NASA Technical Memorandum 100007

An Analysis of Redundancy Management Algorithms for Asynchronous Fault Tolerant Control Systems

Gloria J. Davis

(NASA-TM-100007) AN ANALYSIS OF REDUNDANCY MANAGEMENT ALGORITHMS FOR ASYNCHRONOUS FAULT TOLERANT CONTROL SYSTEMS (NASA) 36 p
Avail: NTIS HC A03/MF A01 CSCL 01B

N87-29408

Unclas
G3/01 0103491

September 1987

National Aeronautics and
Space Administration

NASA Technical Memorandum 100007

An Analysis of Redundancy Management Algorithms for Asynchronous Fault Tolerant Control Systems

Gloria J. Davis, Ames Research Center, Moffett Field, California

September 1987

NASA
National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

SUMMARY

Redundancy management algorithms, commonly referred to as voters, are algorithms used in fault tolerant control systems to vote on incoming redundant data, isolate "bad" signals, and output a single "good" value. In a synchronous environment, this algorithm is a straightforward signal-to-signal comparison with relatively low complexity. The technology of asynchronous control systems, recently realized in the Ultrareliable Fault Tolerant Control System research program at NASA Ames Research Center, requires more complex algorithms for fault detection and signal selection. A variety of algorithms used for this process, a means of testing them, and their basic performance under a simulated environment of the ultrareliable fault-tolerant control system are presented.

INTRODUCTION

The concept of redundancy mandates the use of some algorithm that will examine all incoming signals, decide which, if any, are failed, and output a reasonable signal (see fig. 1). This algorithm is referred to as a Redundancy Data Manager or voter. Many control systems have incorporated it and, subsequently, some type of voter has evolved (refs. 1-7).

Because of the common clock in synchronous environments, redundant signals are examined simultaneously. A predetermined tolerance level allowing for noise and slight interference is used in comparing the redundant signals; and any value exceeding this limit is not considered for output selection. The choice of the midvalue of the remaining signals ensures that a valid signal is output. This voter is simple, fast, and reliable, yet becomes completely invalid when transported into the asynchronous environment because of the skew in the associated times of the redundant signals.

It is the technology of asynchronous redundancy which dictates the generation of voter algorithms whose designs require more thought than those in synchronous applications. Such is the case in the Ultrareliable Fault Tolerant Control System (UFTCS) project at NASA Ames Research Center (ref. 8), based on the previous work of Dunn and Meyer (refs. 9-11). Here, the voter is incorporated in the same environmental setup as in figure 1, but the tasks A(i) each have different times associated with them, which are controlled by their own independent clocks (see fig. 2). In this configuration, the voter receives data values on each of the four pipelines, each tagged with a time by its own clock. Since they come from independent elements, the values vary. They all may or may not be valid. Based on these values, the voter must determine which pipelines are still working and which, if any, are

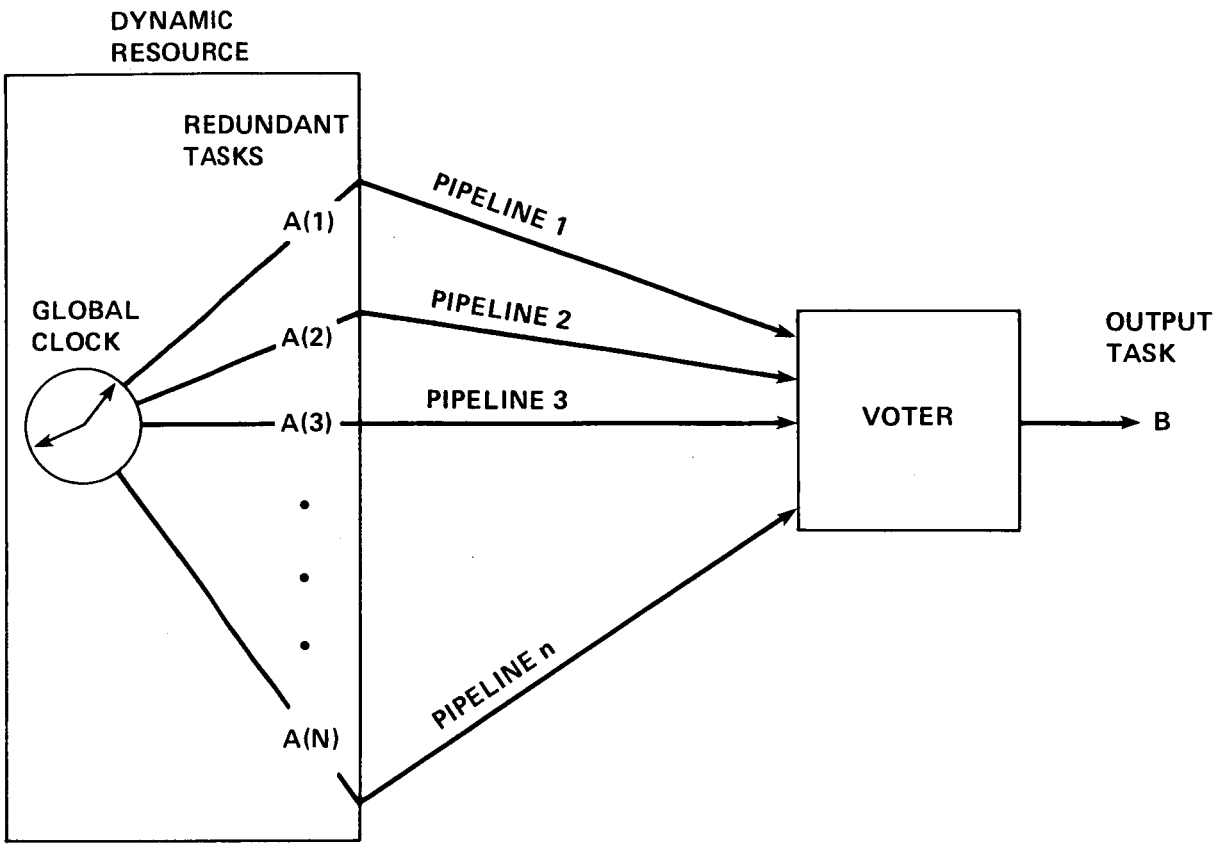


Figure 1.- N-modular redundancy system (synchronous).

not. It must also output a valid controlling value onto task B. This determination process may be by selection from those incoming data, by mathematical formula, or both. Nevertheless, it is the awareness of completely independent elements that dictates the use of a more complicated algorithm in this voter module. The voter must detect faults and generate valid signals based solely on the incoming asynchronous values. In the design of this algorithm one needs to consider the time differences of the values and still be reliable at detecting actual faults.

Many ideas were presented for this application, and a test was necessary to determine which of them would be reliable. A voter is reliable if it properly detects faults and outputs good data. It is not reliable if it: 1) fails to detect faults, 2) indicates faults on correctly working channels, 3) detects faults properly then later uses the faulted lines for output, and/or 4) outputs bad data. All of these possibilities must be noted and properly considered in voter design.

A series of experiments was performed on various voter algorithms to determine their effectiveness in both detecting faults and outputting valid signals in the UFTCS. The study was performed using a software simulation of the system and calculating statistics on the voters' performances. These experiments, the tested voters, and comparative analysis results are presented.

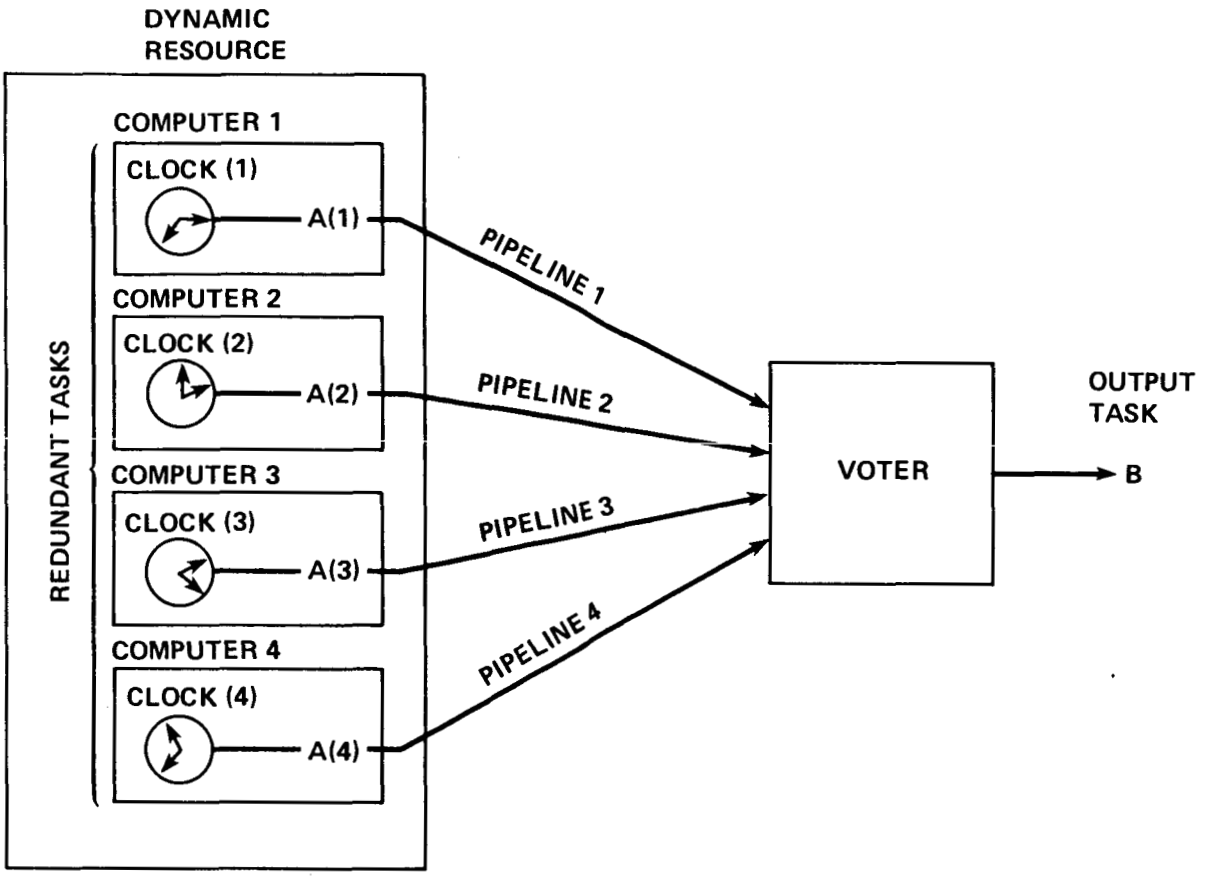


Figure 2.- Asynchronous redundant system.

ULTRARELIABLE FAULT TOLERANT CONTROL SYSTEM DESCRIPTION

The Ultrareliable Fault Tolerant Control System (UFTCS) concept using a systems design philosophy which allows development of system structures containing virtually no common elements was implemented at NASA Ames Research Center. This concept provided the means for removing common system elements by permitting them to operate as independent, uncoupled entities. Multiple versions of the application program are run on dissimilar hardware, thus adhering to the independence criterion. It is this concept of independent operation of redundant system elements that mandates the use of a voter.

A typical UFTCS was constructed to investigate the theoretical and practical bases of the concept with operational hardware and software. The control system of a UH-1H helicopter was implemented in this experimental testbed. This system has proven highly successful in both laboratory and manned simulation testing, with several hundred simulated flight hours logged to date. The environment of the voter is shown in figure 2, with a sample consisting of 4 redundant pipelines coming from 4 independent computers. Each of these pipelines consist of 4 different control

signals, one for each of the 4 axes of the helicopter. Out of these 16 input signals, the voter outputs 1 value for each of the 4 controls. The process is real-time with the voter receiving samples every 50 msec throughout the flight so as to control the helicopter's position.

This specific implementation of the UFTCS was a test aid to demonstrate proof of concept and should not be viewed as an exclusive application. Ultrareliable Fault Tolerant Control System is a general concept which has applications to any control system to enhance its reliability. Similarly, the voters also are of a generic specification.

VOTER CONCEPTS

There are two types of redundancy classifications: static and dynamic. Static redundancy utilizes those voters which detect faults, isolates them, and outputs valid signals. Dynamic redundancy voters, after fault detection, diagnose and correct the faults detected (ref. 12). The voters reported on in this paper are designed for static redundancy; therefore, they consist of two algorithms: fault detection/isolation and signal output. The internal purpose of fault detection/isolation algorithms is to provide a flag and record for system maintenance of both permanent and intermittent faults experienced during flight operations (ref. 1).

