Fault Tolerant Issues in Jet Engine Compressor Control

by

David A. Seal

Submitted to the Department of Aeronautics and Astronautics in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

Massachusetts Institute of Technology

June 1991

Signature of Author ______________________________
Department of Aeronautics and Astronautics
June 1, 1991

Certified by ______________________________
Professor Lena Valavani
Thesis Supervisor

Certified by ______________________________
Dr. Richard E. Harper
Technical Advisor, CSDL

Accepted by ______________________________
Professor H. Y. Wachman
Chairman, Departmental Graduate Committee
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology

Fault Tolerant Issues in Advanced Jet Engine Compressor Control

by

David A. Seal

Submitted to the Department of Aeronautics and Astronautics in partial fulfillment of the requirements for the degree of

Master of Science

Abstract

The application of modern control technology to aircraft turbine engines requires new methods of ensuring reliability in all portions of the control system. Recent advances at the Gas Turbine Laboratory at MIT have implemented active control of rotating stall of a low-speed compressor with a performance improvement better than twenty percent. The distributed systems which implement this control law are especially susceptible to single- and multiple-component failures throughout the compressor's lifetime; therefore, they must be made fault tolerant. The results reported in this study demonstrate the need for Byzantine Resilience and address in detail the application of expectancy modelling in this unique distributed environment. After attending to timing requirements and other issues, four fault detection and isolation mechanisms are designed -- local parity tests, diffuse parity tests, power spectral density tests, and failure detection filters -- and applied to numerous single and simultaneous failure modes with impressive results. For all of the injected faults, one or more algorithms detects the failure error(s) easily within the required response time with a minimum of false alarms, though the operation of each is fundamentally different. The primary conclusion of this report, however, lies in identifying the need for embracing many detection mechanisms employed by an expert system to achieve complete fault coverage.

Thesis Supervisor: Lena Valavani
Title: Associate Professor of Aeronautics and Astronautics
CSDL Advisor: Dr. Richard Harper
Title: Projects Manager, Fault Tolerant Systems Division

Acknowledgements

The author wishes to recognize the outstanding personal and technical support of Dr. Richard Harper, Professors Lena Valavani and Wallace VanderVelde, and James Paduano, who have all contributed significantly to this thesis. Thanks also to Vickie, Paul and Mom for being around.

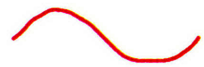
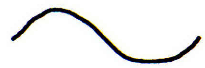
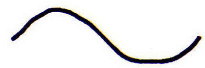
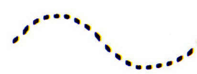
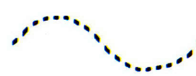
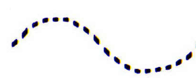
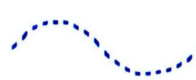
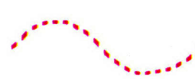
This report was prepared at the Charles Stark Draper Laboratory, Inc., under Corporate Sponsored Research Technical Plan Task 353 and with the cooperation of the Gas Turbine Laboratory at the Massachusetts Institute of Technology.

Table of Contents

List of Figures

Key to Multicomponent Color Graphs

Many of the figures listed above are color plots with eight or twelve lines, each corresponding to error indications of a particular component. Included is a reference key to the colors and their significance.

Element Plotted	Color	Illustration
Component 1	Red	
Component 2	Green	
Component 3	Dark Blue	
Component 4	Light Blue	
Component 5	Pink	
Component 6	Yellow	
Component 7	Dotted Black	
Component 8	Dotted Red	
Component 9	Dotted Green	
Component 10	Dotted Dk Blue	
Component 11	Dotted Lt Blue	
Component 12	Dotted Pink	

Chapter 1
Introduction

1.1 Problem Motivation

Compressor stall and surge are two common problems which plague aircraft engines on both military and commercial vehicles. Small disturbances begin in the inlet flow and , if compounded, can cause a complete stall or surge, reducing thrust severely. The "zeroth order mode" or **surge** manifests as a one-dimensional mass flow oscillation , whereas higher order harmonics are associated with rotating or propagating stall. In the latter case a zone of disturbed flow, the "stall cell," propagates circumferentially and grows with time until it pervades the entire annulus. The consequences can be severe: pressure ratio can drop dra[illegible] thrust is greatly decreased and the compressor blade stresses can be enormous. [illegible]eral, however, these dynamic instabilities start as small disturbances. Stall "precursor waves" can be detected and feedback control used to suppress them.

Cases of compressor stall have been numerous and well documented from the very beginnings of jet-assisted flight. Catastrophic compressor failures have led to civilian and military accidents, and often investigators have concluded that proper actions taken in a [illegible]ion could easily have prevented the disaster. The window during which such [illegible]ive, however, can be as narrow as tens of milliseconds, so pilots cannot be [illegible]per precautions in time. The flight control task is to utilize whatever control resources remain in order to maintain control of the aircraft and to prevent further damage by any failures, while giving the crew time to assess their options.

The MIT Gas Turbine Lab has been exploring compressor control for several years. In disturbance tests of compressors with movable inlet guiding vanes, GTL has discovered that airflow control can recover a low-speed compressor from deep stall and increase the flight envelope to better than 20% in mass flow reduction. Such a system, if implemented in a flight-quality integrated controller, could dramatically improve both performance and safety.

Historically, the use of computer controls in complicated dynamical systems such as compressors have been overlooked for a number of reasons:

- Limited computer capacity and poor sensor development;
- High reliability favored a simple system;

- Mechanical alterations improved performance greatly without the addition of complicated control systems.

Even during the 1980's, however, engine controls have needed to explore advanced technology in order to accommodate stringent processing needs imposed by the dynamic response of flight instabilities. An engine with a rotating stall compressor controller would require a high-speed high-throughput control system, which could significantly increase performance and efficiency. However, the sophisticated control laws involved would make the compressor vulnerable to failures of the many sensors, actuators and electronic components of the system.

Because of this vulnerability, the controller must be designed to be fault-tolerant -- that is, to be able to sustain one or more component malfunctions without loss of performance. A complex system such as an engine compressor is made fault tolerant by installing redundant components, providing back-up modes of operation to be used when nonredundant systems fail, or by incorporating in the system various means for automatically detecting and identifying failures so that the appropriate compensation can be selected [Mes81]. A component or set of components of the active and passive control equipment may fail to perform its task, due to physical damage, fire, lightning, and extreme deviation from the design environment, including temperature, airspeed, vibration, shock, and EMI -- and the resulting errors can be more severe than no control at all. Also, a system composed of many VLSI components, each of which may be highly reliable, can be incapable of satisfying stringent reliability requirements when linked.

1.2 Objective

This project intends to expand the theoretical foundations available for detecting faults analytically in compressor control components. Such an approach would use linear filtering technology, along with basic information about the operation characteristics of the compressor itself to generate warnings if sensed signals depart from the predicted flight envelope. This kind of technique has been demonstrated successfully in numerous single-input single-output systems before; what makes the approach unique here is the inclusion of distributed systems, such as sensor and actuator banks, in the reliability assessment. Analytical redundancy of this kind (which is more a kind of "expectancy assessment" than traditional analytical modelling) cannot detect all failure modes, but can be extremely useful in improving the safety of a coherent compressor system. Issues which will be addressed include: basic fault resiliency requirements; the dynamical nature of the engine compressor; various mathematical methods for detecting errors analytically; and failure tolerant compressor architectures. All of these concepts will be applied within the context of the test stand currently in operation at the MIT Gas Turbine Laboratory.

1.3 Thesis Outline

This study begins with the discussion of relevant fault tolerant theory in chapter two. In chapters three and four, a compressor simulation model is described, and various methods of isolating errors are delineated and applied to specific fault cases in the distributed sensors/actuators. Three approaches for failure identification and isolation are discussed: parity relations, failure detection filters and power spectral density analysis. A set of detection mechanisms are designed, and general methods of error diagnosis are [illegible]ach. In chapter five, failures are simulated using actual compressor data, and [illegible] made according to the results of distributed component system analysis.

Chapter 2

Fault Tolerance Fundamentals

2.1 Definition of Fault Tolerance

It is important to introduce the fundamental issues of fault tolerance in order to establish terminology so as to compare and contrast conventional approaches to those presented within this thesis. This chapter focuses in particular on the issues relevant to robust compressor controls; much of it has been condensed from [Har87] and [Abl88].

Simply put, fault tolerant computing is "the ability to execute specified algorithms regardless of hardware failures, total system flaws or program fallacies [Kim75]." Traditionally, however, efforts devoted to ensuring fault tolerance have been limited to the design and implementation of computer processing systems, and can overlook the peripheral components such as sensors, actuators, and their related communications paths which constitute the link between the processor and the plant. Failure in any such components can be as catastrophic as those in processors, since even systems which are processor-fault-tolerant might be unaware that their measurements are grossly erroneous. A reliable fault-tolerant system cannot be complete without the guarantee of fault tolerance in **all** areas of its influence, including sensing and actuation. This thesis is an attempt at addressing these issues, specifically applied to the distributed sensing and actuation system on the MIT Gas Turbine Lab's low-speed jet engine compressor.

2.2 Approaches to Fault Tolerance

Fault tolerance is conducted using redundancy theory to detect and correct failures as soon as possible, ideally before contamination of the system outputs. There are many forms of redundancy, each with its particular tradeoffs to be considered when designing a fault-tolerant system (FTS). These mechanisms can employ redundancy of device, information, or time, to detect and isolate faults. **Analytical Redundancy**, or **functional methodology**, is a high-level approach which verifies system operations by examining response time, output working area, and the qualitative aspects of the computational results.

Error detection is based on the inherent properties of the system, and where there are clear departures of a component from its estimated envelope, failure assertions can be declared. This thesis deals exclusively with methods in this category.

Two fundamental philosophies exist towards ensuring fault resilience. The first is based on designing systems to cope with only those failure modes which are deemed "likely." However, this approach is seriously questioned when considering the extent to which human assumptions will limit the subsequent robustness of such a system. A reliable sensing and actuation system must be able to cope with **arbitrary** failures of one or more of its components. This arena of fault analysis, which can be termed a Byzantine Fault classification[1], may consist of intermittent bursts of erroneous data, transmission of different information to several targets, increases in noise levels, changes in overall signal magnitude, or many other types of malicious behavior. This is an important approach in diagnosing faults; one can never predict with complete confidence what failures will occur, as lifetime testing under <u>all possible</u> environmental conditions is never possible.

Implementing a Byzantine tolerance of **arbitrary** failure modes by refusing to delineate sets of possible outcomes can be the only approach to designing systems capable of meeting the reliability needs of mission critical situations. This is the core of the approach which the Draper Laboratory has helped pioneer in fault-tolerant technology, and is especially relevant to compressor controllers due to the extreme conditions under which their sensing and actuation subsystems may be operating.

In a jet engine, a Byzantine resilient system must have the capacity to hide the system's redundancy and provide guarantees on the accuracy of sensor and actuator [illegible]. This concept has traditionally been applied only to computers; applying it to sensing and actuation methodologies represents an expansion of the conventional theory into an uncharted area. Nevertheless, the goals of this kind of fault tolerant mechanism are similar. In the face of component faults **of any kind**, the controller must prevent errors from affecting the plant's performance by removing their influence from the control loop. This is illustrated for a distributed sensor system in Figure 2-1:

[1] A complete explanation of the Byzantine Generals problem and its solution in the Draper FTPP can be found in [Har87], [Sak91], [Abl88] and [Lam62].

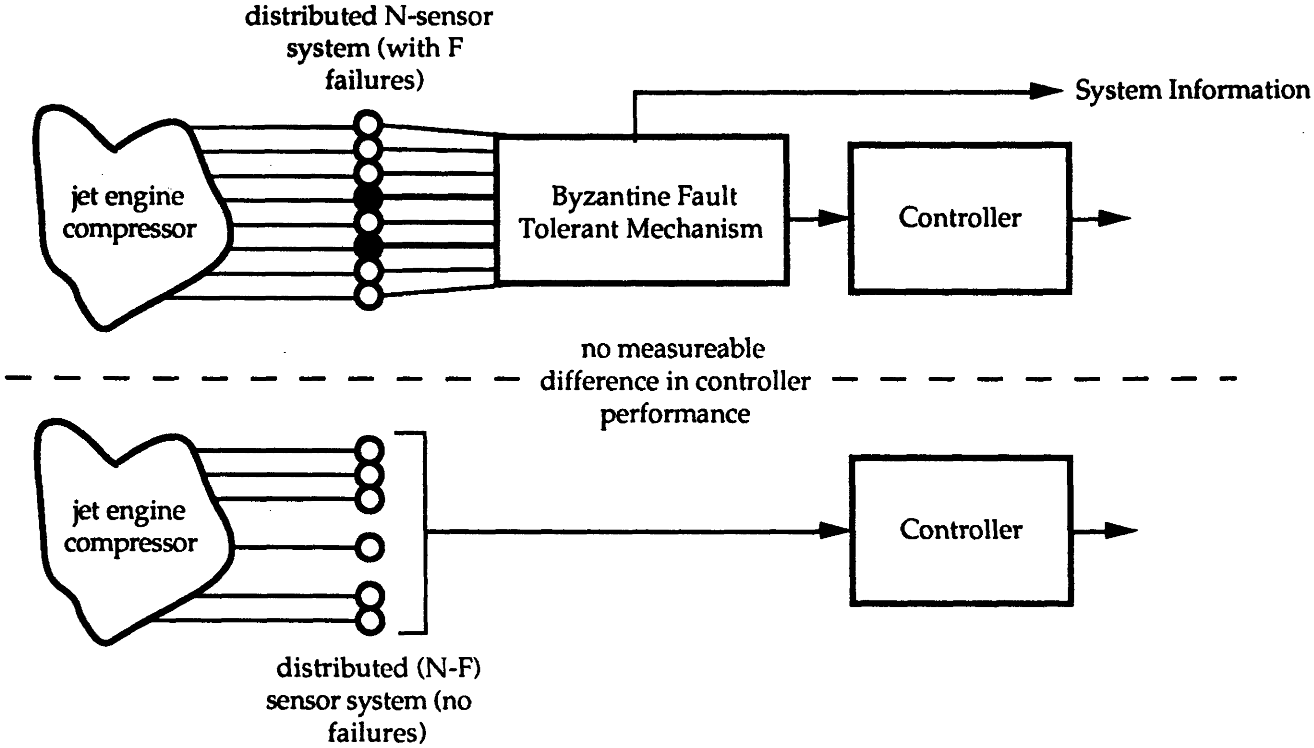


Figure 2-1 Role of a Byzantine Resilient Mechanism

Any type of errors in a number (F) of (blackened) sensors must be masked so as to produce results indistinguishable from an identical (N-F) system with no failures. For compressors, this system could be qualified by experiments where failures of arbitrary nature are seen to have no effect on the stall point of the system (were it functioning with F less sensors), allowing the compressor to perform within the same performance envelope in the face of component failures.

In the past, reliability analyses have been performed with the assumption that faults are detected immediately as they occur, and nonfaulty operation never triggers false alarms. This approach ignores the damage caused by imperfect fault tolerance mechanisms and error latency. In a jet engine, the state variables can vary wildly under momentary gusts or other disturbances so as to exceed the analytically-predicted flight envelope, perhaps causing a failure to be registered where none exists. Extreme care must be taken, when defining analytical mechanisms, that departures from steady-state operation are provided for, though not so stringently as to miss most failures when they do occur. In addition, detection does not always mean isolation. Awareness of a fault's existence is important but in the long run insignificant without the capability to identify which component caused it. Of the several kinds of decision errors, an incorrect isolation is by far the worst, because it combines the effects of both a false alarm and a missed detection occurring simultaneously.

