)                       !+R5.1
  IF(NC.NE.0) EXTPLT = .TRUE.                                           !+R5.1
  EPFFIL(NC+1) = 0                                                      !+R5.1
  IF (.NOT. EXTPLT) GOTO 200                                            !+R5.1
  !
1951      CONTINUE          !ASSIGN PLOTTER OUTPUTS                                 !+R5.1
  DO 1959 I=1,5                       !MAX OF FIVE OUTPUTS              !+R5.1
  J=I                                                                   !+R5.1
1952      WRITE(TI,40)J                       !WHICH OUTPUT                       !+R5.1
  WRITE(TI,41)PLOTV(J)                !WHICH SV                         !+R5.1
  READ(TI,1,END=999,ERR=1952)NC,(TEMP(L),L=1,NC)  !READ ANSWER          !+R5.1
  CALL EDTR(TI,NC,TEMP,FLAG)          !DECIDE WHAT TO DO                !+R5.1
  GOTO (195,1952,1959),FLAG           !BACKUP,REPRINT,NEXT              !+R5.1
  DECODE(NC,2,TEMP,ERR=1952)PLOTV(J)        !ELSE GET PLOTV(J)          !+R5.1
! J=J-1                                                                 !+R5.1
! IF (J.EQ.0) GOTO 195                !BACK UP                          !+R5.1
! GOTO 1952                                                             !+R5.1
1959      CONTINUE                                                                !+R5.1
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!         CONFIGURE NON-LINEAR ELEMENTS BY LINKING
!         USER-DEFINED NON-LINEAR FUNCTIONS TO
!         CORRESPONDING STATE VARIABLES.
!
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
200       IF(LOAD) GOTO 210
  SVF(1)=1                    !DEFAULT ASSIGNMENT FOR LINEAR CASE
  SVF(2)=1                    !* * * NOTE: ALL FUNCTIONS MUST BE DEFINED
  SVF(3)=1                    !* * *       LINEAR IF NOT NEEDED! ! !
  SVF(4)=1
  SVF(5)=1
  !
210       CONTINUE          !ASK FOR NON-LINEARITIES
  WRITE(TI,19)
  READ(TI,1,END=999,ERR=210)NC,TEST
  IF ((NC.EQ.0).OR.((TEST(1).NE.'Y').AND.(TEST(1).NE.'y'))) GOTO 290 ! R5.2
  WRITE(TI,16)
  DO 230 I=1,5
  J=I
220       WRITE(TI,17)J
  WRITE(TI,31)SVF(J)
  READ(TI,1,END=999,ERR=220)NC,(TEMP(L),L=1,NC)
  CALL EDTR(TI,NC,TEMP,FLAG)
  GOTO (180,220,230),FLAG             !BACKUP,REPRINT,NEXT
  DECODE(NC,2,TEMP,ERR=220)SVF(J) !ELSE GET SVF(J)
! J=J-1                                                                 !-R5.1
! IF (J.EQ.0) GOTO 180                !BACK UP                          !-R5.1
! GOTO 220                                                              !-R5.1
230       CONTINUE
  !
  !
  !
  !                 SAVE THE DATA?
290       LOAD = .TRUE.
295       WRITE(TI,32)
  READ(TI,1,END=999,ERR=295)NC,(SVEFIL(I),I=1,NC)
  IF(NC.EQ.0) GOTO 300
  SVEFIL(NC+1) = 0
  !   IF(INHSY(SVEFIL,NC+1,TI)) GOTO 295  !SYSTEM DISC CHECK  !---V2.0---
  OPEN(UNIT=2,NAME=SVEFIL,TYPE='NEW',ERR=290,FORM='UNFORMATTED',
  1 CARRIAGECONTROL='NONE',ACCESS='SEQUENTIAL',RECORDTYPE='VARIABLE')
  CALL SAVE(2,TI)
  CLOSE(UNIT=2)
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!
!         BEGIN CALCULATING THE SYSTEM
!
!
```

```
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!                   FIND PHI(DELTA) = EXP(A*DELTA)
300       CONTINUE
  OPEN (UNIT=2,NAME=OUTFIL,TYPE='NEW')                 !OUTPUT FILE OPEN
  IF (EXTPLT) OPEN (UNIT=PF, NAME=EPFFIL, TYPE='NEW') !PLOT FILE  !+R5.1
  CALL TIME(TME)
  WRITE(TI,20)(TME(I),I=1,8)
!
310       CALL EXPAT(MSZ,A,PHI,DELTA,FLAG,TEMP1,TEMP2,TEMP3)
  MARK = 1
  IF(FLAG.NE.0) GOTO 1000
                                  !      1    RUN-TIME ERROR (UNDEFINED)
                                  !      2    FAILURE TO CONVERGE
!