The signal output algorithm typically occurs after fault detection, thereby ensuring the consideration of only unfaulted signals as output. This part of the voter is usually referred to as signal selection. This title is characteristic of that in a synchronous environment, but is misleading when referring to its use in the asynchronous environment. There tends to be a wider variation between the input values because they are not synchronized. This not only affects the fault detection routine, but also adds new considerations to the "signal selection" routine. Because of this, the underlying method of the signal selection routine varies from voter to voter.

The voters used in these experiments were composed by engineers working on the UFTCS project at Ames Research Center. Some have been altered, others have been eliminated because of poor algorithm design or lack of efficiency. Those presented do not necessarily represent all good available algorithms, but these were designed for this specific system by people intimately familiar with the project.

The criteria that the voter designers had to observe were:

1. There are four sets of inputs called pipelines, each pipeline consisting of four different control values and their respective times (asynchronous)

2. Faults may be present on these inputs, taking on any form

3. Voters are required to detect faults and output good values for each control; and

4. The voter must complete execution within 50 msec.

It is this last criterion that restricts the complexity of the voter. Not only this, but this processing time (referred to as the sampling rate), can add to or detract from the overall reliability of the voter and, hence, the entire system (ref. 8). With these in mind, the following voters were submitted:

1. Midvalue Select (MIDVAL)

2. Residual Voter (RESID)

3. First Order Extrapolator (FRSORD)

4. Second Order Extrapolator (SECORD)

5. Third Order Extrapolator (TRDORD)

The midvalue select and residual voters rely on their fault detection schemes to correctly detect and isolate all faults that are present. They then choose the output from the remaining good values. The remainder of the voters in this paper rely to an extent on the fault detection scheme, but go one step further to ensure good output. For each control, they do not simply output the current value from an unfaulted pipeline, but rather use an unfaulted pipeline and extrapolate to the end of the frame and output this new value. This idea was incorporated to reduce inherent noise on the pipeline and, more specifically, to output a good value for the present time, thereby compensating for the loss of time due to sampling and voter processing.

The voter descriptions follow.

Midvalue Select

The midvalue select algorithm is commonly used in voter design. It processes data quickly and has been used extensively in synchronous environments (refs. 5 and 12). After executing the fault detection routine, MIDVAL essentially outputs the middle value of the controls, hence the title. This ensures the least amount of "hopping" around from extremes, thus outputting a somewhat continuous flow of data, relatively close together.

The basic idea of the fault detection scheme is to first check for any faults or bad data, and then to isolate those detected from being the actual control value output. Faults are detected by three different tests--two timing checks and one specific to the actual input values. To check for timing-related faults, the voter looks at those pipelines whose associated time is within plus or minus four frames (0.2 sec) of the previous output frame. These are not faulted. But those whose time is greater than 0.2 sec ahead or behind are considered failed and are flagged as such. The other, more simple, timing check fails a pipeline if its time has not changed within 20 frames (approximately 1 sec).

To detect faults in the input signals, differences between pipelines are summed and averaged. This is done so as to compare each value to all the other values and see its relative "position." It is this value that is then checked against a predetermined tolerance range of 4% of the signal amplitude for the specific control value. So as not to fail a control in a pipeline prematurely, the calculated average must be out of the tolerance range for 20 frames before it is failed. (This number was arrived at through independent testing and has demonstrated itself to the designer as optimal.)

In this voter, previously failed pipelines can be re-enabled. If the voter had failed because of its time, once the associated time comes back within ±4 frame times of the current selected pipeline's time, its failed flag is changed back to unfailed. To re-enable a pipeline that was failed because its relative average was not within the tolerance level of the other working pipelines, it must come back within that tolerance and remain there for 40 contiguous frames (2 sec).

After the fault detection routine, those pipelines that remain unfaulted are considered for signal selection. Midvalue select will continue to work until two failures have been detected; it is then recommended that the system be maintained since the detection of a third failure is impossible using this scheme. With this in mind, there are three different possible situations for signal selection: that in which all four lines are good; when one line is failed and three are good; and finally, when two failures have been detected and two lines remain good.

When all four lines are unfaulted, MIDVAL disregards the oldest (least recent) pipeline and chooses the middle value of the remaining three. For three working pipelines, the middle value of the working lines is selected. In the situation of two failures, the most recent of the remaining two is selected. It is when these two remaining pipelines drift from one another that all confidence in system reliability is lost (ref. 8).

Residual Voter

Fault detection in the residual voter, RESID, is based on the Taylor series second-order extrapolation. Each pipeline is extrapolated to the time of each of the other pipelines and their differences are compared with a user supplied error bound. This tolerance level was arrived at by experiments performed on the UFTCS. This technique was chosen so as to put the pipeline values at a common time to each other so their faults would become obvious. Each time a difference is noted to be greater than the tolerance level, the corresponding flag is set and, if the differences between one pipeline and all the others exceed the tolerance level, then that pipeline is indicated as failed and is ignored for that specific frame. This voter monitors all pipelines every frame and sets the failure flag accordingly each frame; it does not necessarily permanently fail a pipeline.

After the fault detection scheme, the voter chooses the unfailed pipeline with the most recent time associated with it as output.

First Order Extrapolation

The first order extrapolator voter, FRSORD, bases both fault detection and signal output routines on the calculated slope of a signal. By extrapolating all unfailed pipelines to a common time, FRSORD stores these data from the previous frame in a one frame history table and uses it in the calculation of the current slope. The differences between the pipelines are compared to a previously set tolerance and, when the differences between one pipeline and the other unfaulted lines differ by more than the tolerance level, that pipeline is failed. Even though a pipeline is failed, the voter continues to monitor and process the pipeline every frame in the event that it returns to a good status. When this happens, the voter removes the failed flag and incorporates this now-good channel in its calculations. Another quick check performed to detect a time-related fault is to see if the value of the signals for all four sensors are equal or if the time has not been updated in a series of frames. These two quick checks also result in detected isolated failures that may later allow a pipeline to be resurrected.

The output from this voter is the most recent unfailed pipeline extrapolated to the end of the frame (present time plus 50 msec). This line was chosen because less error is generally introduced when extrapolating shorter rather than longer distances.

Second Order Extrapolation

This voter, based on second-order extrapolation (SECORD) techniques, uses the same logic as FRSORD but keeps a history table of the two previous frames. Second order extrapolation calculates both the first and second derivatives and compares the pipeline differences with a previously set tolerance (like RESID). Pipelines are considered failed if the difference between the second-order extrapolation value of a control is greater than the tolerance level with the other working lines. This routine does not fail a pipeline and isolate it for the remainder of the run, but rather continually monitors each pipeline and sets the failed flag true or false each frame.

At the end of the routine, the most recent unfailed pipeline will be selected and the output is this line's extrapolated value to the end of the frame. The equations used to calculate for failures and output are as follows.

$$\text{MAYBFAIL}(i,j) = \text{SGNL}(i,1) - \text{SGNL}(j,1) - \text{SLOP}(j,1)*dt - \frac{1}{2}*\text{ACEL}(j,1)*dtsq$$

$$\text{OUTPUT}(1) = \text{SGNL}(\text{LATEST},1) + \text{SLOP}(\text{LATEST},1)*dt + \frac{1}{2}*\text{ACEL}(\text{LATEST},1)*dtsq$$

To be failed, MAYBFAIL is compared to the tolerance level and the fault flag is set appropriately.

Third Order Extrapolator

The third-order extrapolating voter executes the same logic as FRSORD and SECORD and uses the Taylor series expansion polynomial to the third order for its calculations. TRDORD keeps a history table of three previous signal values and their associated times: two previous slopes and the previous acceleration. Then with the new signal calculates the most recent slope, acceleration, and jerk, then uses these in the third-order extrapolation calculation. The equations are:

$$\text{MAYBFAIL}(i,j) = \text{SGNL}(i,1) - \text{SGNL}(j,1) - \text{SLOP}(j,1)*dt - \frac{1}{2}*\text{ACEL}(j,1)*dtsq - \frac{1}{6}*\text{JERK}(j,1)*dtcube$$

$$\text{OUTPUT}(1) = \text{SGNL}(\text{LATEST},1) + \text{SLOP}(\text{LATEST},1)*dt + \frac{1}{2}*\text{ACEL}(\text{LATEST},1)*dtsq + \frac{1}{6}*\text{JERK}(\text{LATEST},1)*dtcube$$

As in SECORD, the results of TRDORD's calculations determine failures and output for this voter.

TEST DESCRIPTION

Voter Simulator

Exhaustive experimentation on the voting algorithms was not feasible in the UFTCS hardware because it would have been too expensive in time and it could not provide a quantitative means for comparing their performances. A software program was designed to simulate the UFTCS environment for the voters and provide a means for analysis. This program was written in Fortran and implemented on the Intel Series III microcomputer development workstation.

The program (also referred to as the simulator) simulates the voter configuration shown in figure 2. It included four separate pipelines, each containing the values and associated times for four simulated control functions, any of which could be faulted. The voter frame time is set at 50 msec. Each test continues with faults randomly interjected until three of the four pipelines actually are faulted or the voter indicates that three faults have occurred.

The four control functions that were simulated are shown in figure 3. These functions were selected as being representative of the types of signals that a voter would act upon but do not represent actual control signals. For the tests, the functions shown in figure 3 were randomly phased.

A sample of the simulation input and output parameters that were recorded at each frame is contained in table 1. For this example, a constant offset fault (to be discussed later) has been inserted on pipeline 2 and the SECORD voter is being

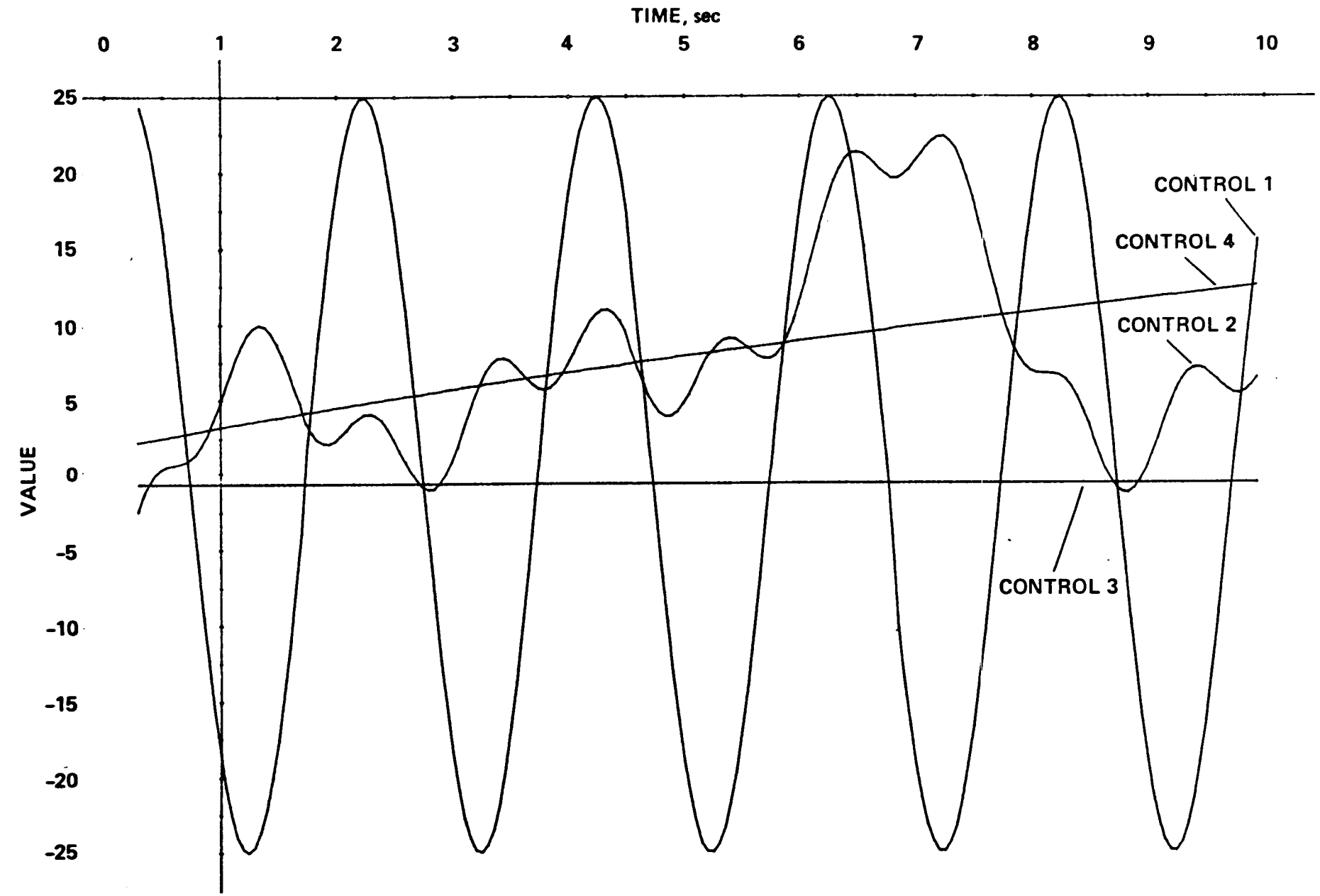


Figure 3.- VOTER input control functions.