Adapting compressor detection mechanisms to eliminate false alarms and missed detections is not an exact process; it is instead an exercise in probability. Based on the reliability requirements one must endeavor to minimize the probability of false alarms and missed detections while maximizing the probability of quick detections. In order to test such a process without exhaustive and time-consuming on-site experimentation, some assumptions have to be made to partition compressor fault behavior so that responses might be quantified.

2.3 Fault Modelling and Classification

Classifying compressor faults into different categories, each of which can be attacked by a different algorithm, can be useful for partitioning responsibility among a number of detection mechanisms to decrease response time. This approach, however, can be very dangerous if fault classes cannot achieve complete coverage. The classifying process must be done intelligently and with some measure of generality. The greater the specificity, the more chance for a missed failure mode. One viable classification scheme, which could apply to virtually any dynamic system, is suggested below.

• **The inert fault.** This can be the most benign fault, in which the output stream simply stops. Causes of this type may include loss of power or an open circuit. Depending on the nature of the component, this fault may manifest itself in a number of ways. A hot wire anemometer measures airspeed in a compressor by the temperature of a heated wire exposed to the airflow. This temperature corresponds both to how much heat is transferred away by the fluid (and thus, how much fluid per second is passing by) and to the resistance of the wire (which varies with temperature). A broken hot wire would result in a measure of infinite resistance which may indicate a zero airspeed reading as the no-signal output stream is interpreted by the control system.

• **The zero fault.** The component's output has become zero, perhaps resulting from shorted wires or power problems.

• **The calibration fault.** The component's output is active and transmitting data that is too large or too small. These failures are regarded as soft failures, and are much harder to detect as they require some measure of boundary checking and/or analytical redundancy. Note that for low magnitude problems, this fault can be combined with the previous one in a single detection mechanism.

- **The pegged fault.** The output stream has pegged at a constant value and is transmitting essentially the same value continuously. This fault could also be combined with the zero fault in a single detection mechanism designed to mask constant-level failures, independently of the first dual mechanism described immediately above.

- **The bias fault** is another "soft" failure where the measurements are shifted above or below the correct values.

- **The arbitrary fault.** This describes any fault whose signal behavior cannot be classified into any of the first three categories (and thus completes the coverage requirements of a Byzantine fault classification). It may manifest itself as random walk or add large components of noise to the measurement. Causes of these are numerous, but can include vibration, noisy transmission interfaces, or EMI.

- One cannot, of course, forget **intermittent faults**, which can exhibit any of the above, or **latent faults**, which will produce no detectable error but have the potential to do so in the future. It is the latter type of fault, existent but inactive, that is by far the most dangerous. However, latent faults by their very nature cannot be detected by analytical redundancy as they produce no output errors; other methods such as off-line tests and signal checking must serve to ascertain their existence. This thesis will focus primarily on nonlatent faults, i.e. those which generate errors.

The essence of good fault tolerant design ascribes to a philosophy of designing systems derived ultimately from a small set of reliable constructs. To build a good fault-tolerant compressor controller, only reliable components can be used as building blocks. Analogously, when designing any system which interfaces with independent components such as sensors and actuators, care must be taken to ensure that failures of these instruments, as potentially catastrophic as that of processors, will not affect the overall performance.

Chapter 3

Phenomenology of the Jet Engine Compressor

3.1 Stall Dynamics of the Compressor

Recent design work at the Gas Turbine Lab at MIT represents the first success in controlling rotating stall in a low-speed axial flow compressor. Fully developed rotating stall can drastically affect overall compression system flow, and has been linked, at least partially, to deep stall inception.

Compressor performance and operating point are often best measured on a pressure map, as in Figure 3-1. The point on a compressor map at which the fluid flow becomes unstable is generally near the maximum pressure rise and, thus, the maximum efficiency of the machine. Operating a compressor is therefore a compromise between efficiency and safety.

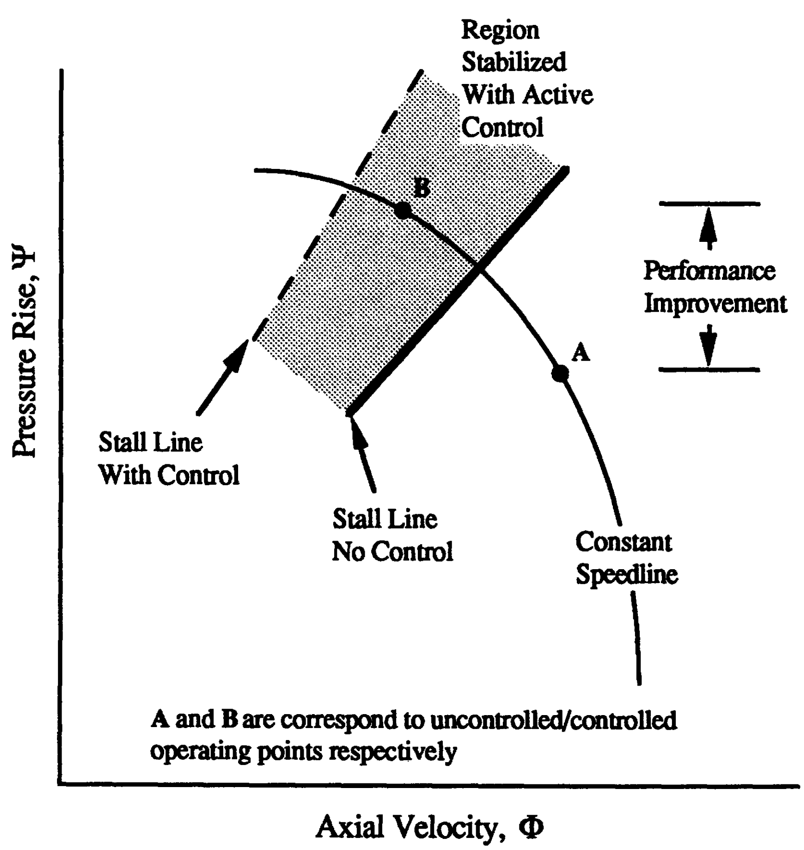


Figure 3-1 Engine Compressor Map

Designers will typically identify a stall "margin" as a buffer of safe operation between the operating point and the stall line. Active control of rotating stall would hopefully push this line upwards in Figure 3-1 and allow the compressor to operate closer or past the open-loop stall line, closer to the region of the highest achievable performance (high pressure and low mass flow) while maintaining adequate stall margins during all phases of flight. This system would permit the pilot to aggressively maneuver near the edge of the uncontrolled flight envelope without fear of sudden irrecoverable departures from controlled flight [Pad90] &[Coh89].

The necessity to avoid rotating and complete stall has been identified as the dominating performance-limiting factor both for centrifugal and axial compressors. Complete stall (also called "surge") is a global one-dimensional instability which involves the whole compression system, including the plenum, duct and throttle. Mass flow undergoes large amplitude oscillations, while the plenum (the combustor) pressurizes and depressurizes. In deep surge, flow reversal can be observed, often causing flames to shoot out the front of the engine. Rotating stall, on the other hand, is a two dimensional phenomenon: fluid velocity varies in both the axial and circumferential directions, and is characterized by local instabilities, i.e. a region of the annulus where little or no through flow exists [Gar89]. Regions of disturbed flow, called "stall cells," are seen to propagate circumferentially until they contaminate the entire annulus, growing into a coherent traveling wave with the speed of rotating stall. The "rotation" characteristic can easily be seen in Figure 3-2.

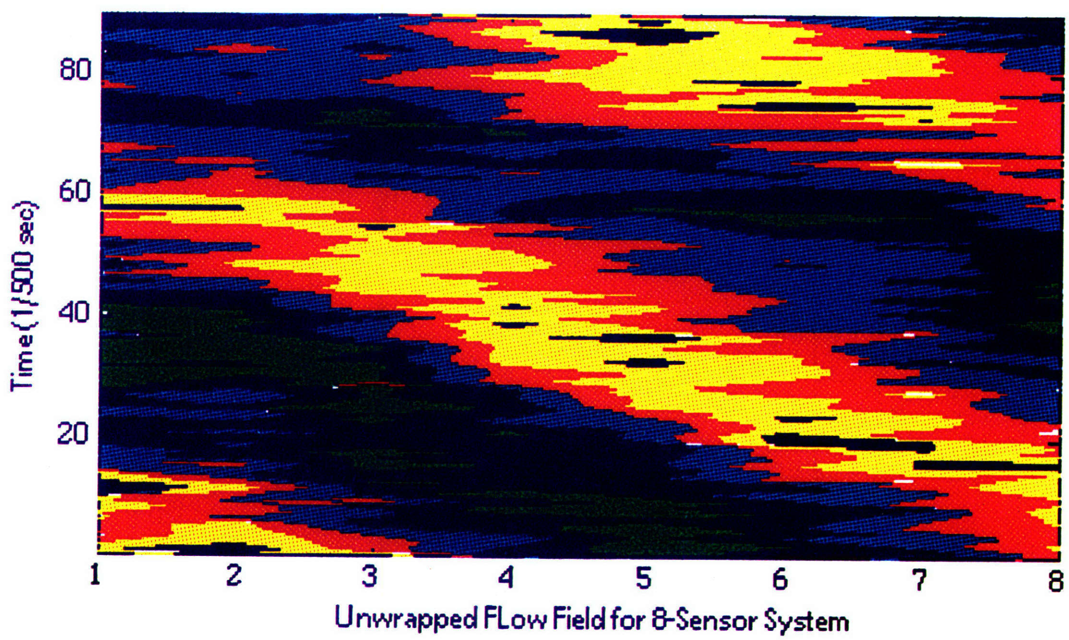


Figure 3-2 Disturbance Propagation Graphic

This graphic shows a map where actual compressor flow data has been "unwrapped" in circumference and displayed much like a USGS topographical map, where equal magnitudes are displayed along contours. High points correspond to black and yellow regions; low points to blue and green. There is a clear indication that disturbances travel from right to left with considerable coherence, at a speed which corresponds to the slope of the paths; a failure in one or more speed sensors (eight in this case) would presumably show up as a vertical band of incoherent colors.

The disturbance velocity typically falls between 20-50% of the rotor speed, and can be identified long before full stall development. These perturbations can cause drastic jumps to compressor map operating points that are very low-efficiency or catastrophically unrecoverable.

3.2 Rotating Stall Modelling

The combination of theoretical and experimental successes by [Gar89], [Moo83], [Gre86] and others in the late 1970's and 1980's combined to capture essentially all the salient dynamics of stall and surge and predict accurately the observed features of engine compressor characteristics. Rotating stall's trademark manifests as small amplitude rolling waves which grow "linearly" from their inception. Machinery which goes into a "deep stall" has been shown to have exhibited this rotating stall condition for some time before instability and, for axial compressors, long enough that it could have been identified and prevented.

At inception, the dynamics of the disturbance propagations discussed above can be accurately modelled by linear theory. The Gas Turbine Lab has developed a theoretical model for such dynamics and proved it effective in feedback control using sensing and actuation devices evenly distributed around the compressor annulus [Pad90]. This method of achieving centralized control through distributed sensing and actuation is a unique feature of this controller, and is documented extensively in [Pad90].

The central method of the linear model relies on composing a set of spatial Fourier coefficients from measurements around the annulus of the compressor to constitute an estimate of the rotating stall process. Given the underlying physics, the model seeks specific traveling velocity perturbations which are periodic in time. State variables are defined as sets of complex Fourier coefficients corresponding to each spatial mode of the disturbance wave. The modes (n) of the travelling wave's real and imaginary Fourier harmonics (X) will behave in the following manner:

$$\dot{X}_n = \begin{bmatrix} \dot{X}_n^{\Re e} \\ \dot{X}_n^{\Im m} \end{bmatrix} = \begin{bmatrix} \sigma_{rs} & -\omega_{rs} \\ \omega_{rs} & \sigma_{rs} \end{bmatrix} \begin{bmatrix} X_n^{\Re e} \\ X_n^{\Im m} \end{bmatrix} \equiv AX_n$$

where ω_{rs} is the rotating stall frequency and σ_{rs} is the slope of the pressure rise characteristic. This model represents the propagation of the travelling wave in time. Clearly, the homogeneous stability depends on the value of σ_{rs}. The eigenvalues of the harmonic's time evolution are obviously

$$\lambda_{1,2} = \sigma_{rs} \pm j\omega_{rs}$$

Increase in these Fourier harmonics can be seen well before full stall develops, and can be implemented in some form of analytical redundancy. Observation of the dynamics of the compressor reveal that the second and third modes are considerably more stable than the first; it is expected that even higher order modes will behave similarly.

A schematic of the GTL setup is shown in Figure 3-3. Included are the test stand as well as the signal processing system, control software blocks and control actuation loop.

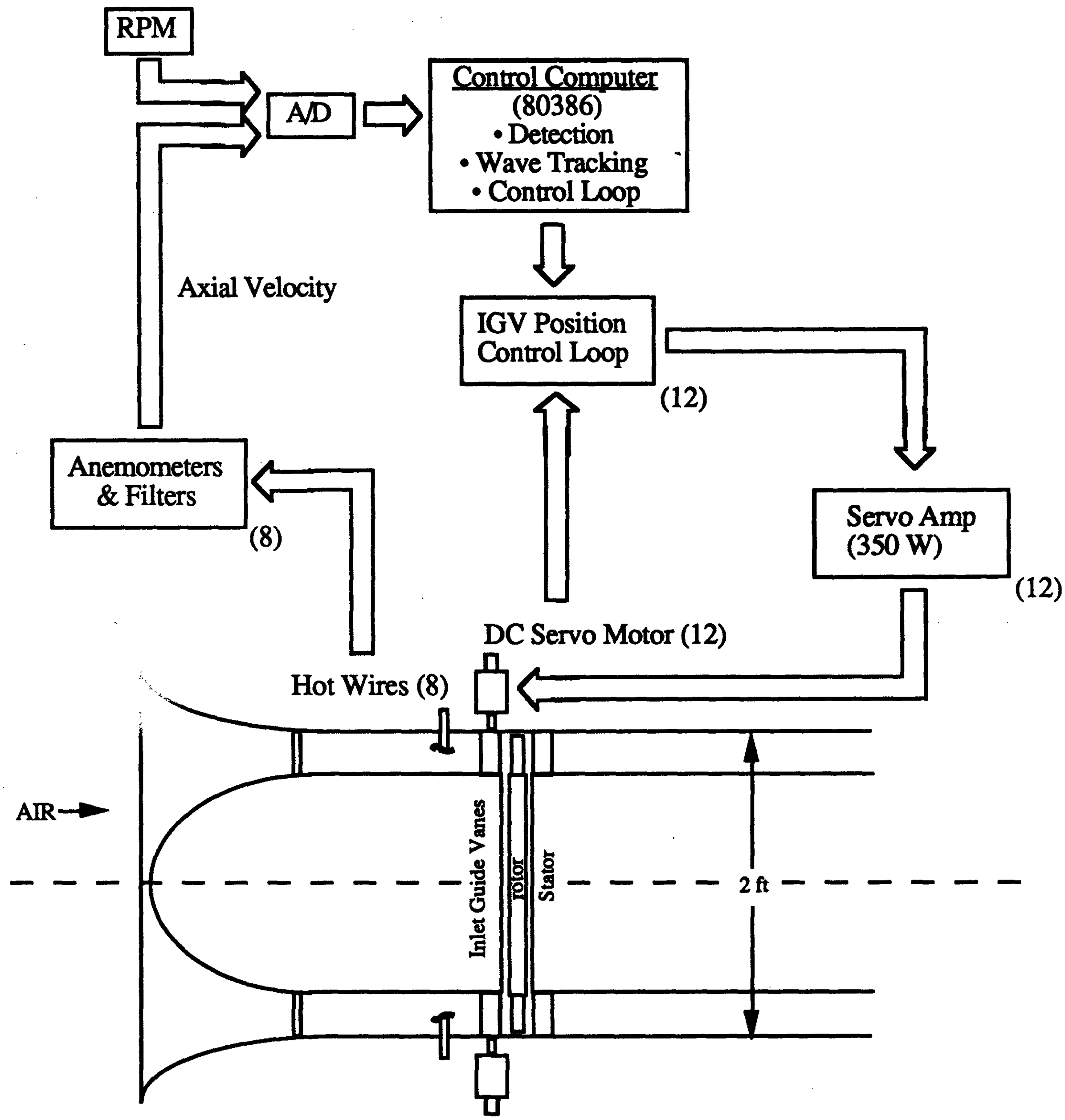


Figure 3-3 Schematic of Rotating Stall Control Setup

The strength of the GTL controller lies in its distributed sensing & actuation systems which, as a whole, constitute a single "virtual" sensor and actuator. Eight hot wire anemometers are equally spaced upstream of the compressor to pick up the perturbations of the flow coefficients; the 12 actuators spaced around the compressor are brushless DC "wave launchers" which can deflect by up to ±30 degrees, usually normalized about the mean Inlet Guide Vane angle.