!                   END CALCULATING PHI(DELTA)
320       CALL TIME(TME)
  WRITE(TI,21)(TME(I),I=1,8)
!
!
!                   FIND X(0+) = B*X0
400       CALL MMPY(B,MSZ,MSZ,X0,MSZ,1,XTP,FLAG)
  OUT(1,0) = 0.
  DO 410 J = 1,MSZ                                                !+R5.4
  OUT(J+1,0) = XTP(J)                                             !R5.4
410       CONTINUE                                                        !+R5.5
  IF (EXTPLT) WRITE(PF,42)(PLOTV(I),I=1,5)          !LABEL OUTPUT   !+R5.1
  IF (EXTPLT) WRITE(PF,38)T,(XTP(PLOTV(I)),I=1,5) !PLOT OUTPUT    !+R5.1
  MARK = 2
  IF(FLAG.NE.0) GOTO 1000          !==>ERROR OCCURED
!
!
!                   BEGIN COMPUTING X(T)
500       T=0.
  BIGI = 0                  !NUMBER OF STATE CALCULATIONS
  PTCNT = 0                 !NUMBER OF PLOTTED POINTS       +V5.0
!
510       CONTINUE          !BEGIN STATE CALCULATION
  T = T + DELTA
  CALL MMPY(PHI,MSZ,MSZ,XTP,MSZ,1,X,FLAG) !  X = PHI * XTP
  MARK = 3
  IF (FLAG.NE.0) GOTO 1000                !==>ERROR
  !
  CALL SMPY(1.,X,XTP,MSZ,1,FLAG)          !  XTP = LAST X(T)
  MARK = 4
  IF (FLAG.NE.0) GOTO 1000                !==>ERROR
  !
  !
  BIGI = BIGI + 1                         !COUNT NUMBER OF STATE CALCS
  IF (MOD(BIGI,100).NE.0) GOTO 520
  !       ELSE  DO TIME-TAG
  CALL TIME(TME)
  WRITE(TI,36,ERR=520)BIGI,(TME(I),I=1,8)
520       CONTINUE                                                !+V5.0
  !
  !                 END  X(T) LOOP AT INTERVAL DELTA
  !
  !
  IF (MOD(BIGI,CLAMP).NE.0) GOTO 530      !NOT READY TO CLAMP YET +V5.0
  !
  !                 ELSE UPDATE THE CLAMP AT INTERVAL DELTA*CLAMP
  CALL MMPY(B,MSZ,MSZ,XTP,MSZ,1,X,FLAG)   !  X(T) = B * LAST X(T)
  MARK = 5
  IF (FLAG.NE.0) GOTO 1000                !==>ERROR
  CALL SMPY(1.,X,XTP,MSZ,1,FLAG)          !  XTP = X(T)
  MARK = 6
  IF (FLAG.NE.0) GOTO 1000                !==>ERROR
D CALL MPRT(B,2,NV,MSZ,TI)
D CALL MPRT(XTP,1,NV,MSZ,TI)
D CALL MPRT(X,1,NV,MSZ,TI)
  !
```

```
!                       COMPUTE THE NON-LINEARITIES
!
XTP(SVF(1)) = FUNC1(X(SVF(1)))
XTP(SVF(2)) = FUNC2(X(SVF(2)))
XTP(SVF(3)) = FUNC3(X(SVF(3)))
XTP(SVF(4)) = FUNC4(X(SVF(4)))
XTP(SVF(5)) = FUNC5(X(SVF(5)))
!
!                       NOW XTP = FUNC(B * X(T)), WHERE FUNC IS THE
!                                 USER-DEFINED NON-LINEARITY
!               END CLAMP LOOP
530     CONTINUE                                                        !+V5.0
!
IF (EXTPLT.AND.(MOD(BIGI,EPLOTI).EQ.0))                                 !+R5.1
1       WRITE(PF,38)T,(XTP(PLOTV(I)),I=1,5)     !PLOT OUTPUT            !+R5.1
!
IF (MOD(BIGI,PLOTI).NE.0) GOTO 540      !NOT READY TO PLOT YET  +V5.0
!               ELSE SAVE THE DESIRED VARIABLE AT INTERVAL
PTCNT = PTCNT + 1       !ONE MORE PLOTTED POINT                 +V5.0
!                               DELTA*PLOTI     !  V5.0
OUT(1,PTCNT) = T        !TIME TAG                               V5.0
DO 535 J = 1,MSZ
OUT(J+1,PTCNT) = XTP(J) !VALUE                                  V5.0
535     CONTINUE
540     CONTINUE                                                        !+V5.0
IF (PTCNT.LT.NPTS) GOTO 510     !AND COMPUTE STATE AGAIN        +V5.0
!               ELSE
!