TABLE 1.- SAMPLE FRAME OF SIMULATOR INPUT/OUTPUT PARAMETERS

INPUT TO THE VOTER

	PIPELINE 1	PIPELINE 2	PIPELINE 3	PIPELINE 4
TIME	2.900	2.914	2.913	2.891
CONTROL 1	-13.125	-24.863	-13.427	-12.986
CONTROL 2	-0.732	-9.234	-0.631	-0.751
CONTROL 3	-0.752	-11.972	-0.752	-0.752
CONTROL 4	5.721	-6.278	5.756	5.709

(a)

SECORD VOTER OUTPUT

TIME	2.963
CONTROL 1	-15.251
CONTROL 2	-0.329
CONTROL 3	-0.752
CONTROL 4	5.906

INDICATED FAULTS

	PIPELINE 1	PIPELINE 2	PIPELINE 3	PIPELINE 4
CONTROL 1	NO	YES	NO	NO
CONTROL 2	NO	YES	NO	NO
CONTROL 3	NO	YES	NO	NO
CONTROL 4	NO	YES	NO	NO

(b)

SIMULATOR OUTPUT/TRUE VALUES AND STATUS

TIME	2.963
CONTROL 1	-15.236
CONTROL 2	-0.337
CONTROL 3	-0.752
CONTROL 4	5.904

PIPELINE / FAULT TYPE	1	2	3	4
S-A-0	NO	NO	NO	NO
RAND. INP.	NO	NO	NO	NO
CON. DRIFT	NO	NO	NO	NO
CON. OFFSET	NO	YES	NO	NO
TRANSIENT	NO	NO	NO	NO
S-A-X	NO	NO	NO	NO

(c)

tested. The input parameters (table 1a) include the time and associated four control values for each of the pipelines. Table 1b shows the voter outputs and computed time as well as a summary of indicated faults. The actual or true values and faults are shown in table 1c.

Test Variables

To thoroughly test each of the voters' performances in the presence of faults, six specific fault characteristics were generated and implemented. They represent all types of faults: permanent, which are those invariant with time; and transient, those that can vary with time (ref. 10). The faults are:

1. Stuck at zero

2. Random replacement of control function with control values

3. Constant drift

4. Constant offset

5. Transient impulse

6. Stuck at last value, with step

Fault type 5 is referred to as a "soft" fault and typically occurs as a sudden short spike in the output. Since it is just a timewise short spike and does not remain in the signal, it should be isolated for the duration of the frame it occurs and not permanently failed. The other five faults are "hard" faults and should be permanently isolated until appropriate maintenance has been performed.

Tests were performed in a systematic fashion on all of the previously described voters to determine both their ability to detect faults and to output good values. These experiments were:

1. Noiseless input, no faults injected

2. Input with various levels of noise, no faults injected

3. Noiseless input, each fault tested separately and randomly introduced in the pipelines

4. Input with various levels of noise, each fault tested separately and randomly introduced in the pipelines

5. Input with noise and all faults allowed to occur

The test results are based on identical experiments run on each voter. The statistics calculated were based on an average of 10 runs each. Each run was limited to 10 sec. The average run lasted 3 sec because the system becomes inoperable upon identification of three faults.

Data Analysis

The statistical information recorded is for diagnostic and analytic purposes. Generated from each test are:

1. Average mean square error (AMSE)

2. Fault detection ratio (FDR)

3. False alarm ratio (FAR)

4. Bad resurrection ratio (BRR)

The average mean square error (AMSE) is relative to the true, noiseless value of the input function. The AMSE should be as close to zero as possible. For purpose of comparison, the AMSE is normalized to the maximum error of control function 2. The fault detection ratio (FDR) is the ratio of the number of faults detected to the number of faults injected. A value of one would be perfect for the FDR. The false alarm ratio (FAR) is the ratio of the number of wrongfully indicated faulted frames to the number of actually unfaulted frames. The FAR should be zero. A bad resurrection is when the voter properly detects a fault on a pipeline then later indicates that the fault is gone from the still-faulted pipeline. The bad resurrection ratio (BRR) is the number of frames a voter falsely resurrects a pipeline divided by the number of frames that pipeline actually is failed.

The following analysis of the voters' performances is based on these generated statistics, by analyzing them independently, and by comparison with each other.

RESULTS AND DISCUSSION

Because of the extent of the test data collected on all four control functions and the resulting lengthy analysis, the following discussion refers only to control signal two. This control function was similar to that of the UFTCS environment and the voters' performance on it was the most interesting.

Noiseless Input/No Faults

For the noiseless input/no-fault case, the AMSE should ideally be zero. Those voters that selected an input channel as output did best (as one would expect) because the clean input was simply passed through. The extrapolating voters, however, were not as close to zero, but still predicted quite well with AMSE between 0.02 and 0.201 on a wave form with an amplitude of 25.0 to 50.0 (see figs. 4 and 5). The differences between FRSORD and SECORD were 0.1 in AMSE, but going from SECORD to TRDORD gained accuracy only in the fourth decimal place of the AMSE which is not worth the cost in computation time. In summary, all of the voters achieved adequate accuracy in the no noise case.

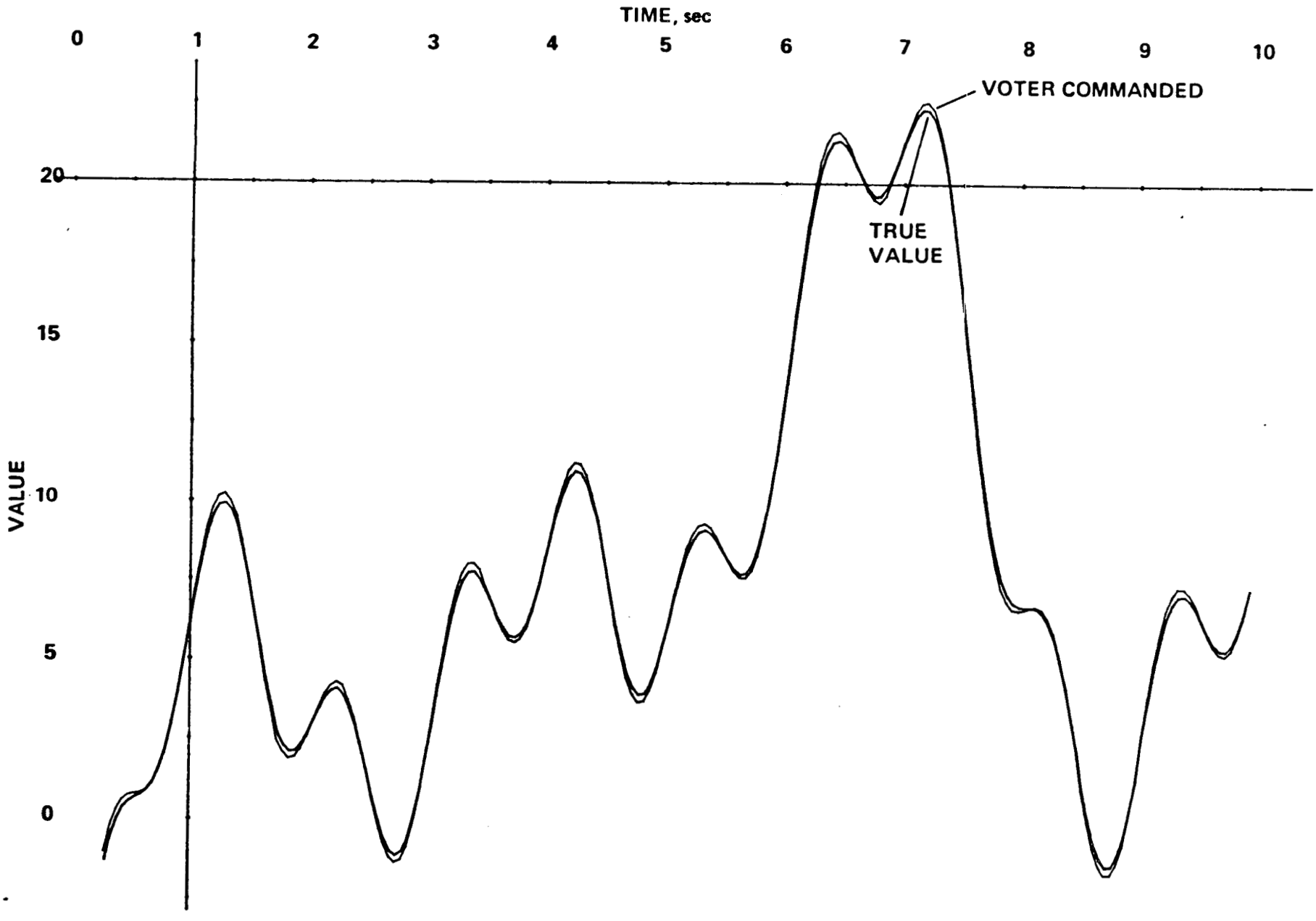


Figure 4.- FRSORD on no-noise, no-faults input.

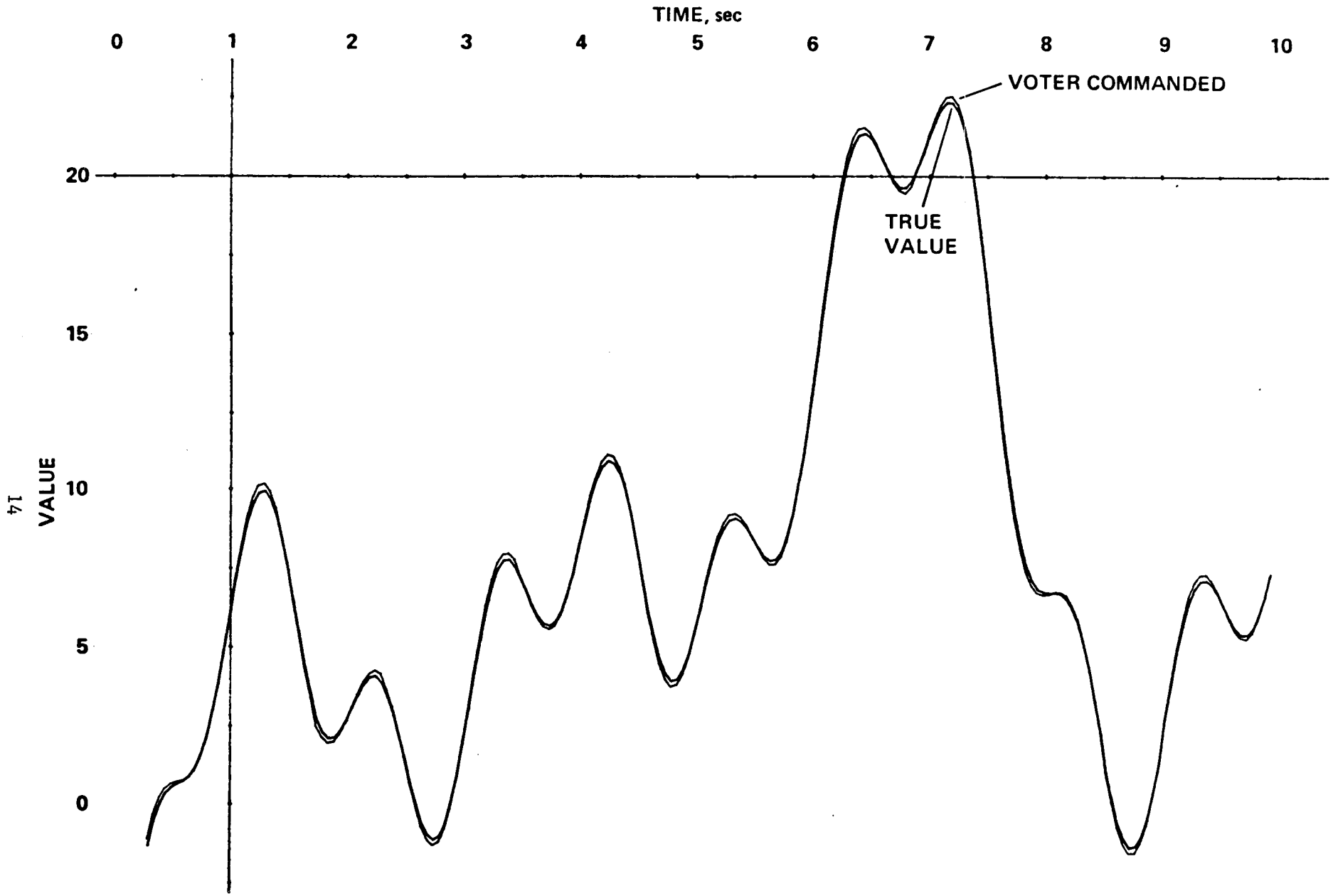


Figure 5.- No-noise, no-faults input.