The eight sensors are low-pass filtered against very high frequency noise and used to compose the set of Fourier coefficients. Since at every location the measurement is a linear combination of the spatial frequencies, the Discrete Time Inverse Fourier Transform is used to synthesize each N^{th} harmonic:

$$X_n(t) = X_n^{\Re e} + iX_n^{\Im m} = \sum_{i=0}^{7} V_i(t)\, e^{-jn\theta_i} = \sum_{i=0}^{7} V_i(t)\, e^{-\frac{jn2\pi i}{8}}$$

Here the harmonics (X) are composed from each velocity measurement (V) within an 8-sensor system. These coefficients are used by the control law to generate the desired actuation coefficients (U) required to stabilize the perturbations (here represented in the Laplace domain):

$$U_n(s) = G(s)X_n(s)$$

This set of harmonics is then decomposed with the DFT

$$B_i(t) = \tfrac{1}{12} U_n(t)\, e^{\frac{jn2\pi i}{12}}$$

where B_i represents the desired deflection of the i^{th} actuator.

With full identification tests, in which actuator effects and sensor-actuator delay are modelled, the final control system state equation has been identified as given below:

$$\dot{X}_n = \begin{bmatrix} \dot{X}_n^{\Re e} \\ \dot{X}_n^{\Im m} \end{bmatrix} = \begin{bmatrix} \sigma_{rs} & -\omega_{rs} \\ \omega_{rs} & \sigma_{rs} \end{bmatrix} \begin{bmatrix} X_n^{\Re e} \\ X_n^{\Im m} \end{bmatrix} + \begin{bmatrix} b_r & -b_i \\ b_i & b_r \end{bmatrix} \begin{bmatrix} u_{\Re e} \\ u_{\Im m} \end{bmatrix} + \begin{bmatrix} 0 & -g_i \\ g_i & 0 \end{bmatrix} \begin{bmatrix} \dot{u}_{\Re e} \\ \dot{u}_{\Im m} \end{bmatrix}$$

or

$$\dot{x} = Ax + Bu + F\dot{u}$$

Again, **u** represents the spatial harmonics of the blade deflection angles; their effect on the system is specified by b_r, b_i and g_i. This state space model is especially difficult to employ in a control system due to the input derivative terms.

The controller G(s) is a simply designed 2x2 feedback gain matrix, which essentially shifts the disturbance wave in phase so that the motors can "launch" a dampening [illegible]. Using both the first and second harmonics for feed control at a rate of 500 Hz, the controller has been proven to improve the compressor performance by 20% in mass flow reduction (see Figure 3-1), an impressive improvement in compressor efficiency.

3.3 Observations on Jet Engine Failures

Failures of components like electrical DC motors and hot wire anemometers have been extensively documented and, though the data processor clearly accounts for most engine subsystem failures, other fault events which may affect a rotating stall controller are certainly not unheard of. One important issue concerns the dependency certain components have on the whole system. A failure in the hydraulic supply may affect several (or all) of its manipulated components, some of which may be hydraulic actuators used in a high-speed compressor controller. Battle damage as well can have sudden drastic effects on a number of sensor or actuators simultaneously. Most conventional fault tolerant design considers only single failures, primarily because simultaneous failure modes in computers occur at such a low probability to justify ignoring them (in most cases). However, distributed measurement systems in a jet engine compressor do not fail like computers do. Multiple simultaneous failures must be addressed in some fashion.

Any of the compressor failure modes can drastically affect the controller's estimate [illegible] internal dynamics of the compressor. Pegged failures are generally the most [illegible] the harmonic magnitude estimates; here their effects are simulated in [illegible]ponent 4. As Figure 3-4 indicates, the controller transforms the erroneous flowfield (dashed line) to its Fourier harmonic estimate (represented, after a subsequent IDFT, by the dotted line) which departs considerably from the correct shape (solid line). Due to the error propagating through the DFT, the system believes the flow disturbances to be not only smaller in magnitude, but shifted in phase (here slightly to the right). This is an important point; should the errors be catastrophic enough, the estimated flow field may be so shifted in phase as to indicate flow perturbations in near-opposite directions.

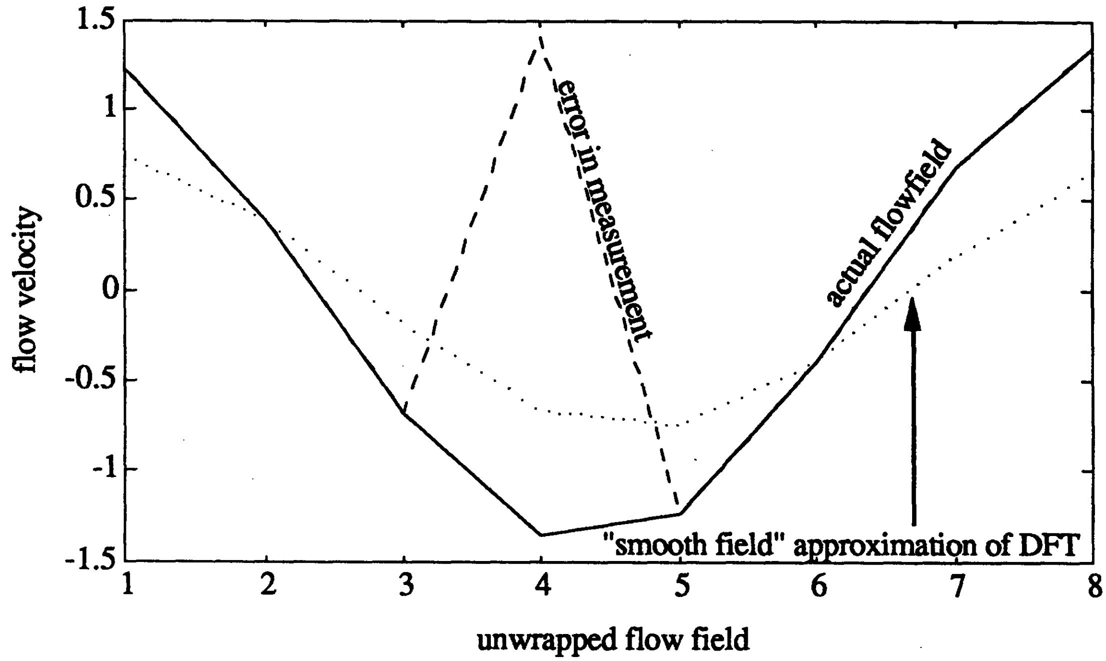


Figure 3-4 Actual (solid line), Incorrectly Measured (dashed), and Harmonically Estimated Flow Field (dotted) for 8 Sensor Measurements (simulated)

Pegged errors are not confined merely to the maximum perturbation velocity; the Gas Turbine Lab's compressor controller computes the perturbations from the time-averaged mean flow, which is usually around 20 meters per second. Sudden failures may cause the hot wire anemometers to read zero total velocity, translating into a huge perturbation velocity of -20 meters per second.

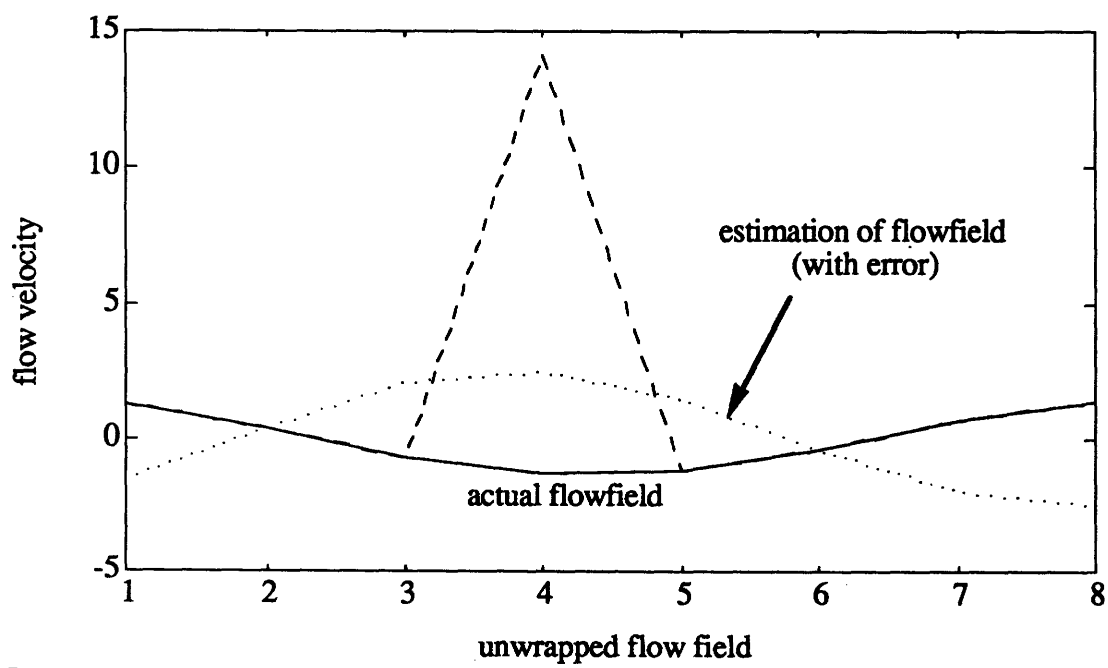


Figure 3-5 Actual(solid line), Incorrectly Measured (dashed), and Fourier Harmonic Flow Field (dotted) for 8 Sensor Measurements (simulated)

Figure 3-5 simulates just this type of fault: here sensor 4 has malfunctioned to read ten times the mean perturbation of the nonfaulty flow measurements. As is clearly seen, the harmonic estimate (dotted line) has almost precisely reversed from that actually occurring within the compressor due to the large magnitude error propagating through the DFT. This failure would cause the controller to degrade the performance of the compressor past the point were the controller not functioning at all, serving to compound the disturbance, acting in unison with the perturbations instead of damping them.

Chapter 4

Failure Detection Mechanisms

Perhaps the most interesting aspect of this report is the embrace of distributed sensing and actuation schemes in flexible detection mechanisms. Traditional analytical redundancy of lumped parameter systems might employ intrinsic dynamical information to form a knowledge base for comparison to one or more independent signals. In distributed systems, where 8, 12 or any number of components together compose a single "virtual sensor" or "virtual actuator," the approach must be fundamentally different. Failure detection of this kind can be modelled as a type of analytical redundancy, but is best described uniquely as what can be termed **expectancy modelling**. Identical sensors which combine to measure essentially one variable set can be cross-compared to elicit fault information. In this sense the cross-correlation of the distributed system is a resource which is shared among all components.

So far it has been established that specification of failure modes is dangerous in Byzantine FDI philosophy; however, several methodologies do exist which 1) do not require mode description and 2) have the ability to cross-compare information from distributed systems.

4.1 Actuator Response Modelling

Failures in the actuators are examined first as they are potentially the most easily detected. In the GTL compressor test stand, both the actuator inputs and outputs are available. If an accurate model exists for the instrument dynamics, failure detection could become trivial.

The transfer functions of DC motors used as the inlet guide vanes can easily be modelled as straight gains with delay from the input to the output. Figure 4-1 shows a typical blade output signal and its input command filtered with the following method developed by observation:

$$(B_{predicted})_n = 12.1\,(B_{cmd})_{n-2}$$

at each sample n. In other words, the blade commands B_{cmd} are delayed two samples and multiplied by 12.1.

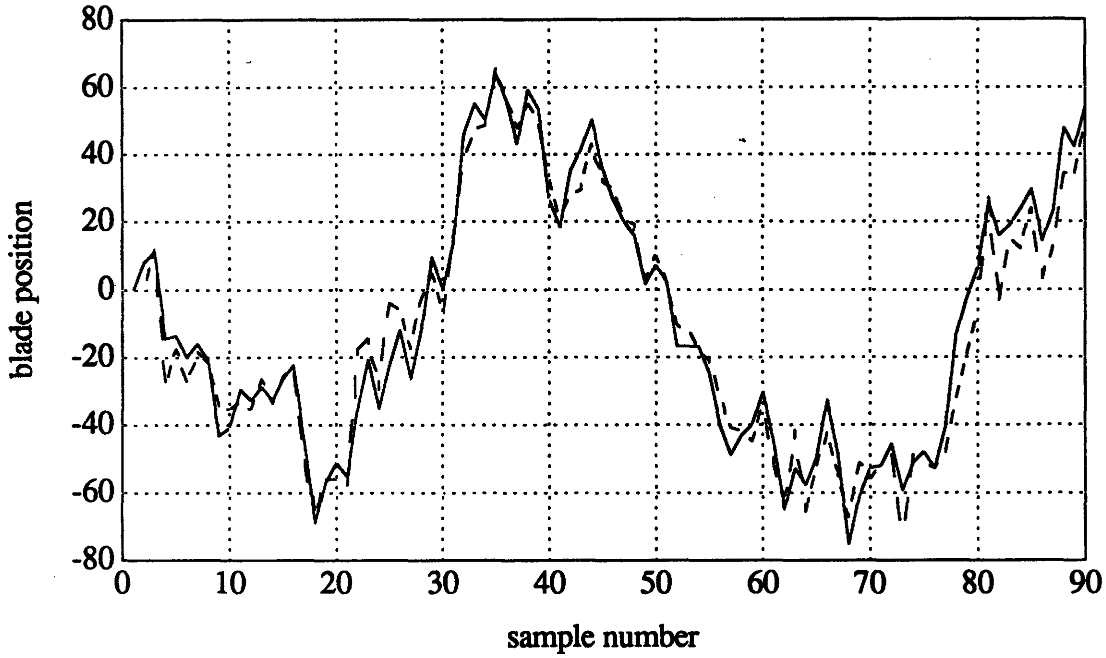


Figure 4-1 Modelled Blade Response vs. Actual

The predicted response from the transfer function, when compared to the actual response, looks very accurate. This evidence strongly indicates that faults manifesting even slight departures from this model producing errors equal to

$$\mathrm{ERROR}_n = (B_{actual})_n - 12.1(B_{cmd})_{n-2}$$

for each sample n can be identified with a probability approaching unity. With this in mind, the discussion hereto will focus primarily on sensor failure detection mechanisms.