!               FINISHED COMPUTING THE SYSTEM
CALL TIME(TME)
WRITE(TI,22)(TME(I),I=1,8)
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!
!       COMPUTE MIN AND MAX, THEN PLOT THE RESULTS
!
!
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
800     MIN = +1.E+38
MAX = -1.E+38
!
DO 810 L = 0,NPTS
IF(OUT(SVOUT+1,L).GT.MAX) MAX = OUT(SVOUT+1,L)
IF(OUT(SVOUT+1,L).LT.MIN) MIN = OUT(SVOUT+1,L)
810     CONTINUE
!
!
!       PLOT
CALL PAXIS(2,50,GTITLE,25,XTITLE,25,YTITLE,MIN,MAX)
DO 900 I = 0,NPTS
CALL PLOT(OUT(1,I),OUT(SVOUT+1,I))
900     CONTINUE
!
WRITE(TI,37)                                    !PLOT ANOTHER?
READ(TI,1,END=910,ERR=900)NC,(TEMP(L),L=1,NC)
CALL EDTR(TI,NC,TEMP,FLAG)
GOTO (900,900,910)FLAG          !BACKUP, REPRINT, NEXT
DECODE(NC,2,TEMP,ERR=900)SVOUT  !ELSE GET SVOUT
C D     WRITE(TI,*)' SVOUT = ',SVOUT
YTITLE(10) = "20                        !ERASE OLD TEN'S DIGIT
IF (SVOUT.LE.9) GOTO 905                !SKIP 'TENS' DIGIT
YTITLE(10) = "060 + SVOUT/10            !CONVERT 'TENS' DIGIT TO ASCII
905     YTITLE(11) = "060 + MOD(SVOUT,10)       !   "    'ONES'   "    "   "
IF ((SVOUT.GT.0).AND.(SVOUT.LE.NV)) GOTO 800    !AND RE-PLOT
WRITE(TI,*)' * * * REQUESTED VARIABLE NOT IN SYSTEM * * *'
GOTO 900
!
910     CONTINUE
!               WRITE THE INPUT DATA TO THE OUTPUT FILE
!
```

```
  WRITE(2,25)DELTA,CLAMP*DELTA,PLOTI*DELTA                              !R5.1
  WRITE(2,26)                                   !A-MATRIX
  CALL MPRT(A,2,NV,MSZ,2)
  WRITE(2,27)                                   !B-MATRIX
  CALL MPRT(B,2,NV,MSZ,2)
  WRITE(2,28)                                   !X0-MATRIX
  CALL MPRT(X0,1,NV,MSZ,2)
  WRITE(2,29)                                   !PHI-MATRIX
  CALL MPRT(PHI,2,NV,MSZ,2)
  WRITE(2,33)((I,SVF(I)),I=1,5)                 !NON-LIN FUNCTS !R5.3
  CLOSE(UNIT=2)
  IF (EXTPLT) CLOSE(UNIT=PF)                                            !+R5.1
  !
  !       SAVE FINAL STATE                                              ! +R5.2
920       WRITE(TI,32)                                                          ! +R5.2
  READ(TI,1,END=999,ERR=920)NC,(SVEFIL(I),I=1,NC)                       ! +R5.2
  IF(NC.EQ.0) GOTO 930                                                  ! +R5.2
  CALL SMPY(1.,X,X0,MSZ,1,FLAG)             !  X0 = X(FINAL)        ! +R5.2
  MARK = 7                                                              ! +R5.2
  IF (FLAG.NE.0) GOTO 1000                  !==>ERROR                   ! +R5.2
  SVEFIL(NC+1) = 0                                                      ! +R5.2
  OPEN(UNIT=2,NAME=SVEFIL,TYPE='NEW',ERR=290,FORM='UNFORMATTED',        ! +R5.2
  1 CARRIAGECONTROL='NONE',ACCESS='SEQUENTIAL',RECORDTYPE='VARIABLE') ! +R5.2
  CALL SAVE(2,TI)                                                       ! +R5.2
  CLOSE(UNIT=2)                                                         ! +R5.2
930       CONTINUE                                                              ! +R5.2
  !
  !       FINISHED!
999       CALL TIME(TME)
  WRITE(TI,23)(TME(I),I=1,8)
  !
  !       CALL START(RAD50('...MCR'),250,1,I)     !---V2.0---
  STOP
1000      WRITE(TI,24)FLAG,MARK,T
  GOTO 999
  END
```

Appendix B

LIMIT Limit Function Subroutine

```
!
! LIMIT FUNCTION SUBROUTINES
!    FOR USE IN THE 'STATEVAR' PROGRAM
!
!  V1.0  24-APR-84 D.J. FELTNER
!