Noise Test

The next set of tests consisted of a series of runs with still no faults injected but with various degrees of noise on the input signals. These tests were primarily conducted to determine the effectiveness of the different voters in selecting the signal or appropriate control value to pass on to the next task. The noise was randomly injected at levels of 1%, 2.5%, 5%, and 10% of the peak signal value.

Results indicate that the MIDVAL and RESID voters perform substantially better than the extrapolating voters. This is illustrated in figures 6 and 7 which contain results for the SECORD and MIDVAL voters, respectively. The large errors evident with the SECORD voter are all due to the voter's attempt to extrapolate a noisy signal beyond its correlation time. The signal selection difference between the voters may not be as significant in the presence of highly correlated noise (relative to the frame time).

Fault Test

The remainder of the experiments were performed to determine the voters' performances in the presence of various faults. For the sake of brevity, the results presented are for environments with up to 1% noise. The AMSE for each of the tests in the 1% noise environment is given in table 2.

Stuck at zero- The next test was to determine the voters' ability to detect the stuck at zero fault. This common fault typically occurs when power ceases in a component resulting in a dead or flat signal.

Based on the AMSE, all of the voters provided good controlling values in the event of this fault, with the worst being TRDORD which had a normalized error of 0.02. They all achieved a fault detection rate of at least 95%, false alarm rates of less than 5%, and a bad resurrection ratio of 0% (see table 3).

Random input- The random-input fault (replacing control function values with random values) yielded interesting results. In the presence of noise 5% and above, this fault could not reliably be detected because the noise and this particular fault are of the same characteristics. However, in the no- and low-noise environment, RESID and the extrapolating voters performed well, reliably detecting the faults with 100% FDR and FAR of at most 5% (see table 4). Their reliable fault detection resulted in a good output of controlling values (see fig. 8). MIDVAL failed to reliably detect the faults because of its lenient fault detection scheme therefore allowing this voter to output a failed pipeline as the control values. This resulted in a large AMSE.

Constant drift- One of the most difficult faults to detect is when one pipeline drifts from the others. This constant drift is hardest to detect when only two pipelines are working because they are each drifting away with respect to each other. In figure 9, all four pipelines are relatively the same until some time

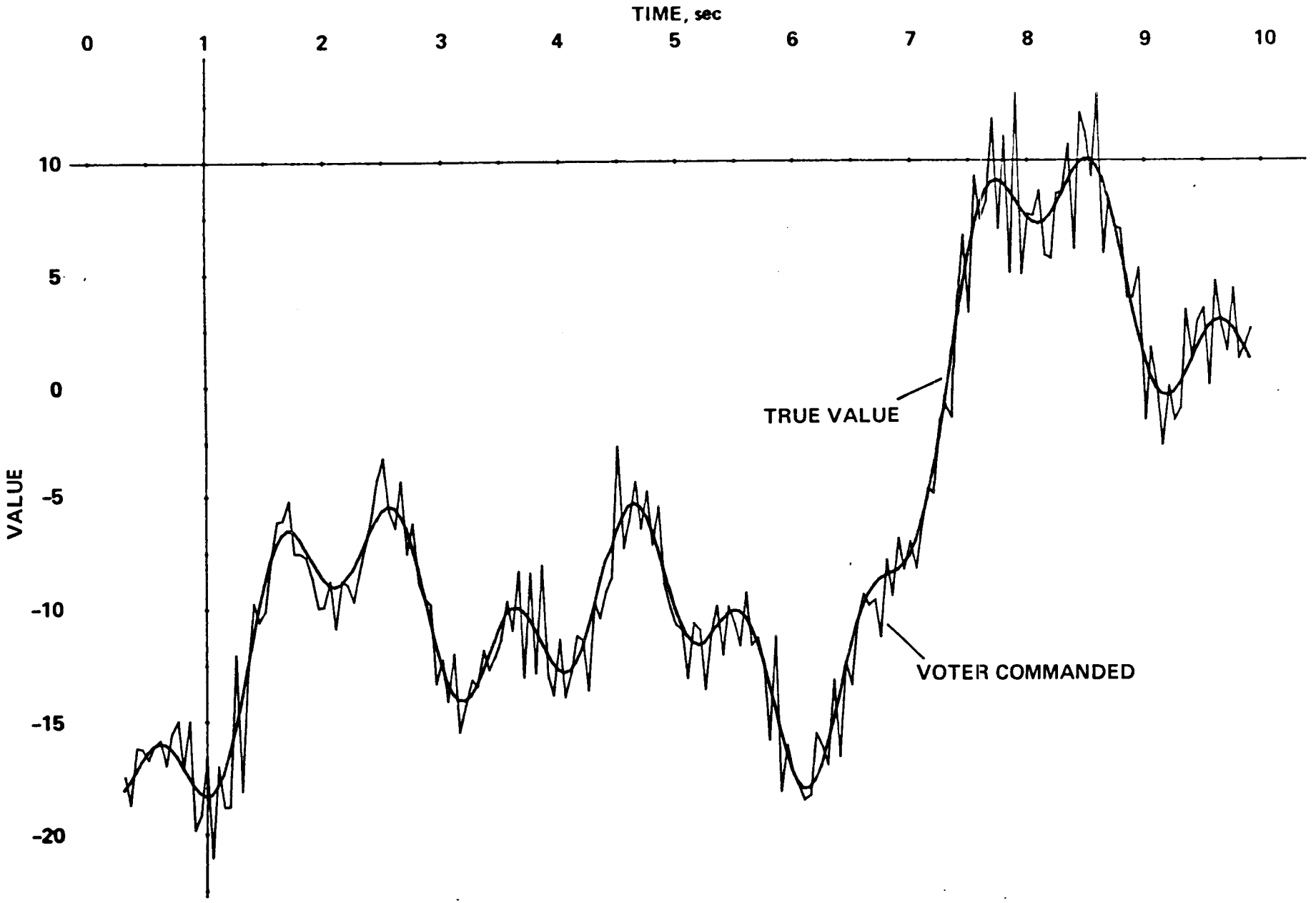


Figure 6.- SECORD in 2.5% noise test.

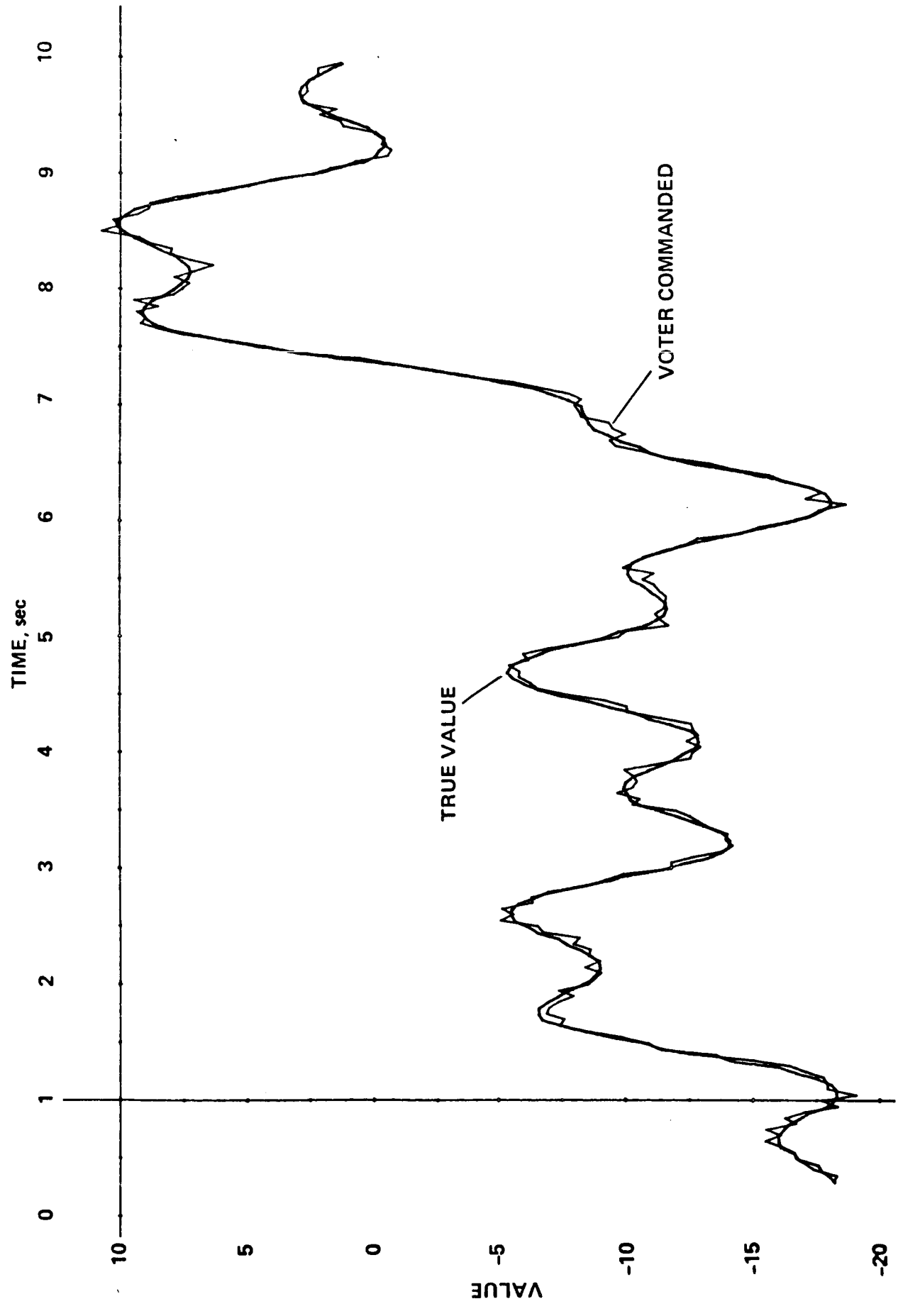


Figure 7.- MIDVAL in 2.5% noise test.

TABLE 2.- NORMALIZED AMSE IN 1% NOISE ENVIRONMENT

CONDITION \ VOTER	MIDVAL	RESID	FRSORD	SECORD	TRDORD
NO FAULTS	0.00036	0.00058	0.00316	0.00440	0.00465
STUCK-AT-ZERO	0.00221	0.00298	0.01230	0.01809	0.02047
RANDOM INPUT	0.2016	0.0142	0.0114	0.0156	0.0195
CONSTANT DRIFT	0.0071	0.0188	0.0132	0.0206	0.0208
CONSTANT OFFSET	0.0285	0.0281	0.0281	0.0652	0.0809
TRANSIENT IMPULSE	0.00163	0.00615	0.0126	0.0183	0.0198
STUCK-AT-LAST	0.00262	0.12368	0.0153	0.0195	0.0211
OVERALL	0.1759	0.1312	0.0286	0.0404	0.0415