4.2 Parity Relations

Parity relations rely on some form of linear dependence of the measurements of interest. Given that the dynamics of the system are such that any particular output y_j can be expressed as a linear combination of the other outputs,

$$\hat{y}_j(t) = \sum_{\substack{i \in I \\ i \neq j}} \alpha_i y_i(t)$$

each estimate of y_j can be compared with the actual output y_j to form error residuals indicating the relative suspicion of each sensor's measurement [Van84].

There are two basic approaches to implementing such relationships with distributed systems: spatial and chronological. Spatial parity tests examine all outputs simultaneously to determine faults, whereas chronological parity takes into account the time history of a single output to determine if it has failed. The detailed methods will not be discussed here as parity relations are classical techniques and need only be cited [Van84], [Schal91]. It will be useful, however, to show how they apply to distributed compressor systems.

4.2.1 Spatial Parity Tests

Spatial parity relations assume multiple sensor configurations simultaneously measuring (m) the same quantity (x):

$$
\begin{aligned}
&m_1 = h_1\underline{x} + \nu_1 \\
&m_2 = h_2\underline{x} + \nu_2 \\
&\vdots \\
&m_n = h_n\underline{x} + \nu_n \\
&\text{or } \underline{m} = H\underline{x} + \underline{\nu}
\end{aligned}
$$

where m_n is a column vector of n measurements, the h's are row vectors of length equal to the number of quantities of interest (x) and ν is the n-dimensional vector corresponding to the noise associated with each measurement. The parity relation

$$\underline{p} = V\underline{m}$$

is chosen so that p is independent of x and is a function of the measurement noise alone. A failure in instrument j will change the parity equation to

$$\underline{p} = V\underline{\nu} + V\underline{a}_j f$$

where a_j is called the **event vector** for instrument j, and is of unit length. A distributed system such as the rotating stall controller uses many more sensors (m) than states to be measured (x) and thus H will have rank much less than its number of rows. This technique has the potential advantage of using all of the sensor information at a single point in time to compute error estimates, and would utilize the full observation power of the coherent sensing system. Provided that the fault errors f are high enough, parity relations can identify erroneous behavior from the magnitude of P and isolate the source by comparing its direction to the event vectors a_j. With regard to actuators, though, this method is only viable if the actuator outputs are available and can be linked to a coherent command model.

In the distributed architecture at the Gas Turbine Lab, a spatial detection mechanism might be implemented, for example, by comparing individual measurements to the value expected at the sensor's location from the harmonic estimate. Error indications can be computed for each sensor:

$$p_i = m_i - \hat{m}_i = m_i - \tfrac{1}{8} X_n(t) e^{\frac{jn2\pi i}{8}}$$

...ificant deviations (such as those simulated in Figures 3-4 and 3-5) can be detected. ...nately, even the most primitive of spatial tests such as this were sufficient to ... any failure information $e_j f$ is completely lost in a sea of process noise (ν). The ...nce in sensor measurements from one sample to the next is so high, and the erroneous flow shapes so unpredictable, that even sudden pegged failures are difficult to identify. Any preliminary thoughts of using this method alone were quickly eliminated.

4.2.2 Chronological Parity Tests

Chronological parity tests, in contrast, traditionally use a discrete-time state-space model

$$x_{n+1} = \Phi x_n + \Gamma u_n$$
$$y_n = C x_n + D u_n$$

to generate a system function which approximates the present sampled output from successive matrix multiplications of past samples:[2]

[2] This equation follows from expanding the discrete-time state-space model.

$$(\hat{y}_i)_{n+k} = c'_i\Phi^k x_n + c'_i\Phi^{k-1}\Gamma u_n + \ldots + c'_i\Phi x_n + c'_i\Gamma u_{n+k-1} + d'_i u_{n+k}$$

where c_i and d_i are the i^{th} rows of C and D respectively, corresponding to the ith sensor output y_i, and k is the number of samples included in the estimation.

This method has the advantage of employing a "moving window" (of size k) of each component's past history to see sudden changes in operating point. Furthermore, this window could be varied in size so as to detect not only severe failures which happen quickly but also 1) soft failures which degrade slowly or 2) random failures which can wander roughly within the operating envelope but soon become obvious with time. Once again, the theoretical specifics are extensively documented in [Schal91] and will not be entertained here. It is clear from [Schal91], however, that this method is exhaustive and time-consuming in the number of matrix multiplications necessary to venture even a few samples to the past. In addition, chronology methods are extremely vulnerable to even small amounts of noise and modelling error, a liability which a compressor controller could simply not afford. This is apparent as successive exponentiation of matrices can polarize the condition (the ratio between the largest and smallest singular value) of the matrix to points where even the smallest of input errors can explode catastrophically.

4.2.3 Combining Philosophies

Clearly, the advantages of both methods must be applied in a single mechanism for compressor measurements which, by their nature, have significant noise levels and time-varying dynamics. This combinational approach, and its two respective mechanisms which will follow, comprise a large part of the distributed detection concepts introduced in this thesis and are considered especially well suited to noisy, distributed systems.

What is needed is a mechanism which, for each component, compares a "moving window" of its measurements to similar "moving windows" of other components' measurements. The spatial parity approach is illustrated in how these "moving windows" are compared and which components (if not all of them) are used. On the other hand, the method has strong chronological implications as well since a window of past data is summed together for each error point. With an intelligent choice of window size and spatial comparison, the problems which plague the individual systems above can be addressed in turn:

- Chronological parity is invoked in direct comparison of data streams **only** and any matrix multiplication or exponentiation is eliminated. No state-space modelling is included at all so no modelling errors from such a source can affect the system. Any modelling errors depend, of course, on the method of processing each data stream.

- Indications of failures are likely to be seen only in the standard deviation of errors from the predicted vs. actual calculations. Since spatial parity tests include inherent information of the compressor dynamics, however, sudden jumps in operating point can be masked, assuming that all components track them in the same manner (when functioning correctly).

Which of the distributed components are to be chosen, however, is critical to how the mechanism works. Either 1) some or all of the components spaced relatively evenly about the system can be selected, to glean diffuse information about global disturbances; or 2) only [illegible] the "nearby" (whose dynamics at any given time will be closest in shape) are [illegible]pare. A diffuse coverage may provide more gross information power, but a local approach is likely to be more accurate with intermittent disturbances which affect only sections of the whole distributed system. Clearly, it is a tradeoff to be tested.

4.3 Parity Detection Mechanism Design

After considering these issues and performing some preliminary computer testing using actual flow data, two mechanisms were identified as viable possibilities for a thorough failure analysis regimen. These tests were applied to sensor information from the hot wire anemometers only, and were not tested with actuator faults.

4.3.1 Local Expectancy

The first method, which shall be called **local expectancy**, considers only those sensors "near" the measurement of interest, as the name suggests. As Figure 3-2 implies, the disturbances which develop within the compressor track around its annulus quite coherently. Thus, each stall cell should be observed by consecutive sensors as it rotates around the circumference of the compressor. Comparison of one measurement with others shifted in time should elicit information as to how accurately each component is functioning. The number of adjacent sensors to be "polled" in this manner is yet another tradeoff to be studied. Clearly, single comparisons (see Figure 4-2) would be inadvisable in light of the ambiguity in determining which one had failed; successive adjacent comparisons would be necessary to establish the source of the errors.

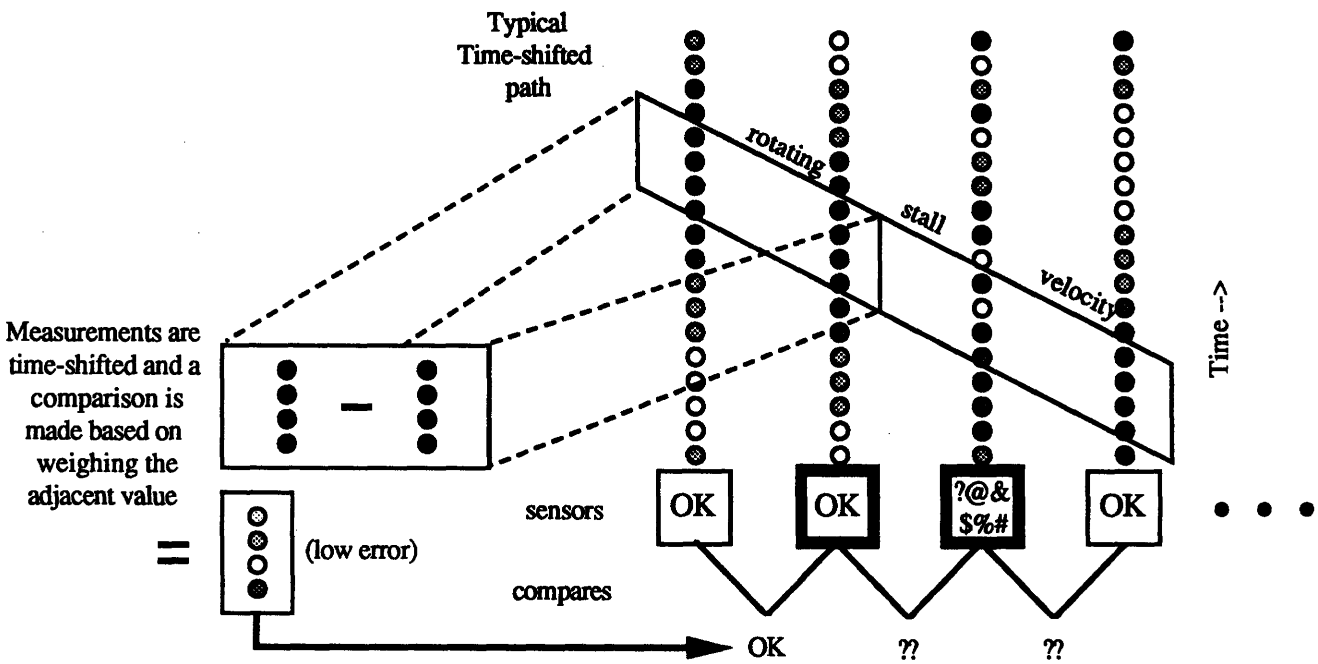


Figure 4-2 Single Comparison Schematic

It is obvious from the perspective in Figure 4-2 that a failed instrument has affected comparison tests two and three, and from the nature of the comparisons one easily concludes the failed sensor to be the third. But why not take more advantage of the distributed network and form a unique parity approach? Here lies one solution. By sampling several other sensors at a time in a single comparison, one can determine with more confidence whether the measurement of interest is erroneous (see Figure 4-3).

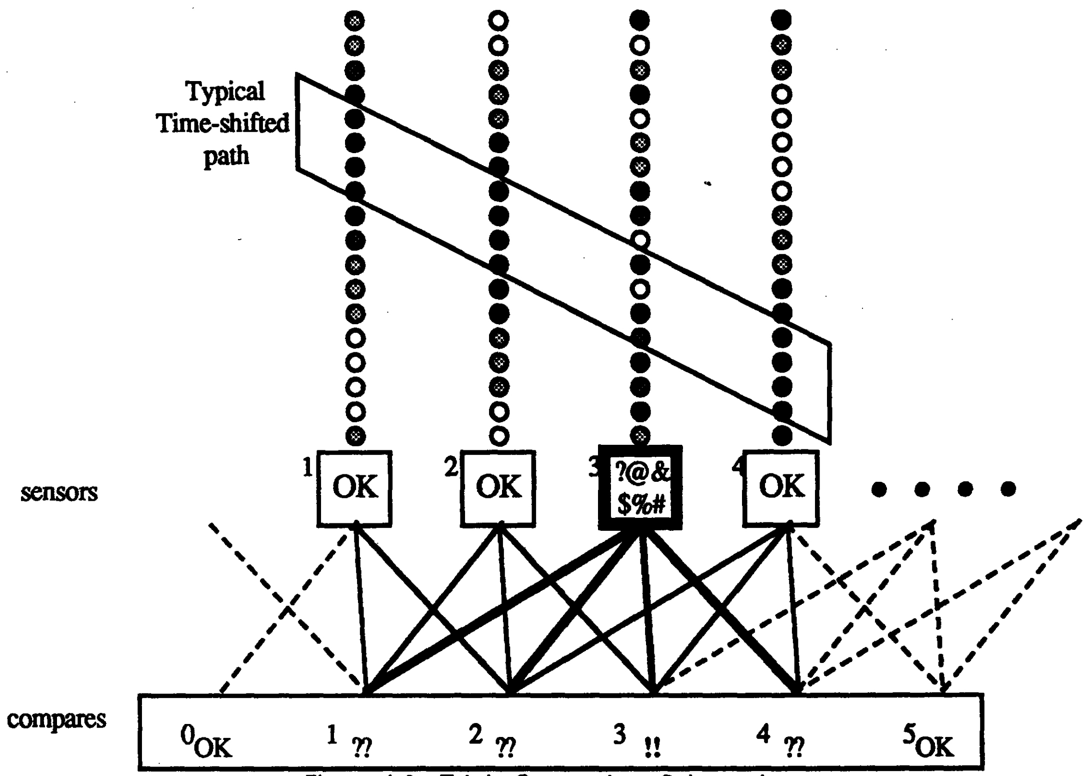


Figure 4-3 Triple Comparison Schematic

In this example each sensor output is compared to some average of those immediately adjacent to it and the component two spaces behind, forming the parity relation

$$(p_i)_n = (m_i)_n - \frac{(m_{i-1})_{n-V} + (m_{i-2})_{n-2V} + (m_{i+1})_{n+V}}{3}$$

for each sensor i and sample n, where V is the propagation velocity in # of samples per one-eighth circumference of the compressor.

A failure in sensor i = 3 would only partially corrupt other parity relations (here tests 1, 2, and 4), since the errors are averaged among two other measurements. The comparison t the failure site itself (site 3 in the figure) would be much more obvious; idence of adjacent test results of 1, 2, and 4 might point to a failure nearby.

is approach relies heavily on the dynamical nature of the compressor. Disturbances, though they can occur rapidly from a pilot's perspective, change their overall shape only slightly in the space of a few samples (for this system, 500 Hz). The greater number of sensors compared, the greater the time-shift required, and thus the more chance that the nature of the distortions have been changed. Also, the fault error is spread over many comparisons and may cause uncertainties when several parity tests show high errors.

On the other hand, few-element comparisons do not use the full power of the entire distributed network and may cause false alarms as stall cells enter their region of control. Figure 4-4 shows some of the relevant issues as the simulated error values (containing no noise or time-varying stall cells) change as a function of the number of sensors used in comparisons.