 REAL FUNCTION FUNC1(X)
 REAL X
 IF (ABS(X) .GT. 1.) THEN
   FUNC1 = SIGN(1.,X)
 ELSE
   FUNC1 = X
 END IF
 RETURN
 END
!
!
!
 REAL FUNCTION FUNC2(X)
 REAL X
 IF (ABS(X).GT.1.) THEN
   FUNC2 = SIGN(1.,X)
 ELSE
   FUNC2 = X
 END IF
 RETURN
 END
!
!
!
 REAL FUNCTION FUNC3(X)
 REAL X
 IF (ABS(X).GT.1.) THEN
   FUNC3 = SIGN(1.,X)
 ELSE
   FUNC3 = X
 END IF
 RETURN
 END
!
!
!
 REAL FUNCTION FUNC4(X)
 REAL X
 FUNC4 = X
 RETURN
 END
!
!
!
 REAL FUNCTION FUNC5(X)
 REAL X
 FUNC5 = X
 RETURN
 END
```

Appendix C

MVSVOTE Function Subroutine

This set of function subroutines includes the LIMIT functions as well as the MVSVOTE mid-value select function for the voter (FUNC5).

```
!
!    MID-VALUE SELECT VOTER
!       AND
! LIMIT FUNCTION SUBROUTINES
!     FOR USE IN THE 'STATEVAR' PROGRAM
!
!  V1.0  02-MAY-84 D.J. FELTNER
!
 REAL FUNCTION FUNC1(X)
 REAL X
 IF (ABS(X) .GT. 1.) THEN
   FUNC1 = SIGN(1.,X)
 ELSE
   FUNC1 = X
 END IF
 RETURN
 END
!
!
!
 REAL FUNCTION FUNC2(X)
 REAL X
 IF (ABS(X).GT.1.) THEN
   FUNC2 = SIGN(1.,X)
 ELSE
   FUNC2 = X
 END IF
 RETURN
 END
!
!
!
 REAL FUNCTION FUNC3(X)
 REAL X
 IF (ABS(X).GT.1.) THEN
   FUNC3 = SIGN(1.,X)
 ELSE
   FUNC3 = X
 END IF
 RETURN
 END
!
!
!
 REAL FUNCTION FUNC4(X)
 REAL X
 FUNC4 = X
 RETURN
 END
!
!
!
!     MID-VALUE SELECT FUNCTION
!
 REAL FUNCTION FUNC5(X)
 REAL X, SMALLEST, LARGEST
 REAL SV(15)
 COMMON /STATEVECTOR/SV
 IF ((SV(2).EQ.SV(4)).AND.(SV(4).EQ.SV(6))) THEN
    FUNC5 = SV(2)
 ELSE
    LARGEST = MAX(SV(2),MAX(SV(4),SV(6)))
```

```
    SMALLEST = MIN(SV(2),MIN(SV(4),SV(6)))
    IF ((SV(2).NE.LARGEST).AND.(SV(2).NE.SMALLEST)) FUNC5 = SV(2)
    IF ((SV(4).NE.LARGEST).AND.(SV(4).NE.SMALLEST)) FUNC5 = SV(4)
    IF ((SV(6).NE.LARGEST).AND.(SV(6).NE.SMALLEST)) FUNC5 = SV(6)
ENDIF
RETURN
END
```

VITA

I was born on September 25, 1951 in Alemeda, California where I lived for the next seven years. I lived the next fifteen years in Roanoke, Virginia and attended Andrew Lewis High School. My undergraduate education was at Georgia Institute of Technology where I majored in Electrical Engineering, receiving my Bachelor of Electrical Engineering degree in 1972.

After graduation I served Active Duty for Training with the U.S. Army Reserve as an officer in the Signal Corps to complete my ROTC obligation. In June 1973 I accepted a position as a Field Engineer with General Electric's Installation and Service Engineering Department. During the next five years I provided technical direction to customers for the installation, maintenance, and repair of industrial electronic equipment including low and medium voltage switchgear, AC and DC motors from 5 to 5000 Hp, adjustable voltage DC drives, AC inverters for both motor drive and uninterruptable power supply applications, and a variety of control systems for applications including steel, paper, plastics, synthetic fibers, and sewage pumping. In 1978 I transfered to the Drive Systems Department in Salem, Virginia as a Control Systems Engineer for Gas Turbine Controls, where I designed diagnostic software for microprocessor-based control systems and systems for engineering automation. During this assignment I entered the Advanced Course in Engineering program affiliated with Virginia Polytechnic Institute and State University with the goal of a Masters of Science in Electrical Engineering degree. In 1982 I took my present assignment as a System Design Engineer for Gas Turbine Control Systems, where I am presently part of a team designing a distributed data base information and control system for plant automation.