TABLE 3.- STUCK AT ZERO FAULT WITH 1% NOISE

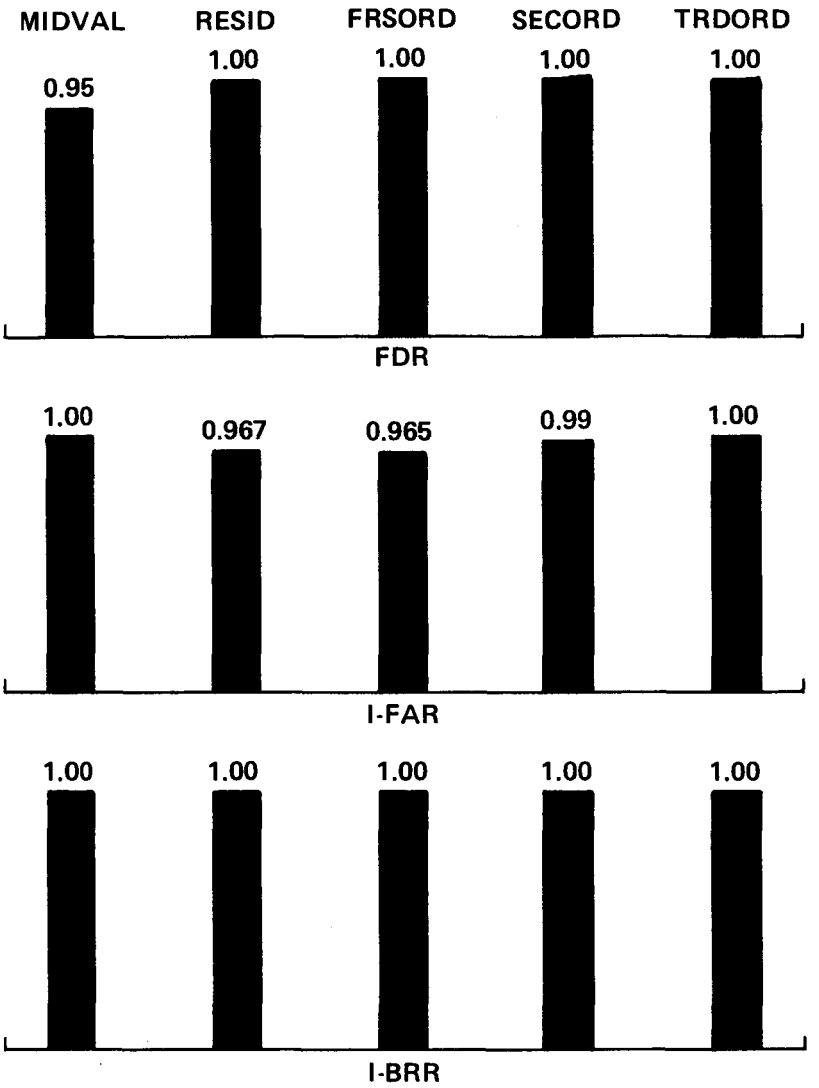
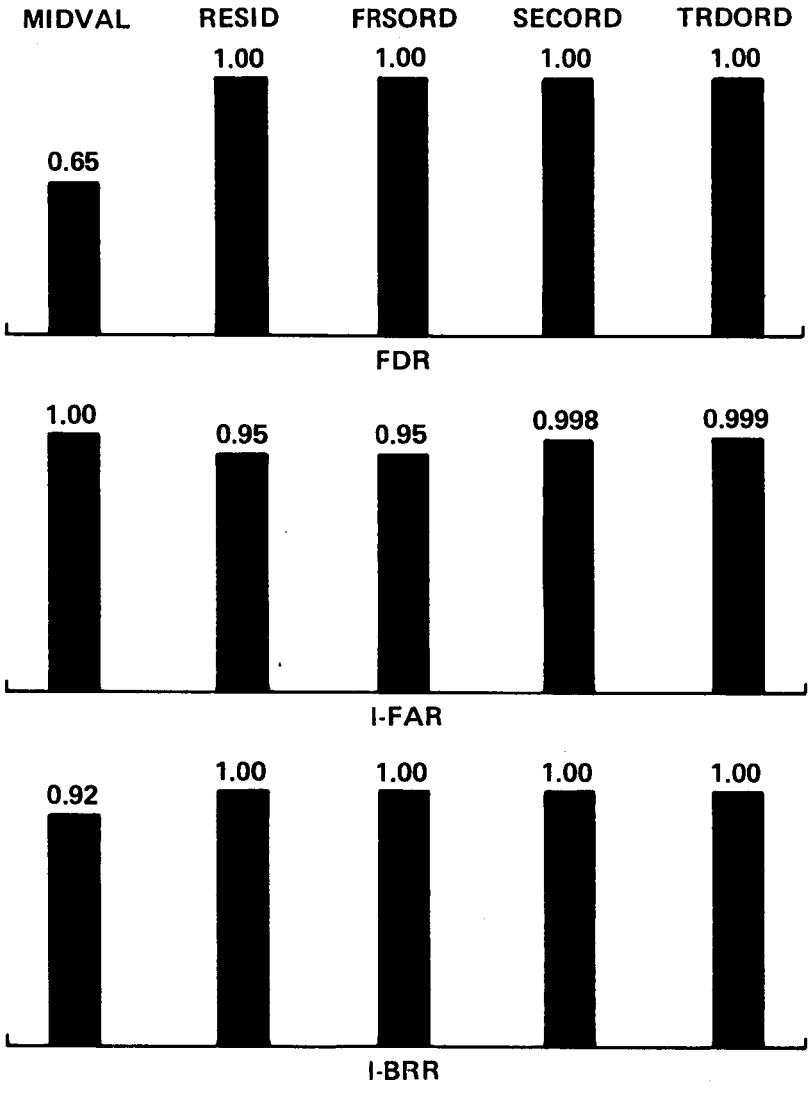


TABLE 4.- RANDOM INPUT FAULT WITH 1% NOISE



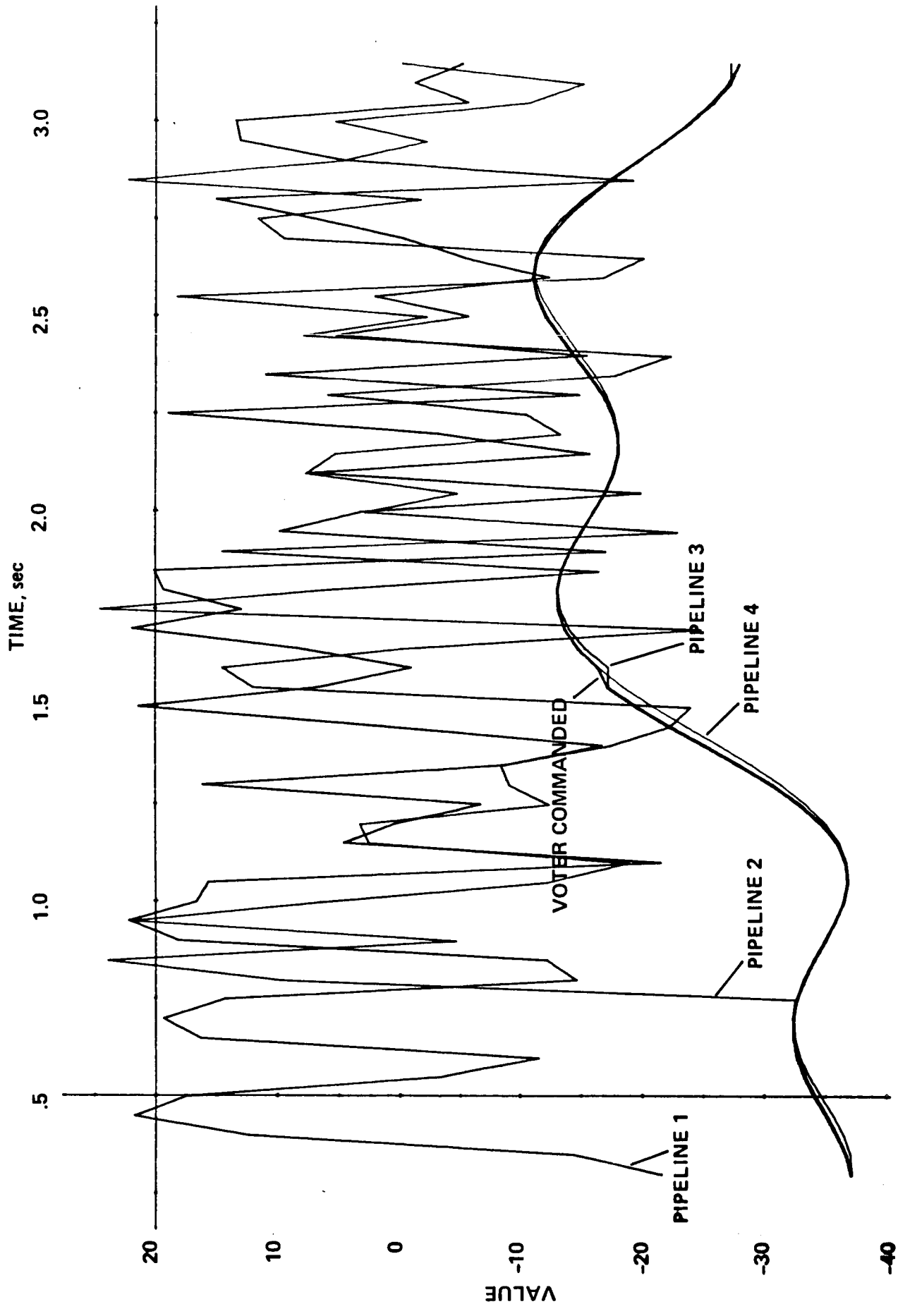


Figure 8.- RESID in random input fault.

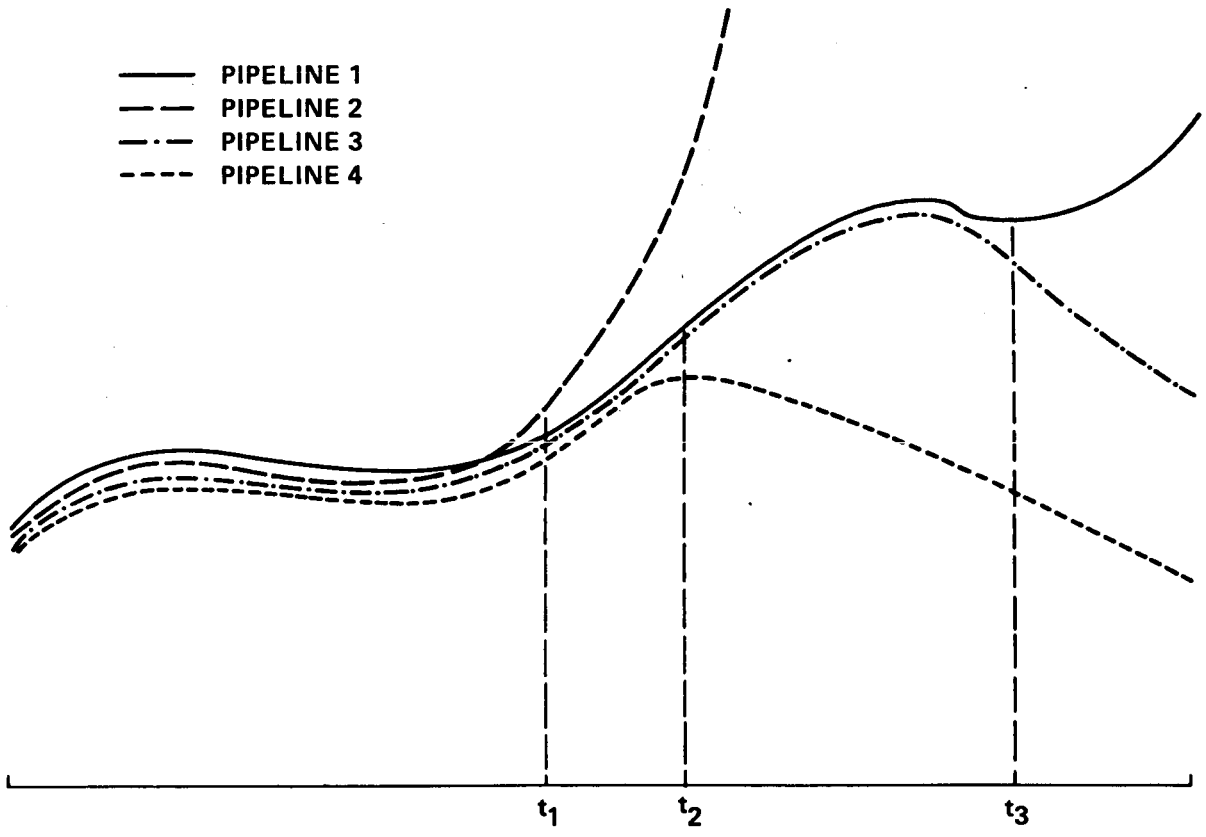
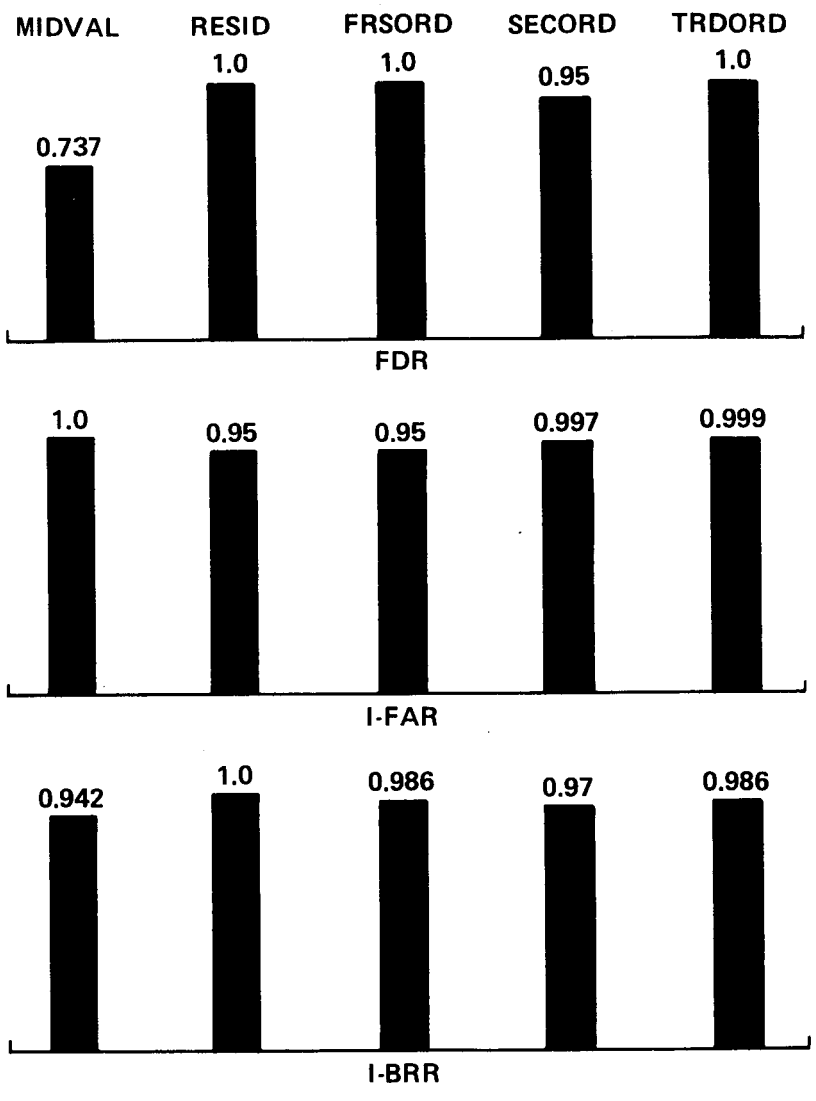