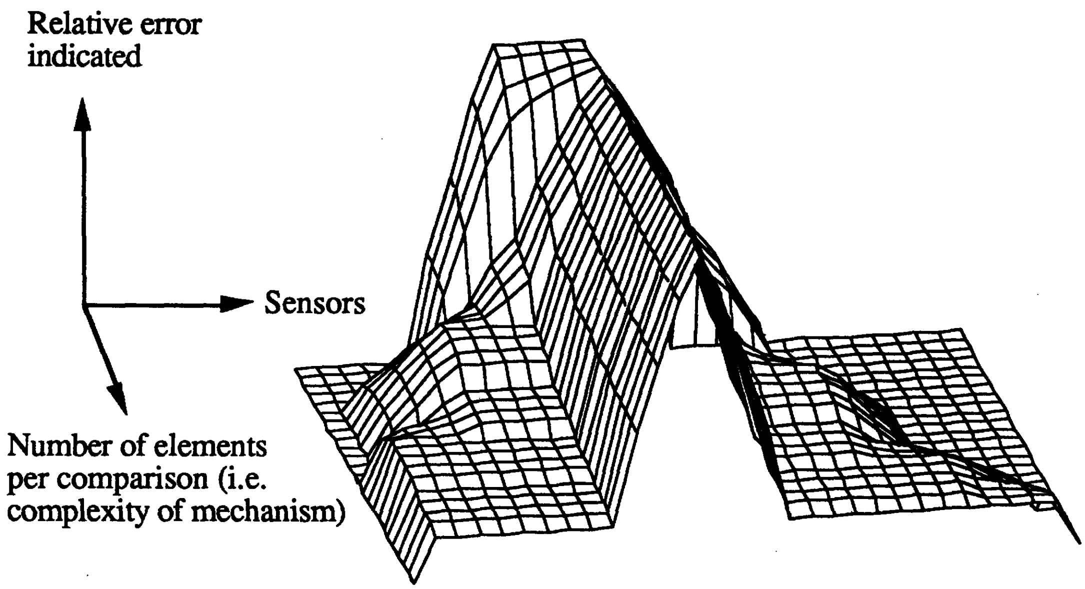


Figure 4-4 Map of Error Spike Clarity vs. Number of Sensors Polled

Here a sensor near the middle of the unwrapped distribution has failed and occupies the line (in depth) around which the ridge is centered. In the case of only single element-to-one-adjacent-element comparison (which occupies the rearmost horizontal) the error spike is spread out, but all other values are zero. Towards the front we see a more sharply defined spike as the error effects spread out into the other sensors. Variations in operating point around the compressor are more likely to degrade the performance of this parity test as more sensors are used. One can only imagine what this graph may look like with realistic compressor noise levels.

The compressor characteristic upon which this model relies is the rotation of traveling waves around the compressor, and their average speed must be ascertained to elicit the proper time-shift value from sensor to sensor. Previous experiments on the Gas Turbine Lab's low speed compressor indicated a rotating stall speed (σ_{rs} from the GTL compressor model) of about 67 rad per second, or 10.66 Hz, at the flow coefficient Φ=.397. This is about 27% of the rotor speed (the compressor is usually run at about 2700 rpm). At a controller sampling rate of 500 Hz, and with eight hot wires spaced around the compressor, the rotation speed corresponds to 500/(10.66 *8) = close to six samples from one sensor to the next. To confirm this, a sample of velocity measurements were shifted from one to twelve counts to determine the value which best "overlaps" the flow speedlines.

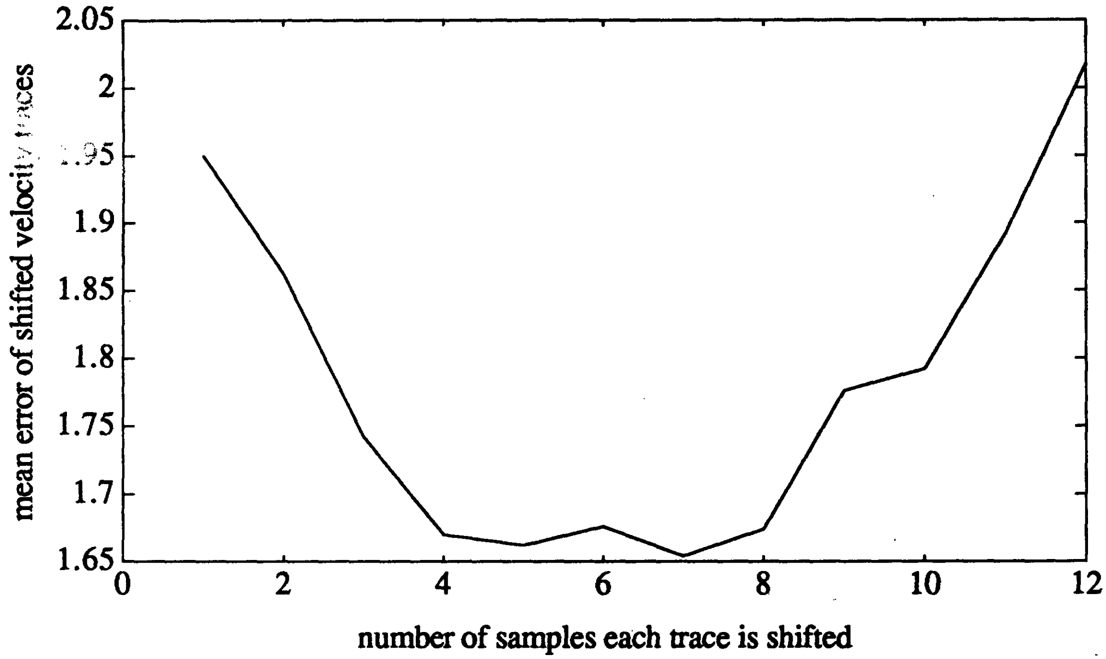


Figure 4-5 Overlap Error vs. Amount Shifted

Figure 4-5 confirms the rotation speed estimate as about 6 samples from one sensor to the next. Though the minimum error occurs at 7 samples, clearly the critical point of the curve is nearer to 6 samples. Slight differences in errors at 5, 6 and 7 samples are due most likely to noise or changing dynamics. To get a better feel for the time-shifting approach, one can apply the 6-sample shift to the same velocity data to see how the traces overlap.

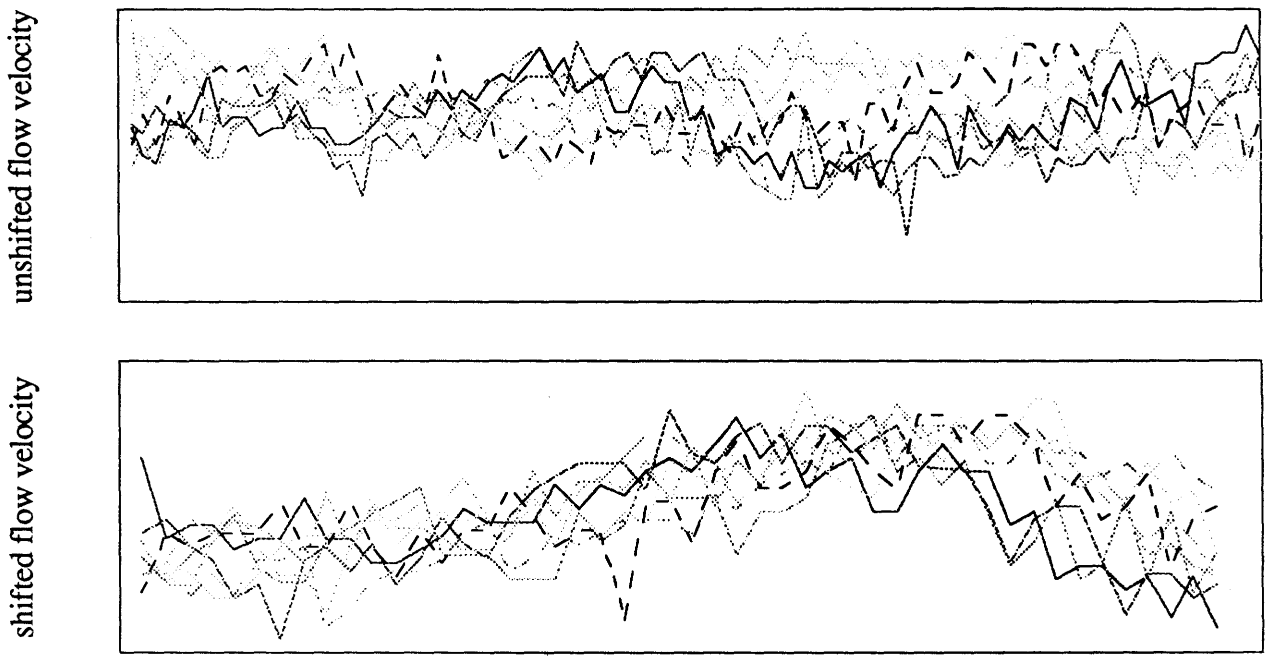


Figure 4-6 Shifted and Unshifted Flow Traces

Figure 4-6 shows the coherent shape produced from time-shifting the measurements, above, to produce the shape of the disturbance travelling around the compressor, below.

4.3.2 Diffuse Expectancy

The second method, which shall be referred to as **diffuse expectancy**, is derived from the harmonic nature of travelling waves in the compressor. Some, and not all, of the sensor measurements can be used to compose the Fourier coefficients:

$$X_n = \mathrm{DFT}\begin{pmatrix} m_a \\ m_b \\ m_c \\ \vdots \end{pmatrix} \text{ where } m_{a,\,b,\,c} \subset m_{1-8}$$

where X contains the real and imaginary components of the harmonic and are recomposed from a Discrete Fourier Transform (DFT) which has been adapted to compute the coefficients with partial information only. The choices a, b, c ... depend on what distribution of samples is desired, and would generally exclude the failure site of interest for the comparison.

Recomposing Fourier coefficients with partial information is a very easy matter. While typical DFT matrices are designed for even distributions, their design stems from an elegant theory which is valid for any distribution:

$$\begin{Bmatrix} x_{\Re e} \\ x_{\Im m} \end{Bmatrix} = 2\begin{bmatrix} \Sigma\cos^2(wt) & \Sigma\cos(wt)\sin(wt) \\ \Sigma\sin(wt)\cos(wt) & \Sigma\sin^2(wt) \end{bmatrix}^{-1}\begin{bmatrix} \cos(wt) \\ -\sin(wt) \end{bmatrix}\overline{m}$$

where w is the frequency of the harmonic being modelled and t spaced between 0 and Π according to the distribution.

For the first mode, these operations can be applied to a maximum of five (out of eight) measurements for each sensor; two on either side of it, the sensor directly opposite, and the two on either side of that. The remaining three are inaccessible, one being at the fault site itself, and the other two being harmonically orthogonal to it (and therefore providing no coefficient information). After the X values have been recomposed from the partial information, they are then decomposed back to the actual value expected at the ... in the parity relation:

$$p_i = m_i - \hat{m}_i = m_i - \tfrac{1}{8}X_n(t)e^{\frac{jn2\pi i}{8}}$$

for each ith sensor, computed at each sample.[3] Xn is computed through simple matrix multiplications as described above, so the parity relation still maintains its basic form.

Here the "moving window" theory asserts itself once again, in the form of the standard deviation of error between the past "band" of predicted measurements and the corresponding "band" of actual data:

$$\text{error}_i(n) = \text{STD}\left(\sum_{k=(n-\text{band\_size})}^{n} p_i(k)\right)$$

The use of only several measurements is the key approach to diffuse mechanism design. Failures will affect only some of the comparisons; intelligent examination of rising errors can, much like local parity approaches, lead to easy isolations. In addition,

[3] This parity relation is identical to that suggested during the discussion of spatial parity mechanisms.

this process may appear to involve a great deal of matrix manipulation, but the altered DFT and IDFT matrices are fixed and entail only ninety-six multiplications for an entire round of diffuse expectancy comparisons, a paltry figure as far as current VLSI circuit speeds are concerned.[4]

With a smooth single mode sinusoid, the Fourier harmonic can always be recovered exactly, provided there are at least three measurements. Under realistic conditions where process noise becomes a factor, however, a thickly distributed system is required. Exactly how many sensors are necessary (or, conversely, how many failures can be tolerated) depends on the accuracy requirements. As the number of sensors increases, inadvertent Fourier smoothing becomes less of a concern. One can, however, quantify the effect of noise on harmonic recomposition:

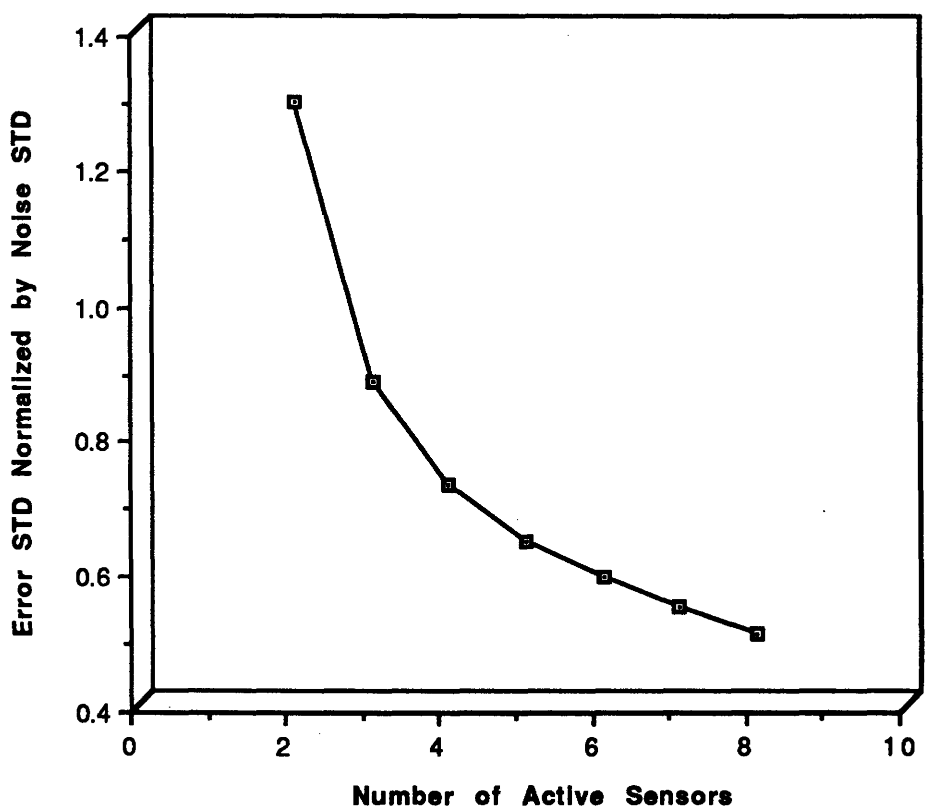


Figure 4-7 Scaling of Noise Magnitude with Reduced-Point DFT's

[4]At 10 MFLOPS, 12 multiplications would require approximately 10 microseconds.

When all sensors are functioning, any random noise is equally distributed into error for the real and imaginary parts of each harmonic, i.e. each component experiences noise whose standard deviation is half that of the total noise in the flow field. Figure 4-7 shows how the same noise affects the system as it experiences subsequent sensor degradations and is forced to compute the same coefficients with less data. Clearly the noise is scaled upwards with fewer sensors. The error due to noise seems to scale roughly as the inverse of the number of functioning sensors:

$$\sigma^2_{error} \approx \sigma^2_{noise}\left(\frac{2}{n_{sensors\_active}}\right)$$

Where σ represents the standard deviation. At low $n_{sensors}$ (<4) spatial aliasing, and not random noise, begins to dominate the recomposition error and as a result the Fourier harmonics become virtually impossible to compute.

4.2.? Observations on Parity Relation Distinctions

The **local parity** mechanism relies more heavily on the chronological approach, whereas the second examines the spatial ramifications of compressor activity. However, they both include some of the strengths of each.

Conventional approaches to sequential data testing have two major difficulties: 1) backtracking in space and time results in high complexity, and 2) tests generated neglecting circuit delays cause races and hazards even under fault-free conditions [Che88]. There is some argument to reducing the number of operations in computing the "moving window" errors. The standard deviation σ of a vector z is expressed as

$$\mathrm{STD}(z) \equiv \sigma_z = \frac{\sqrt{\mathrm{sum}(\mathrm{abs}(z - \frac{\mathrm{sum}(z)}{b}))^2}}{\sqrt{b-1}}$$

where b is the number of elements in z (the "moving window" width). To compute this at each sample for eight sensors can be tedious and unnecessary. Essentially, the standard deviation normalizes the vector, takes the square of the absolute value, and sums it. The denominator is unnecessary for a fixed window; furthermore, no normalization should be done since we are interested in the gross magnitude of the error; one only need take the square of the absolute value. This requires, in itself, only one update for every cycle:

$$\mathrm{sum}(\mathrm{abs}(z_{i \to j}))^2 = \mathrm{sum}(\mathrm{abs}(z_{(i-1) \to (j-1)}))^2 + \frac{|z_j|^2 - |z_{i-1}|^2}{2}$$

where the moving window begins at i and ends at j. This operation requires only three multiplications per sensor.