Figure 9.- Constant drift fault.

t1 when pipeline 2 drifts away from the other three. The point at which the voter fails this pipeline depends on the tolerance or variation allowance that was set. Most failure detection schemes will, at some point, detect this drift, its detection time being a function of the drift rate or a frame time allowance. Likewise, with the three remaining pipelines tracking, a drift fault on pipeline 4 becomes obvious at some time t2. After this point, when a drift occurs on one of the two remaining pipelines, proper detection is impossible without extensive historical data to aid in prediction or prior knowledge to what the system is supposed to be doing. At time t3, is it pipeline 1 that failed or pipeline 3? To be able to confidently predict the continuance of the function, the historical data must be lengthy, three or more frames, and requiring calculations too complex for our processing time restriction. It is this reason that all of the voters tested quit after two failure detections.

As seen in table 5, the extrapolating voters and RESID performed reasonably well, RESID and FRSORD's FAR of 5% kept them from being near perfect and SECORD's FDR of 95% was its restrictor. TRDORD was best with less than 2% BRR and 0.1% FAR. In this case, it is obvious that as the order of extrapolation increases, so does the effectiveness of the performance. Figure 10 is an example of SECORD's performance under this fault situation. MIDVAL's performance was too mediocre for a high reliability system with FDR of only 73.7%.

TABLE 5.- CONSTANT DRIFT FAULT WITH 1% NOISE



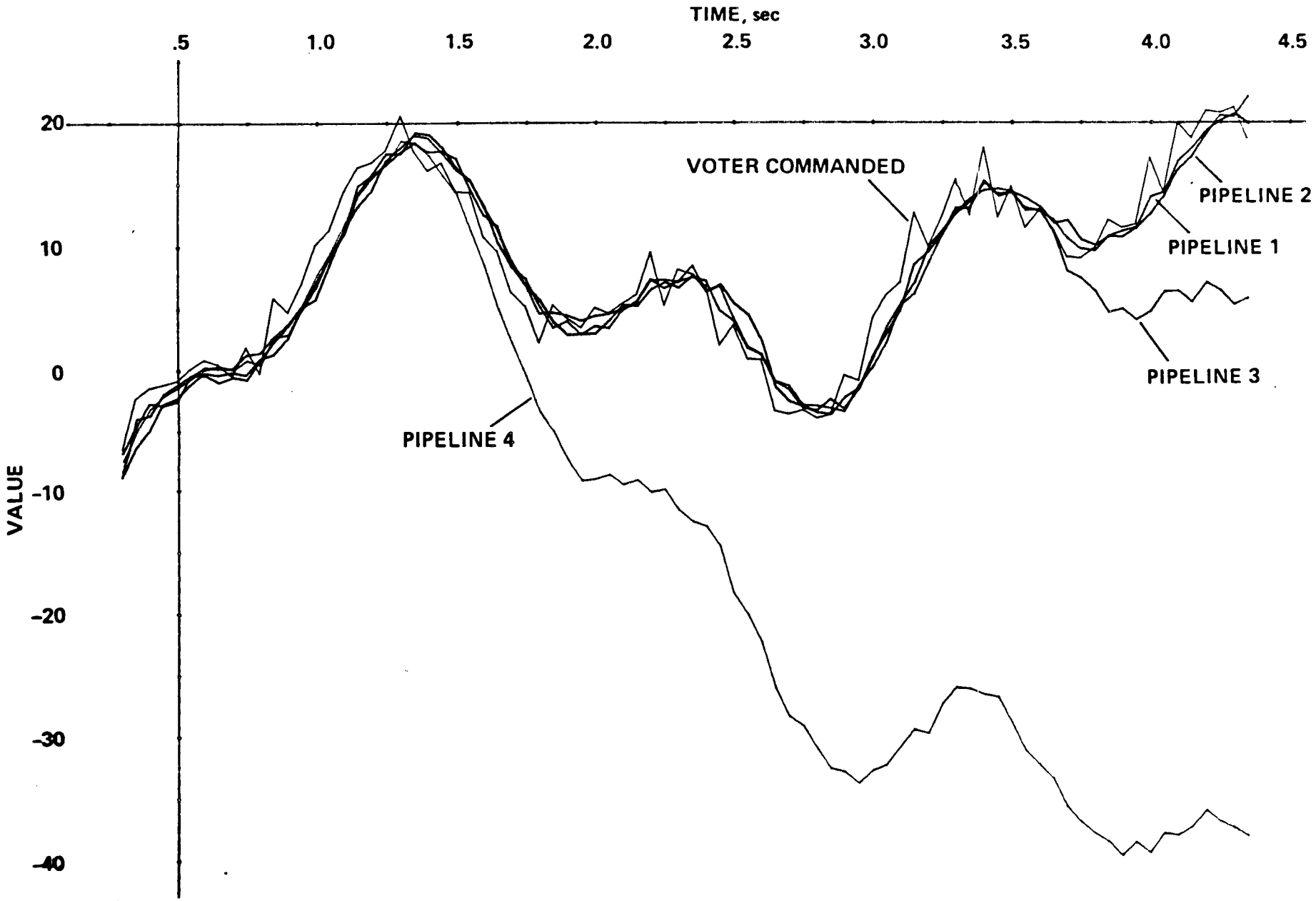
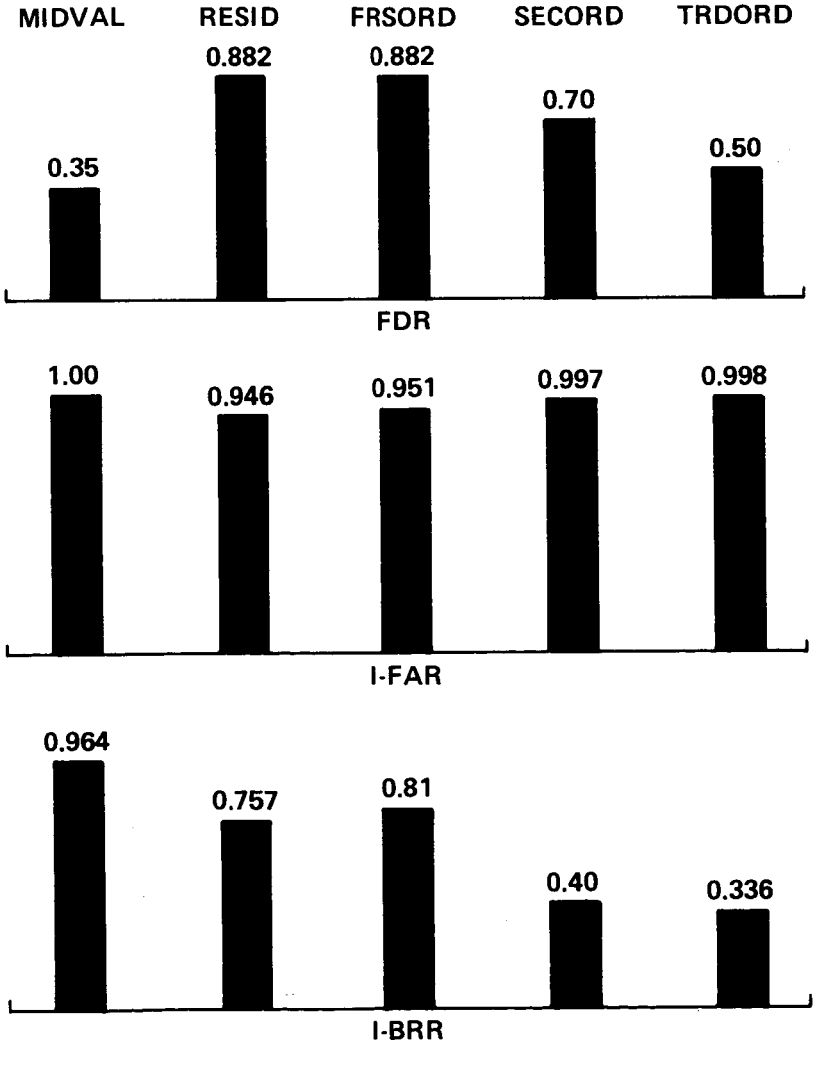


Figure 10.- SECORD and constant drift fault.

Constant offset- Another fault that proved difficult for the voters to detect was the constant offset fault. When present on a channel, this fault does not alter the form of the signal, it simply offsets the value of the signals, thus allowing them to continue to track each other.

This fault was tested with various levels of offset. The lower levels of offset tested, 2.5% to 10%, yielded very low FDRs, high BRRs, and in the presence of noise high FARs. The AMSE was comparable to the level of offset injected. The higher levels of offset, 25%, 50%, and 100%, produced better, yet still unreliable, results. Unlike the constant drift fault, this fault stayed a constant distance away from the unfaulted signals; and, when two channels were faulted, without extensive background data, voters could not detect which set of two channels was faulted and which was correct (see table 6). The extrapolating voters continued to use them all, bouncing back and forth among the channels and beyond. This resulted in a high BRR and added to their AMSE. RESID was better most of the time, but it did hop back and forth some (see figs. 11 and 12).

TABLE 6.- CONSTANT OFFSET FAULT WITH 1% NOISE



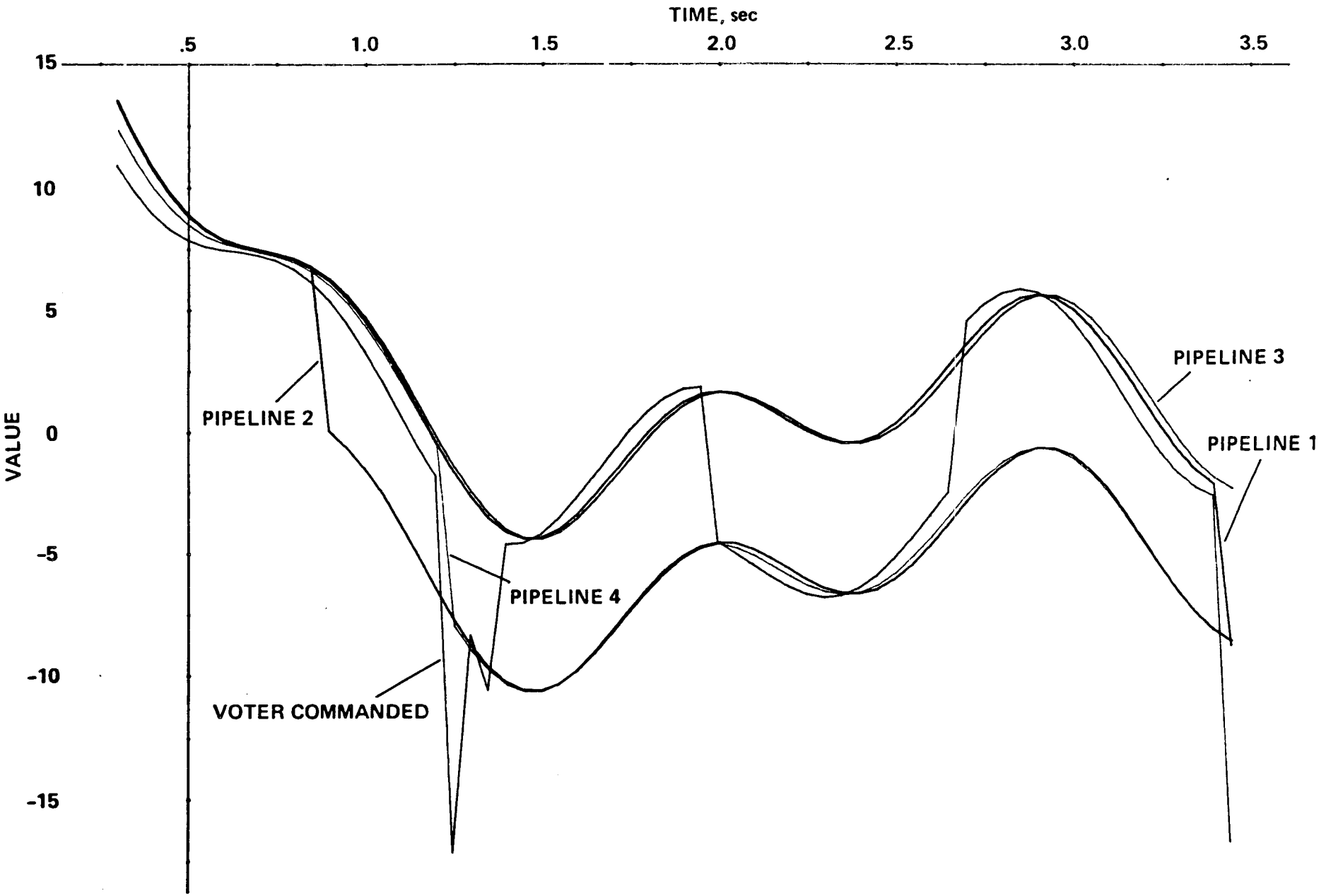


Figure 11.- SECORD in 25% constant offset fault.

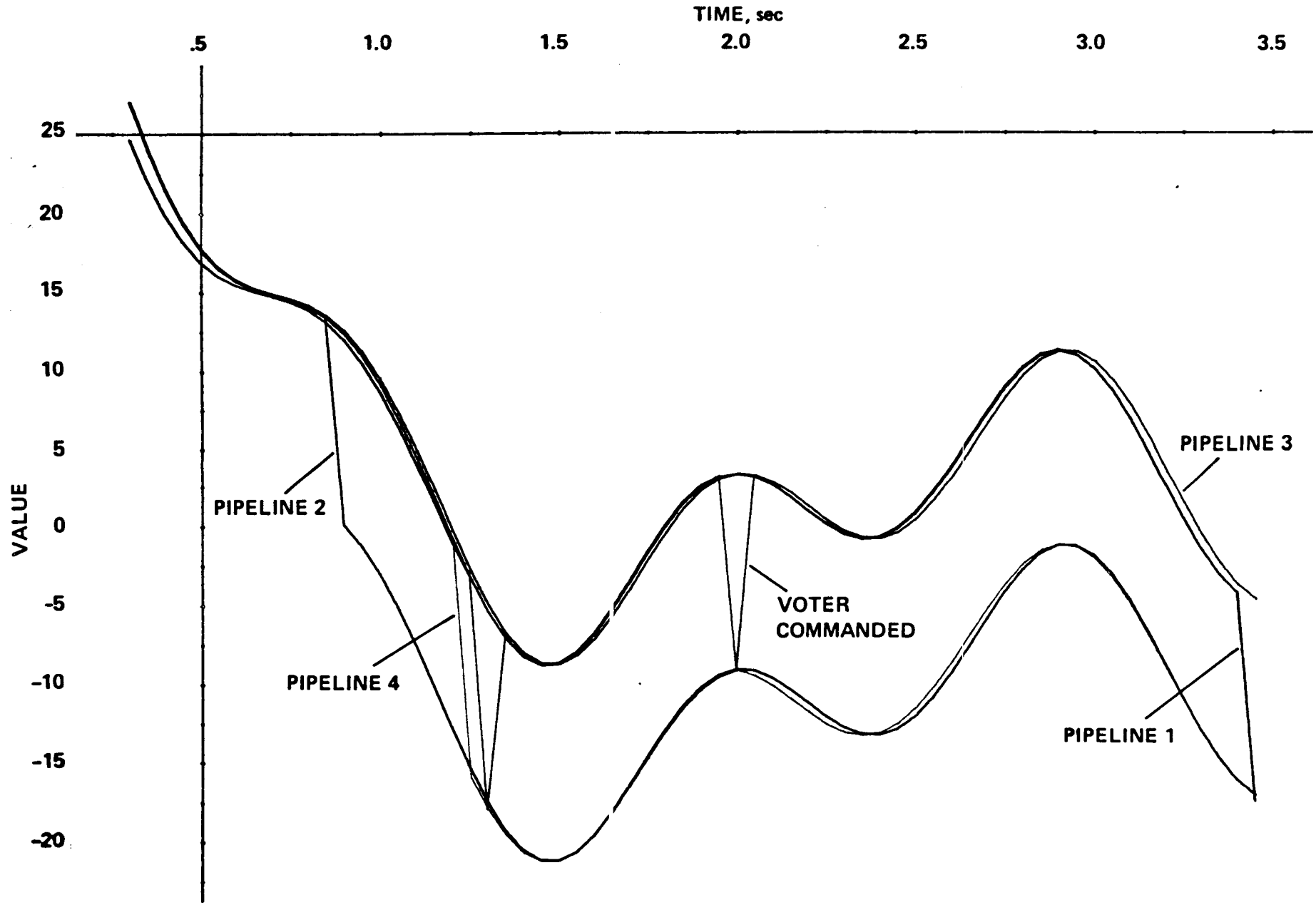


Figure 12. RESID in 25% constant offset fault.

The constant offset fault was never reliably detected, possibly because of overly lenient tolerance levels. In the smaller levels of offset, improper detection of this fault did not produce "bad" output signals, but suffice it to say that under none of the conditions tested was this fault satisfactorily isolated and, therefore, none of these voters is recommended for use in a system with this condition.

Transient spike- Transient impulses or "glitches" occur in most every system. They are not faults in the same sense as those previously tested in that they are not permanent. Their duration is typically less than one frame. Because of the lack of permanence, a pipeline with a transient pulse present should be ignored and not failed permanently by the voter. It is because of the nature of this soft failure that voters either wait several frames to fail a pipeline or continuously monitor a failed line in case it returns to a valid state.

The tests performed on the voters regarding the transient pulse consisted of injections 50%, 30%, and 5% of the control function value with and without noise. The 5% impulse test produced results comparable to those with little and no noise. It did not induce FAR or alter the AMSE. It took injections of 30% and 50% to noticeably affect the voter performance, inducing up to 0.1% FAR in RESID under the presence of 0% noise (see fig. 13). The tests with transients and noise maintained a higher FAR (see table 7) but results discussed earlier indicate this was due to the noise rather than the transient injection. Overall, the voters performed quite well, while not prematurely failing the resident pipeline of the impulses but rather ignoring the signal for the duration of the fault.

Stuck at last with step- The final fault tested was the stuck at last with step increase (S-A-x). This commonly occurs when one or more components of the system pipeline fail but the pipeline continues to output values. This value is generally a stuck-at-x (ref. 8).

To detect this fault, voters usually incorporate an additional detection mechanism for the stuck clock. If the output time has not changed within several frames previously set by the designer based on the performance expectations of the resident system, the pipeline is failed. It is re-enabled only when the time and values are valid again. The stuck signal's detection is generally encompassed by the overall fault detection scheme; no special test is specifically designed for this one fault.

In the presence of the stuck-at-x fault (which is one of the easiest faults to detect) all of the voters performed reasonably well (see table 8) except for FRSORD and RESID. Their performances in FAR were substandard for a high reliability system. The AMSE again reflected the voters performance in fault detection. With high levels of noise, small step faults were difficult to detect. Higher levels of noise introduced higher FARs which also hampered the detection of the stuck fault. SECORD and TRDORD again performed the best on this fault with 100% FDR in 1% noise and 50% step with less than 1% FAR and less than 5% BRR. Obviously, the best scheme for this type of fault is SECORD or TRDORD.

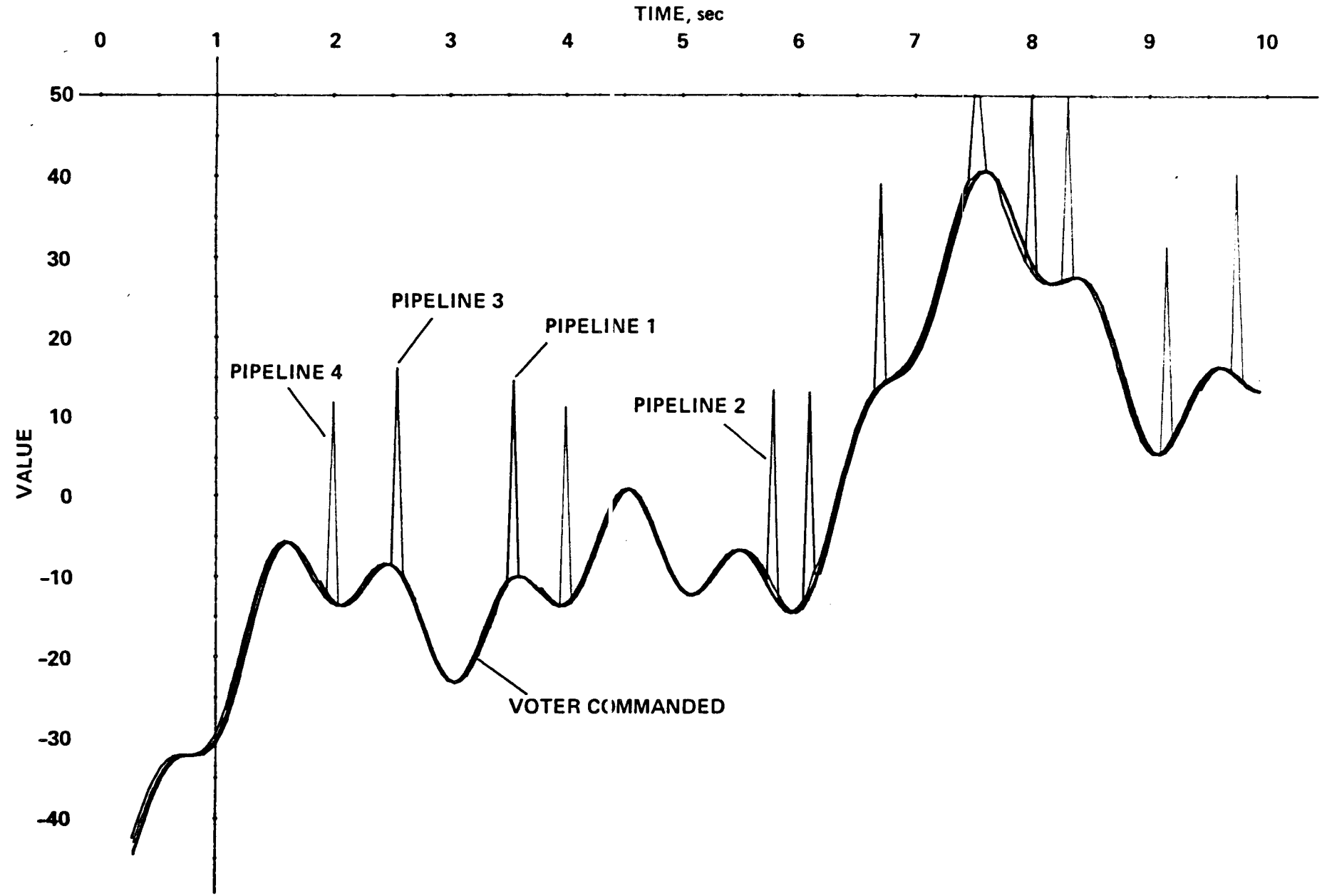


Figure 13.- RESID and 50% transient impulse.