4.4 The Failure Detection Filter

Often mentioned with parity relations in the description of classical fault detection methods is the failure detection filter. The detection filter, which was first proposed in [Bea71], relies on a linear dynamic model of the system and compares the model's predictions to the actual performance. Figure 4-8 illustrates the block diagram of this arrangement.

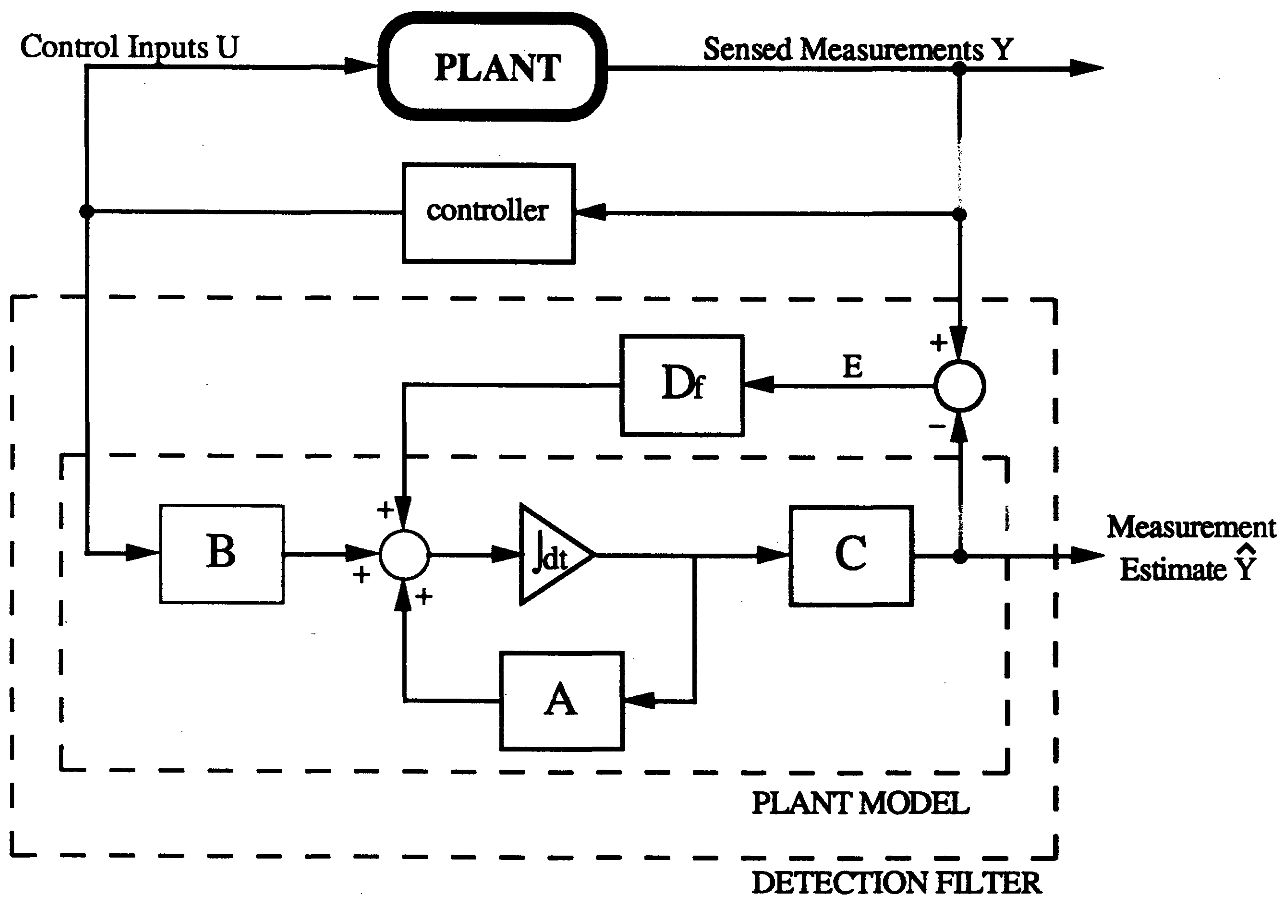


Figure 4-8 Failure Detection Filter Block Diagram

The detection filter thus has the following state space representation

$$\dot{\hat{x}} = A\hat{x} + Bu + D_f(y - \hat{y}) = (A - D_fC)\hat{x} + [D_f \quad B]\begin{bmatrix} y \\ u \end{bmatrix}$$

$$\hat{y} = C\hat{x}$$

and is similar to Kalman and other linear filters in that the system is driven by the residual (E) between the predicted and actual outputs, fed through a gain matrix D_f which is chosen both to stabilize the system and elicit precise fault information. As long as the system remains at a nominal operating point, initial condition errors will die away and the filter will track the system behavior with an accuracy close to that of the model.

The approach of the detection filter, as opposed to Kalman and many other filters, r... primarily on the ability to decouple the predictions and actual measurements from ...r. When fault errors enter the system, the residuals do not propagate through the ...ad remain fixed in a direction particular to the failed component. Specific ...applications of the detection filter are numerous and are discussed in [Mes81], and [Mar85]. Only those issues which are relevant to the jet engine compressor will be addressed here.

4.4.1 Establishing the Detection Filter Model

There are several methods to implementing the detection filter as illustrated in Figure 4-8. Choosing an approach suited to this jet engine compressor is extraordinarily difficult for several reasons:

First, the nature of the distributed system entails composing eight measurements into one set of two values. The way these measurements are combined through the DFT will have a significant impact on distinguishing failure directions. The eight sensor DFT as described in section 3.2 generates the coefficients X from the eight measurements V through the following matrix:

$$\begin{bmatrix} X_n^{\Re e} \\ X_n^{\Im m} \end{bmatrix} = \begin{bmatrix} .250 & .177 & .000 & -.177 & -.250 & -.177 & .000 & .177 \\ .000 & .177 & .250 & .177 & .000 & -.177 & -.250 & -.177 \end{bmatrix} \overline{V}$$

Immediately it becomes clear that each measurement shares its direction of influence with one other (though its magnitude is opposite). Changes in the first measurement will have essentially the same effect on the Fourier harmonics as changes in the fifth, second like the sixth, and so on. This has important repercussions for failure detection. Since this matrix is also the event vector matrix through which failures propagate to the detection filter, errors in sensors 1 through 8 will effect the residuals in the following directions:

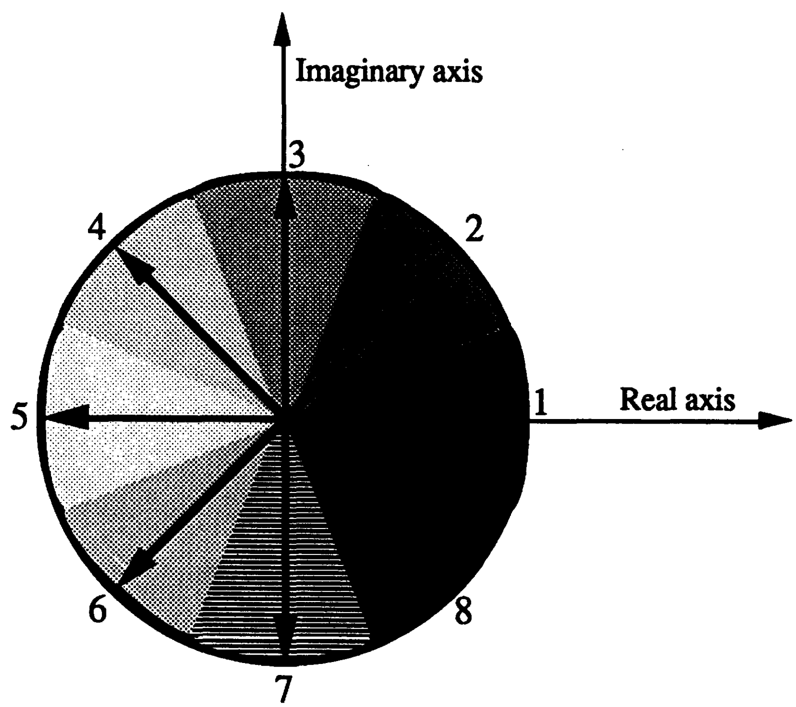


Figure 4-9 Directional Influence of Each Sensor (a graphical restatement of the event vector matrix)

Since no assumptions on failure magnitudes can be made, it may be that detection filters can only resolve fault errors to pairs of sensors only.

Second, the sensor inputs are not modelled in the state equation and cannot be held fixed in direction for the standard detection filter structure. Since the measurement vector Y is fed back within the detection filter through the term $Df(y - \hat{y})$, a failure of the j'th sensor, for example, alters the model by the amount

$$D_f e_j n(t)$$

where e_j is the event vector (one column of the event vector matrix) for the j'th sensor, and n(t) is an arbitrary time function. Consequently, the state equations for the general system become

$$\dot{\hat{x}}(t) = (A - D_f C)\hat{x}(t) + [D_f \quad B]\begin{bmatrix} y(t) \\ u(t) \end{bmatrix} - D_f e_j n(t)$$
$$\hat{y}(t) = C\hat{x}(t) + e_j n(t)$$

It is possible to choose D_f such that the contribution to the residuals is unidirectional, but this direction will not be the same as e_j. Therefore, the most this filter structure can do is constrain the residual generated by a sensor failure to the plane spanned by ej and $CD_f e_j$. This kind of behavior is exceedingly difficult to identify, especially when there are many signatures in the event vector space which must be distinguished for correct fault isolations.

Third, the compressor model as described in section 3.2 contains a term driven by the derivative of the input. This term could be modelled through a change in state variables:

$$z = x - Fu$$

which results in a system described by

$$\dot{z} = Az + (AF + B)u$$
$$y = z + Fu$$

removing the input derivative term.[5] However, this new model now includes a direct input-output term (Fu, second equation) which the detection filters were not intended to handle. This term may present difficulties and resist efforts to constrain failure signatures to single [illegible] ead occupying planes.

[5] Refer to section 3.2 for variable and matrix assignments. The actual measurements and control inputs (Y and U respectively) are, again, the IDFT of the Fourier harmonics listed here.

4.4.2 Adapting the Detection Filter to the Compressor Model

One solution to the last problem consists of a digital implementation of the detection filter as presented in [Mes81] and is especially suited to the compressor model since the input derivative can be "simulated" using both the current input and the previous input. The transformation to a discrete-time equation would be

$$\dot{x}(t) = Ax(t) + Bu(t) + F\dot{u}(t) \quad \Rightarrow \quad x_{n+1} = \Phi x_n + \Gamma_b u_n + \Gamma_f\left(\frac{u_n - u_{n-1}}{\Delta t}\right)$$

where Φ is the discrete matrix representation of A, and Γ_b and Γ_f are the corresponding analogs of B and F respectively. This model eliminates the addition of any input-output terms that would be necessary for a continuous-time system.

Choosing Df for this solution becomes a simple manner; since both state variables are fully measurable (by a C matrix which is identity) one can select a filter matrix which does not explicitly depend on the event vectors. The algorithm as presented in [Mes81] thus reduces to

$$D_f = \Phi - \lambda_d I$$

placing the eigenvalues of the detection filter at the discrete-time poles λ_d (which must, of course, be within the unit circle). This property is a useful one; a filter designed in this manner is suited to any application where sensor failures are the most important consideration.

As to the second dilemma, there exists at least one filter architecture which has the capacity to constrain single sensor failures to lines: segmented-measurement detection filters. This approach relies on dividing a distributed sensing system into two or more subsets, each of which is used to compute the flow harmonics which are then fed through individual detection filters. It has been shown that the Fourier coefficients can be calculated with limited information; in the same way, each filter generates an estimate of the entire compressor flow field from only part of the distributed system. The estimates from each filter can be cross-compared as illustrated in Figure 4-10.

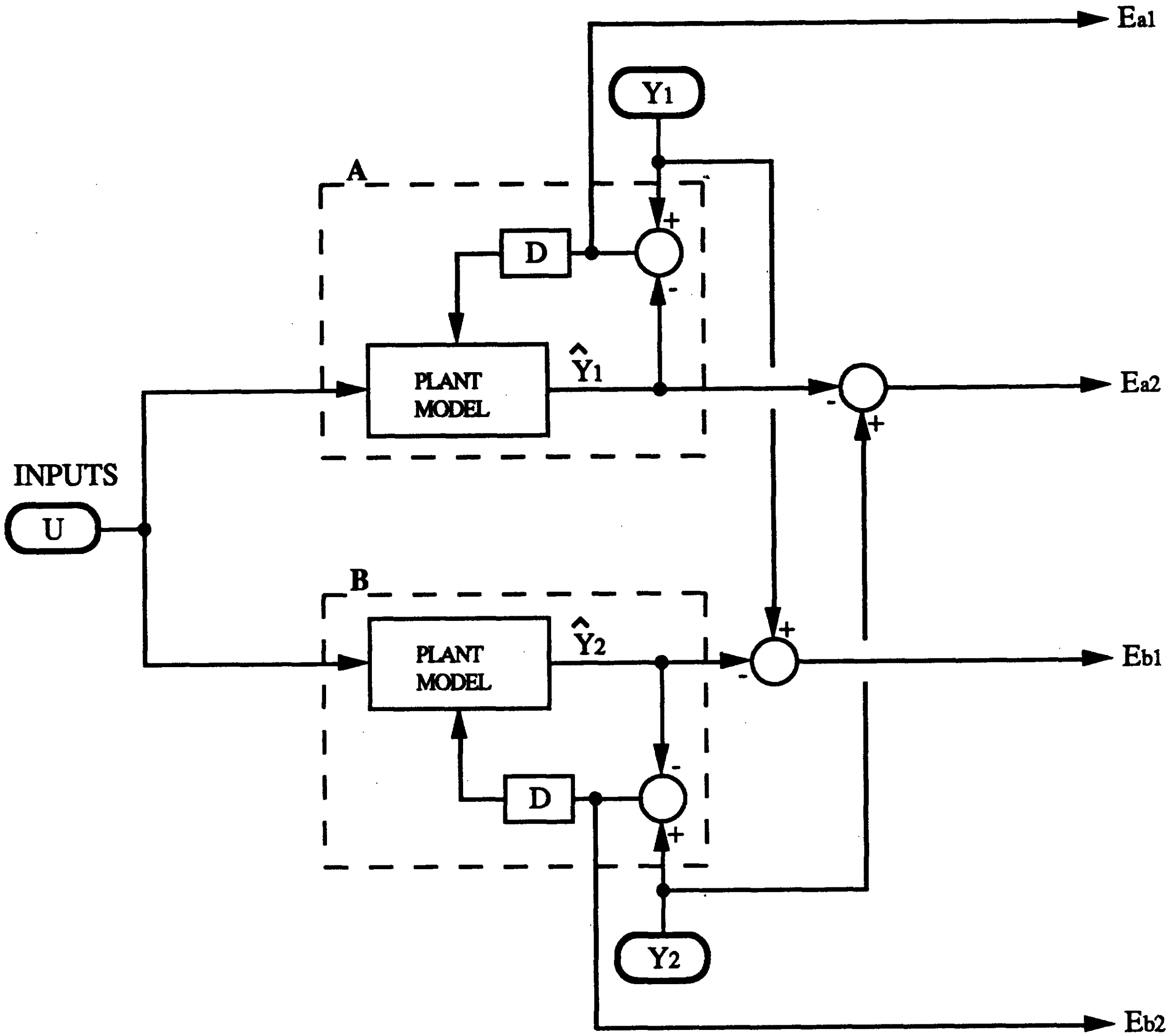


Figure 4-10 Segmented Failure Detection Filters

Here the actuator outputs U are sent to both filters A and B. Filter A uses the Fourier coefficients Y_1 computed from half of the distributed system, while filter B takes the coefficients Y2 computed from the remaining sensors. Both filters generate whole flowfield estimates $\hat{Y}$. Essentially, Y1, Y2, and both $\hat{Y}$ s all represent the same variables; the distinction lies in how each is computed, and more importantly, how each is affected by sensor failures. A failure in one of the sensors from which Y1 is computed will influence filter A only, and residuals EA1 and EA2 will immediately manifest errors. EB1 will show some error as well, as its estimate is compared with the faulty sensor subset; however, presuming the estimate of Y2 is reasonably accurate, the residual will remain fixed in a direction corresponding precisely with that sensor's event vector. Furthermore, EB2 will manifest no errors, further indicating that filter B is functioning normally. Thus the presence of a failure in subset 1 or 2 can be indicated by a rise in EA1 or EB2; while its direction is clear from EB1 or EA2 respectively. This is an important advantage: fault detection and isolation can be decoupled and need not be simultaneous.

This architecture also solves the event vector aliasing as described above quite easily. The allocation of sensors to set 1 and 2 can be conducted so that each pair of sensors that share event vectors are separated, one obvious choice being (refer to Figure 4-9) sensors 1 through 4 for filter A and 5 through 8 for filter B.

Also, this segmented approach need not be confined to two filters only; it may be useful to combine multiple filter sets, each allocating the distributed system differently in order to vote comparable outputs. Other allocation schemes will not guarantee event vector separability as the above design does; however, they do not need to if used only as a participant in isolation voting.

4.4.3 A Detection Filter Example

To better illustrate the segmented failure process, a failure of sensor 4 is discussed here with two segmented filter schemes: the first set satisfying event vector separability above (filter A: sensors 1 through 4; filter B: sensors 5 through 8) and the second an evenly distributed set (filter C: odd sensors; filter D: even sensors).[6] A failure in the fourth sensor influences filters A and D, and their residuals begin to rise while the model outputs of filters B and C show no deviation from the measurements. This implies that there is a failure in an even sensor from 1 to 4 (i.e. either #2 or #4).

Examining the cross residual EB1 from filter B (which should be constrained to the direction of sensor 4) at any particular time might show the following:

[6] It is interesting here that the first filter set seems to utilize local observations, while the second is more diffuse, very similar in theory to the parity test approaches.

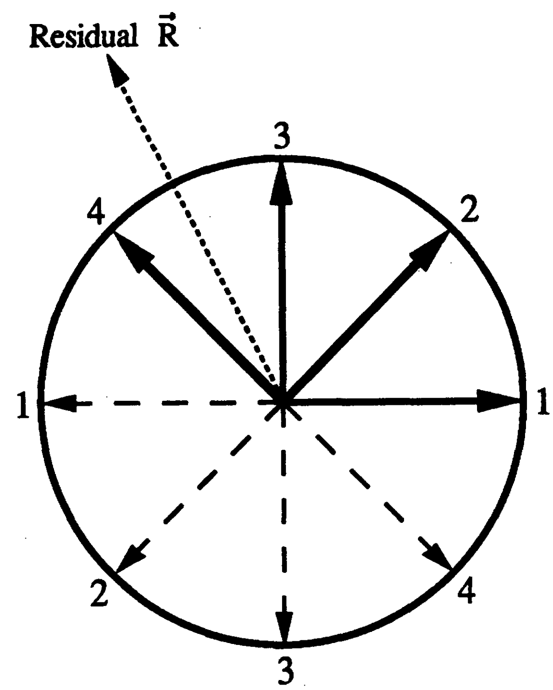


Figure 4-11 Sensors 1 through 4 Event Vectors and Residual Direction Example

where R represents the residual vector as compared to the four sensor event vectors. This residual has wandered far enough from event vector 4 to be relatively indistinguishable from a sensor 3 failure; however, since filters C and D express suspicion of even numbered sensors only, the detection mechanism can easily decide that instrument 4 is the faulty one. Furthermore, the residual EA2 from filter C (which should be constrained to the direction of sensor 4 -- and its aliased counterpart sensor 8) may also show the following:

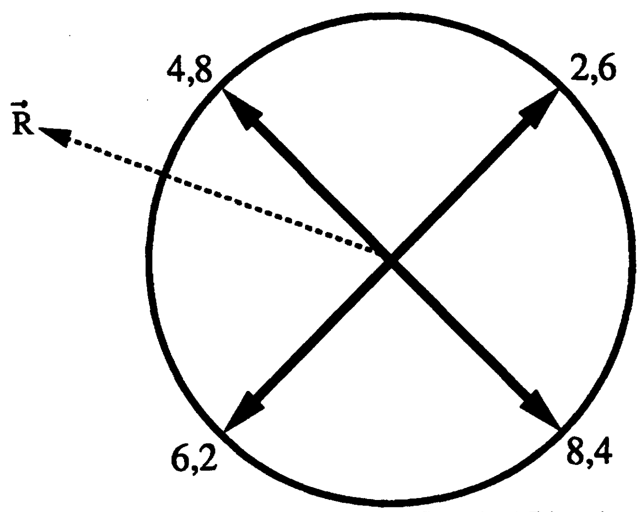


Figure 4-12 Even Sensor Event Vectors and Residual Direction Example

incriminating sensor 4 with even greater confidence.

The overall advantage of a segmented filter design is its capacity to restrict failures to small subsets of the distributed system, or fault containment regions; observing which filters are infected can replace virtually all of the exhaustive computation that would be required to examine residuals not confined to lines.

4.4.4 Identifying Failure Signatures

Some thought must be given to the specific procedure used to identify unidirectional signatures once a failure detection has been established from residuals EA1 or EB2. The relevant information from the compressor model and the detection filter can be separated essentially into three pieces:

- The magnitude of the appropriate residual vector (EA2 or EB1).
- The direction of the residual vector.
- The directions of each of the possible failure signatures.

All three of these variables have been illustrated in Figures 4-11 and 4-12. What is desired is a single number that measures both the severity of the failure and its proximity to a particular event vector. The dot product

$$\overline{R} \bullet e_j = |R|\,|e_j|\cos(\theta) = |R|\cos(\theta)$$

could be used directly; however, the cosine function is broadly peaked and is thus not very selective of residual direction. A more selective measure is $|R|\cos^N(\Theta)$ where N is some positive even integer that determines the selectivity of the process. Such functions shall be denoted as the "proximity signals," four of which (each corresponding to a particular sensor within a suspected subset) can be computed for each filter set in the example above. No threshold comparisons are necessary; should a component's failure signal be among the two highest in both the 1 to 4/5 to 8 and the odd/even filters, an isolation is asserted. The only process which requires comparison tests is the identification algorithm; it must compare the relevant residuals to each other and to a priori knowledge of "normal" versus "aberrant" levels. Here the same comparisons as those designed for the parity tests can be implemented.

4.4.5 Detection Filter Modelling Accuracy

All of the procedures identified so far depend closely on the accuracy of the compressor model as implemented in the detection filter; before continuing, it is necessary to ensure that the model is suitable for implementation in such a segmented filter regimen. The complete discrete-time detection filter equations are

$$\hat{x}_{n+1} = (\Phi - D_f)\hat{x}_n + \begin{bmatrix} D_f & \vdots & \Gamma_b + \dfrac{\Gamma_f}{\Delta t} & \vdots & -\dfrac{\Gamma_f}{\Delta t} \end{bmatrix} \begin{bmatrix} y_n \\ u_n \\ u_{n-1} \end{bmatrix}$$

$$\hat{y}_{n+1} = I\,\hat{x}_{n+1}$$

Figure 4-13 shows the model's performance with all sensors included in the Fourier recomposition (i.e. no segmentation) and detection filter poles at λ_d=.54. The actual coefficients are plotted as solid lines, the estimates as dotted ones.

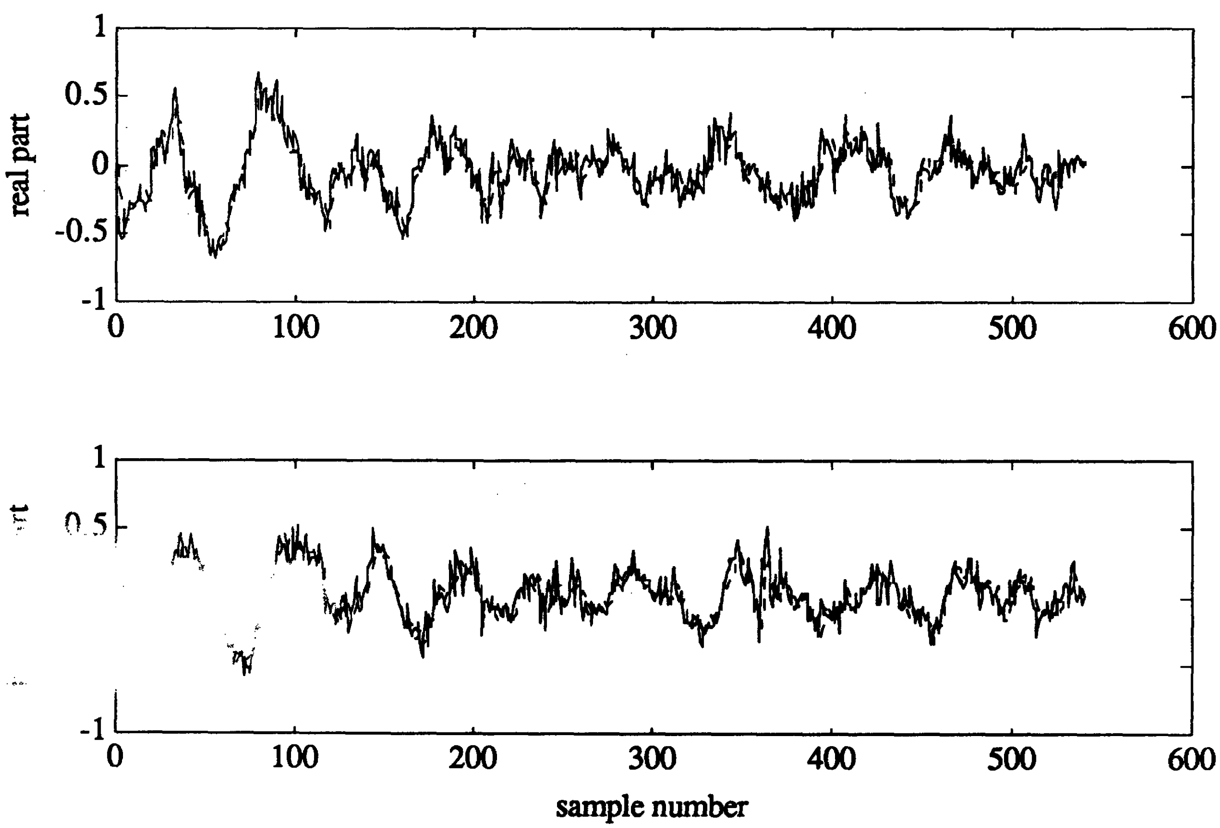


Figure 4-13 Actual vs. Predicted Flow Harmonics, No Segmentation

The performance is clearly exceptional; the dotted lines are barely visible.

The segmented detection filter response should also perform reasonably well for both 1to4/5to8 and odd/even allocations; this is shown in Figure 4-14 for detection filter poles at λ_d=.8.

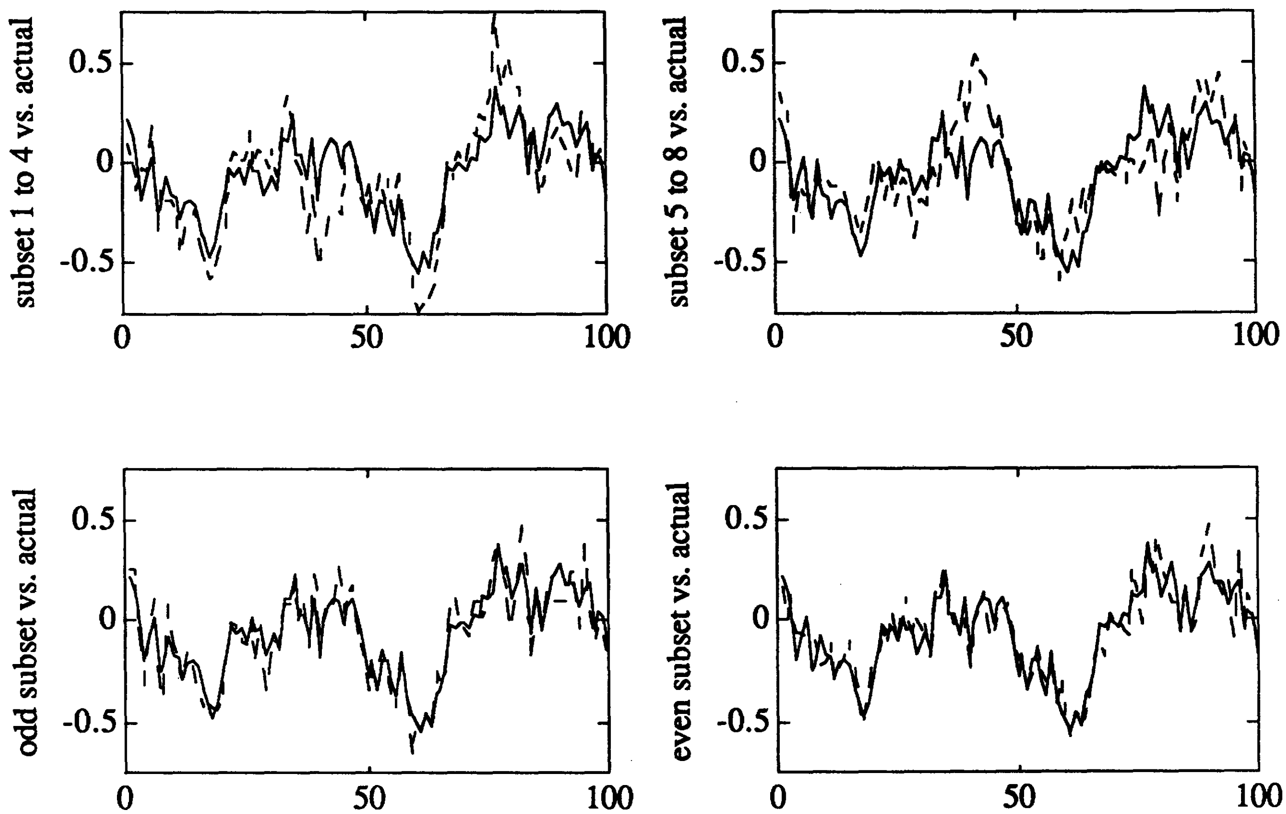


Figure 4-14 Actual vs. Predicted Flow Harmonics with Segmentation

The model response is not as good, especially for the local (1to4/5to8) allocation, but should hopefully be sufficient enough to elicit reasonable failure information. However, the detection filter poles must be chosen, at least initially, so that the r[illegible]tes the compressor as closely as possible (as opposed to an arbitrary choice above). Large poles will make the system too slow in reaction to the residuals, whereas small ones will produce a hyperactive system which reacts too fast. The following two figures illustrate each detection filter set's performance at various values of λ_d. The performance is measured as the standard deviation of the residuals normalized by the standard deviation of the measurements.