TABLE 7.- TRANSIENT IMPULSE WITH 1% NOISE

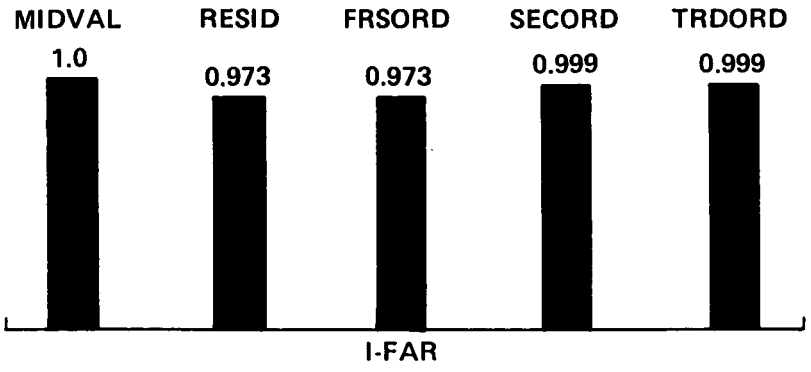
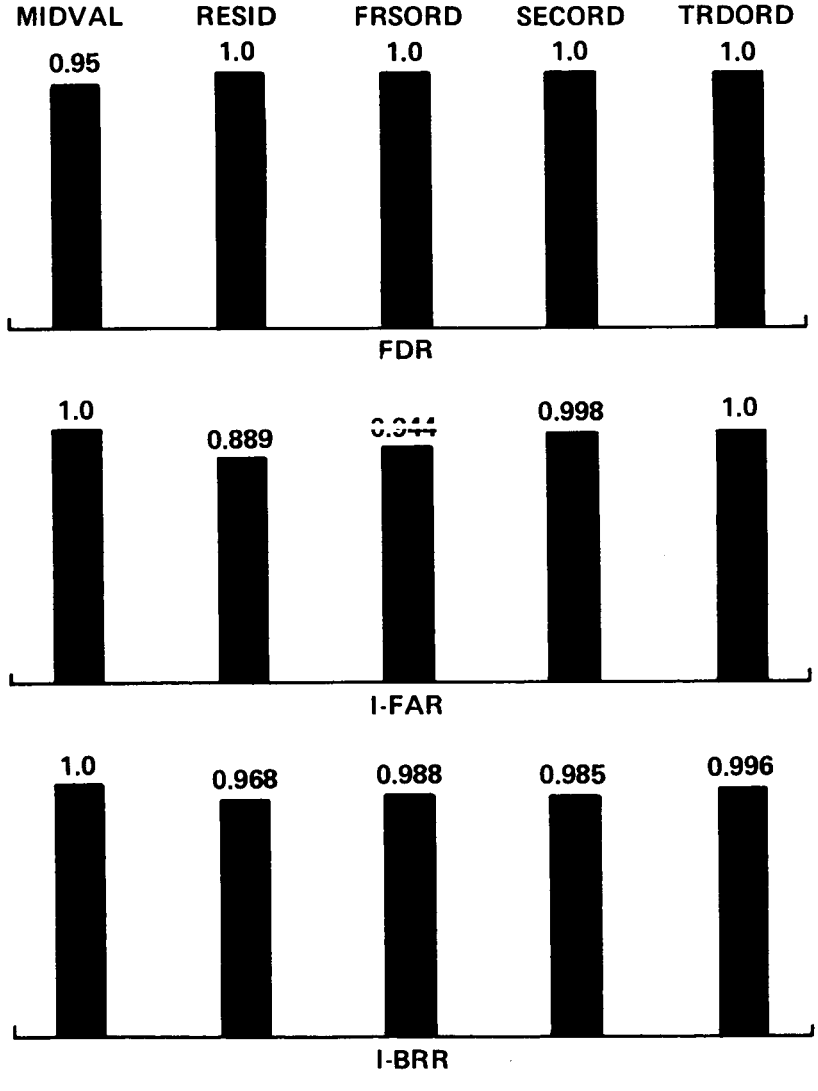


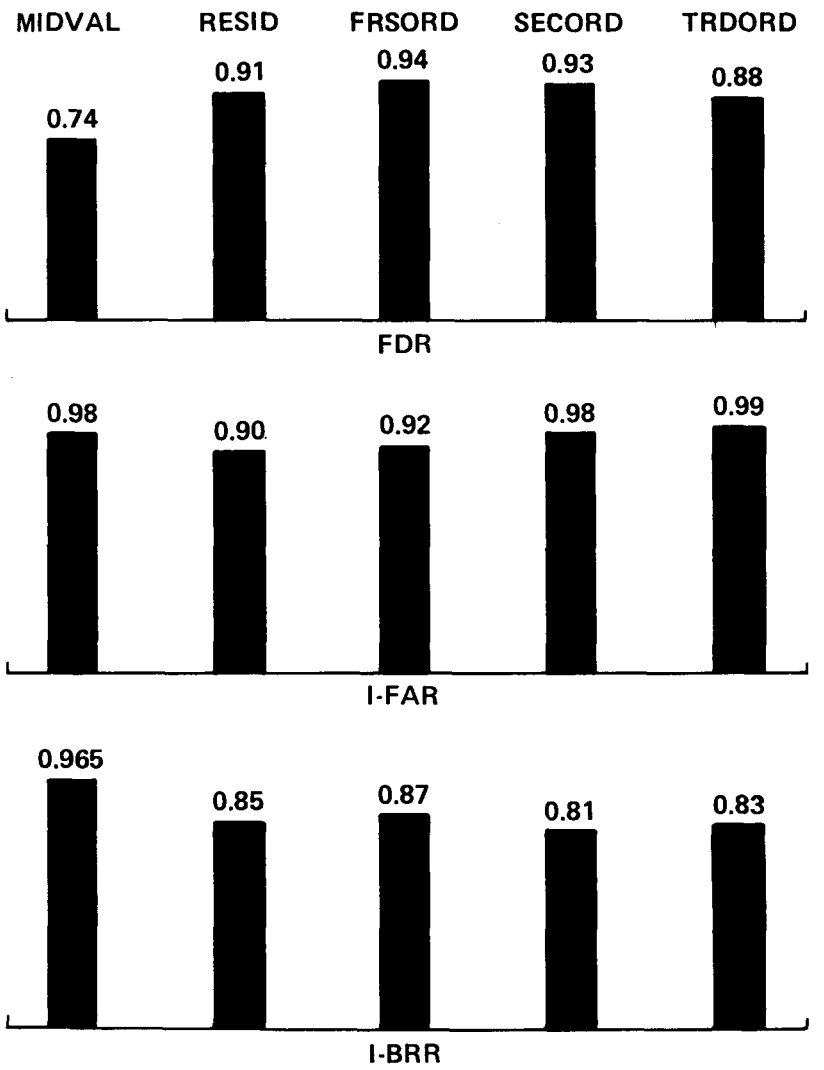
TABLE 8.- STUCK AT LAST FAULT WITH 1% NOISE



Average performance--all faults allowed- A final analysis was performed to determine the reliability of the voters subjected to all of the types of faults equally. These results are an average of all of the tests run using each fault under each of 0, 1, and 2.5% noise conditions.

The extrapolating voters performed best in FDR (see table 9) and FAR. This resulted in their low AMSEs. Their BRR should be improved upon for high reliability. MIDVAL was poor in FDR and its AMSE was intolerable because of this. The AMSE of RESID was also poor. In overall performance, the extrapolating voters are best with their consistent statistics.

TABLE 9.- OVERALL AVERAGE PERFORMANCE (ALL FAULTS WITH 0, 1, 2.5% NOISE)



CONCLUSIONS/RECOMMENDATIONS

To design an effective voter requires extensive knowledge of the types of errors that may occur in the system and the nature of the signals being tested. The results presented in this paper are a first attempt to identify the attributes of alternative voter concepts as applied to the specific set of faults and signal/noise characteristics hypothesized for UFTCS.

General Recommendations

If the control signals are contaminated with high-frequency noise, it is found best to choose a given signal (as in MIDVAL or RESID voters) rather than to extrapolate the output. It was further found that the midvalue select provided a better choice for a given signal than the most recent signal. This is attributed to the inherent averaging in the middle value select method. Improved fault detection performance was achieved using the extrapolator methods (providing the noise content was reasonably low). In order to provide good fault detection rate with a low number of false alarms requires an extrapolation of the signals to a common time. High-frequency noise in the system will, however, significantly degrade the performance of these methods for fault detection.

Specific to the Voters

Midvalue select is a good signal selection routine. In noisy environments it does succeed in sifting out noise when faults have been properly detected. But the fault detection routine used in these experiments in the midvalue select voter was not sufficient for this job.

The extrapolating voters were consistent in their performances throughout the experiments. While having higher AMSEs than the other voters because of magnification at the peaks of the input signals, they also had consistently good performance in fault detection (with the exception of constant offset fault, where none of the voters performed reliably).

Specific to UFTCS

Based on these tests, none of the voters considered were completely adequate for the UFTCS. The signal select voters did not provide adequate failure detection while the extrapolators provided poor signal selection in the presence of high frequency noise. This points toward a new voter design comprised of FRSORD, SECORD, or TRDORD for fault detection and isolation but using the MIDVAL voter logic for signal selection. To validate that these separate concepts can be brought together to form an effective voter for the UFTCS will require further testing.

REFERENCES

1. Westermeier, F.: Triplex Digital Fly-by-Wire Redundancy Management Techniques. AIAA Paper 78-1279, 1978.

2. Wensley, F.; Lamport, L.; Goldburg, J.; Green, M.; Levitt, K.; Melliar-Smith, P.; Shostak, R.; and Weinstock, C.: SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control. Proceedings of the IEEE, vol. 66, no. 10, Oct. 1978, pp. 1240-1253.

3. Sebring, D.; and Young, J.: Redundancy Management of Skewed and Dispersed Inertial Sensors. 4th Digital Avionics System Conference, St. Louis, MO, Paper 81-2296, Nov. 1981.

4. Ammons, E.: F-16 Flight Control System Redundancy Concepts. AIAA Paper 79-1771, Aug. 1979.

5. Noble, W.: Fault Isolation Methodology for the L-1011 Digital Avionic Flight Control System. 4th Digital Avionics System Conference, St. Louis, MO, Paper 81-2223, Nov. 1981.

6. Broen, R.: New Voters for Redundant Systems. ASME Journal of Dynamic Systems Measurement and Control, vol. 97, Series G, no. 1, Mar. 1975, pp. 41-45.

7. Bumby, E.: Redundancy Management for Fly-By-Wire Systems. AIAA paper 72-884, Aug. 1972.

8. Webster, L.; Slykhouse, R.; Booth, L.; Carson, T.; Davis, G.; and Howard, J.: Ultrareliable Fault-Tolerant Control Systems. AIAA Paper 84-2650-CP, Dec. 1984.

9. Dunn, W.; and Meyer, G.: A Fault-Tolerant Distributed Microcomputer Structure for Aircraft Control Systems. AIAA Paper 78-1278, Aug. 1978.

10. Dunn, W.; and Meyer, G.: Design and Analysis of a Fault-Tolerant Distributed Microcomputer Control System. IEEE Conference on Decision Control, San Diego, CA, Jan. 1979.

11. Dunn, W.; Johnston, J.; and Meyer, G.: RAMP--A Fault Tolerant Distributed Microcomputer Structure for Aircraft Navigation and Control. Fourteenth Asilomar Conference on Circuits, Systems and Computers, Pacific Grove, CA, AIAA Paper 82-A-27714 and NASA CR 163615, Nov. 1980.

12. Yeh, E.: Applied Computation Theory, Analysis and Modeling. Prentice Hall, 1976, pp. 285-287, 352-359.

NASA National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM 100007		

4. Title and Subtitle	5. Report Date
Analysis of Redundancy Management Algorithms for Asynchronous Fault Tolerant Control Systems	September 1987
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Gloria J. Davis	A87286
	10. Work Unit No. 505-66-11

9. Performing Organization Name and Address	11. Contract or Grant No.
Ames Research Center Moffett Field, California 94035	
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington DC, 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Point of Contact: Gloria J. Davis, Ames Research Center, MS 244-4, Moffett Field, CA 94035 (415) 694-6526 or FTS 464-6526

16. Abstract

Redundancy management algorithms, commonly referred to as voters, are algorithms used in fault-tolerant control systems to vote on incoming redundant data, isolate "bad" signals, and output a single "good" value. In a synchronous environment, this algorithm is a straightforward signal-to-signal comparison with relatively low complexity. The technology of asynchronous control systems, recently realized in the Ultrareliable Fault Tolerant Control System research program at NASA Ames Research Center, requires more complex algorithms for fault detection and signal selection. A variety of algorithms used for this process, a means of testing them, and their basic performance under a simulated environment of the ultrareliable fault-tolerant control system are presented.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Redundancy management Voter(s) Asynchronous redundancy Fault tolerance	Unlimited - Unclassified Subject category: 01

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	35	A03