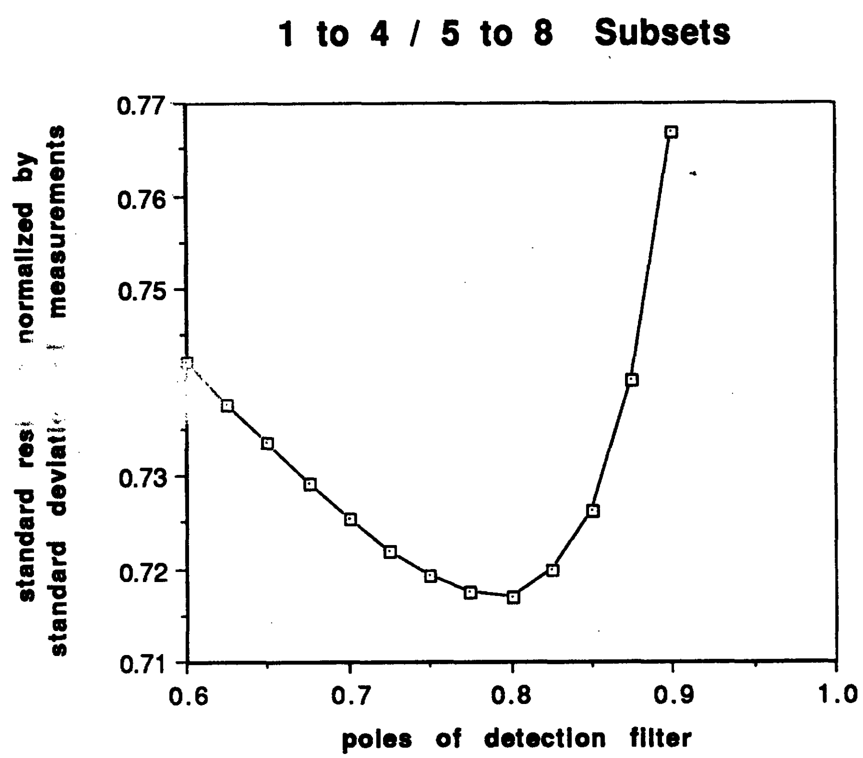


Figure 4-15 Model Performance as a Function of Filter Pole Placement, 1to4/5to8 allocation

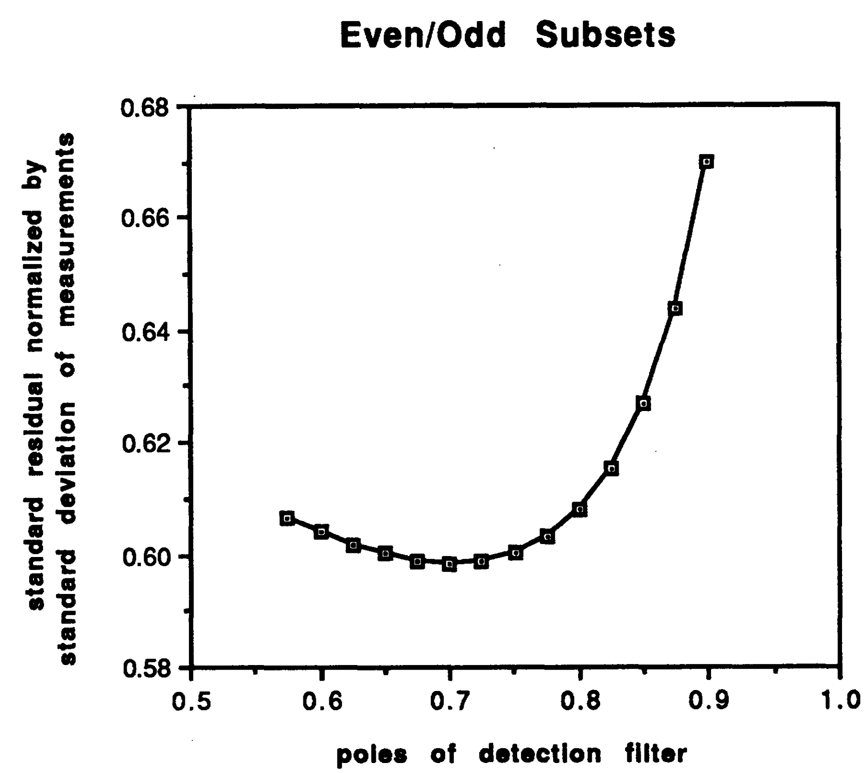


Figure 4-16 Model Performance as a Function of Filter Pole Placement, odd/even allocation

As indicated, the best pole choice for the first filter set seems to be at .8, whereas the second works best at .7. These discrete-time poles correspond to continuous-time eigenvalues at -180 and -110 respectively.

4.5 Power Spectral Density Relations

Power Spectral Density computation, which also uses the Fast Fourier Transform, is an extremely popular method of examining the full range of dynamics exhibited by complicated signals. It is most often used in identifying harmonic components buried in large amounts of background noise, and, at first glance seems ideally suited to identifying errors in velocity signals. However, PSD algorithms have their limitations as well. In the compressor,the power of any pre-stall waves is proportional to the power of the excitation, namely the amplitude and intermittency of the driving disturbances. PSD analysis can thus only elicit information about these two variables, and does not give direct, quantitative information about the system state itself. Also, few-point FFT's which only sample small fragments of a complex or changing signal can cause energy leakage from peaks into "sidelobes," distorting the spectral response. These limitations are particularly troublesome in the case of short data records or time-varying spectral content [Gar89], and is clearly an issue here as compressor dynamics change over time. Clearly, a tradeoff exists in selecting the size of data to be scanned by an FFT. The main advantage to PSD theory in general, however, is the ability to resolve sharp spectral features, even with short data records.

Briefly, the n-point Fast Fourier Transform works in the following way:

$$Y = \mathrm{FFT}(x) = F_n x \quad \text{where } (F_n)_{jk} = e^{2\pi i \frac{jk}{n}}$$

large FFT's can be successively "butterflied" into fewer-point FFT's as long as n remains even. The PSD of a signal with n values is

$$P_{yy} = Y^* \mathrm{conj}(Y)$$

where conj(Y) represents the conjugate of each value of Y.

Examining a typical 128-point PSD of first-mode controlled compressor velocity data produces the following spectrum[7]:

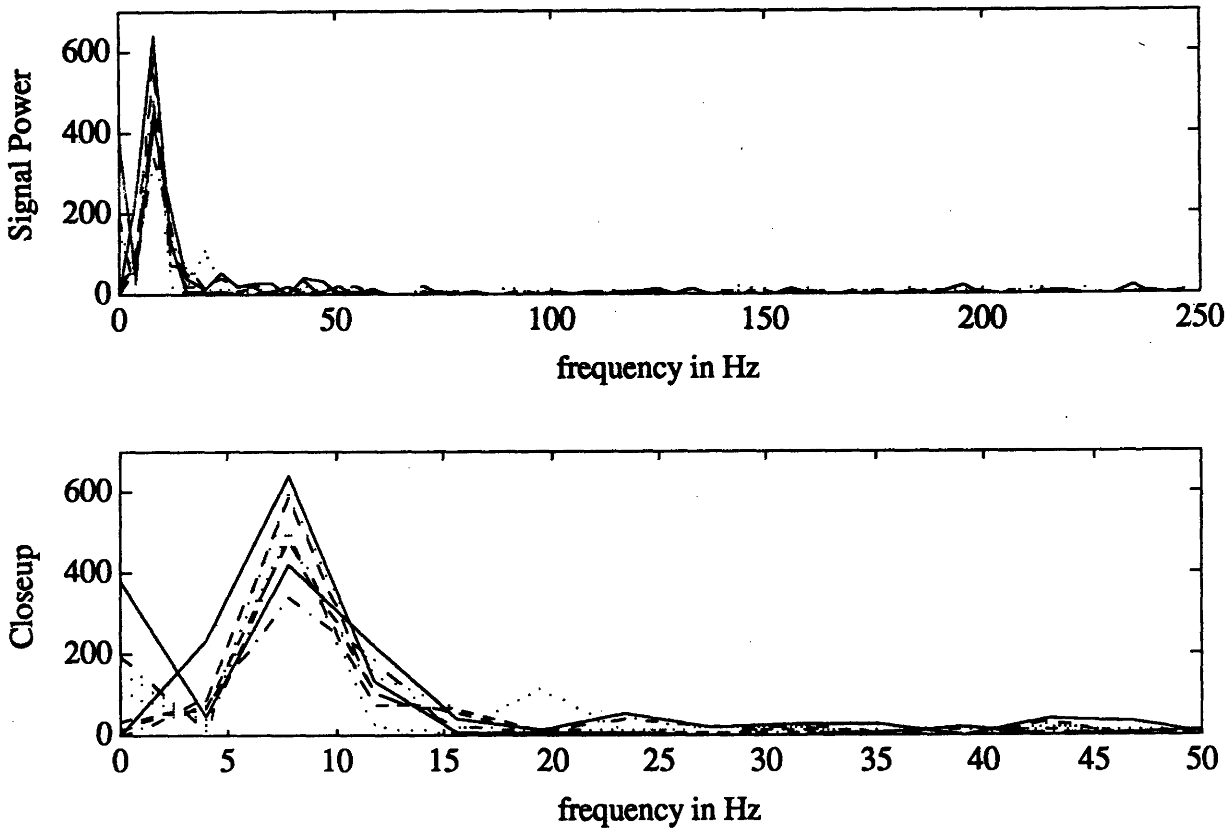


Figure 4-17 128-Point Power Spectral Density of Each Flow Trace (Example)

This plot compared favorably with higher-point (256- and 512-sample) FFT tests in identifying the salient characteristics of the velocity frequency distribution; however, fewer-point (64- and 32-sample) transforms quickly showed discrepancies in the estimated PSD and were not tested in fault simulation. Of particular (and virtually exclusive) interest is the well-defined peak close to 10 Hz corresponding to the rotating stall speed. This peak is not exactly at 10 Hz in this diagram; the exact velocity will tend to wander slightly, but is of [illegible]n in local parity tests as the movement is slight. As an example, random [illegible]s chaotic under PSD examination and distributes its power far more thinly than compressor disturbances (note the relative magnitudes of the vertical axes of Figures 4-17 and 4-18).

[7] A spectrum of this type will only be produced when there are disturbances in the compressor flow. Without these, the PSD looks relatively flat. Also note that modes higher than the first will show up as spikes at 20 Hz, 30 Hz, etc.

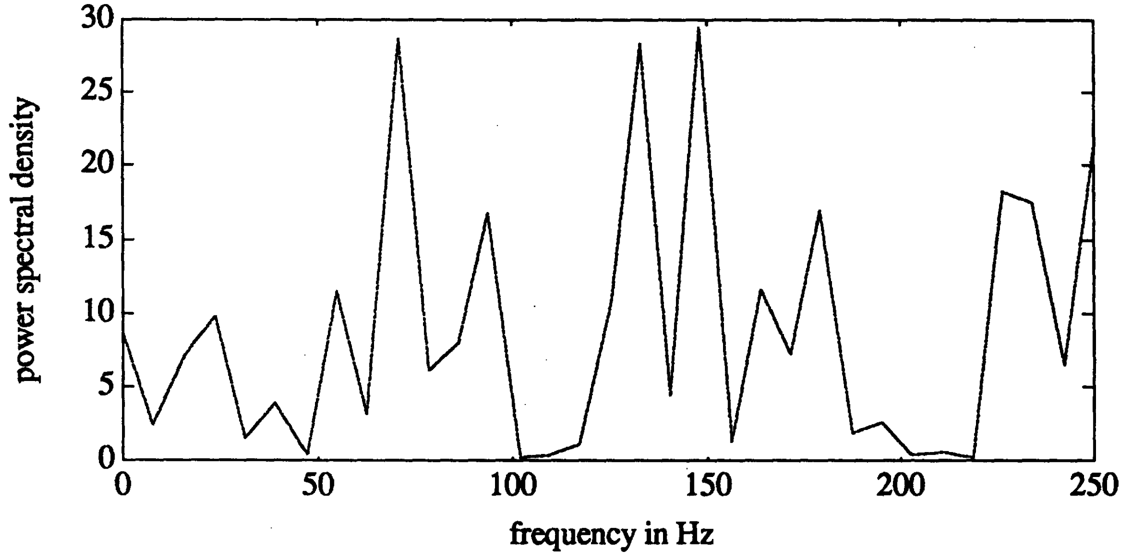


Figure 4-18 Power Spectral Density of Random Noise at Standard Deviation of Typical Flow Velocity

Pegged and zero values, of course, have zero signal power above their DC value at 0 Hz.

Since the FFT is expressed as a matrix, computing the signal power at a particular frequency involves selecting a single row of the FFT and computing its PSD. According to the PSD plots above, any signal which does not contain the "trademark" spike at 10 Hz, or whose spike is too large (perhaps due to a **calibration error**) can be easily identified.

This method, however, involves far more processing than those previously covered. 128 multiplications are required for determining the signal magnitude at 10 Hz; more are needed for error processing. But implementing high-point Fast Fourier Transforms is not necessarily an unrealistic or costly task; performing 128 multiplications quickly is an easy matter for VLSI chips, and Digital Signal Processors already have the capacity to execute FFT's at high speed. The Bendix custom DSP chip operates on a 11 bit word length and can perform a full 128-point transformation in under 145 microseconds [Hut86].

4.6 General Methods of Error Diagnosis

Interpreting noisy error signals which one is sure to see, even with "moving window" and standard deviation filtering can be the most time-consuming constituent of the failure detection and isolation process for the compressor controller. The probability of false alarms and missed detections, as well as that of correct isolations depends closely on the particular diagnostic process applied. Several approaches to error inspection must be weighed:

- **Awareness of the mechanism's fault signature**: after a failure, local parity tests will show increased errors in adjacent components; diffuse tests manifest errors similarly, though spaced around the compressor; failure detection filters will, of course, manifest errors in the direction of the failure event vector; while power spectral density methods and gain-delay actuator model comparisons are singular to each component. These signatures are often liabilities requiring conservative analysis so as to avoid confusion when multiple sensors are suspected. However, they can also be an advantage to a algorithm alert for these specific effects. Intelligent examination of five adjacent errors rising simultaneously in a local four-poll parity test might lead one to conclude easily that the central component is at fault where simpler diagnoses would be confused.

- **Accounting for sudden dynamic changes in operating point**: sudden jumps in several errors simultaneously should not trigger a false alarm. The possibility that these jumps could, by chance, behave like a particular failure signature as discussed above must be accounted for in qualifications such as duration and magnitude of the errors.

- **Awareness of the plant flight envelope:** near the stall line, failures may have to be identified much earlier than usual to avoid stall, while far away mechanisms may have the room to diagnose errors much more exhaustively. To what extent time response would affect the complexity of the diagnosis depends primarily on how well the mechanism can perform subject to the minimum time response as identified in section 5.1. Also, computing ratios can be dangerous during calm moments when the errors are very close to zero.

Based on the above issues, several simple approaches are suggested here in identifying failure signals correctly:

- Mean-value comparison: if the error is "well above" (the threshold to be defined for each circumstance) some average of errors, there is evidence of significant deviation. For local parity tests it may be necessary to average those component errors outside the polled region, should the error spike be too difficult to distinguish (refer to Figure 4-4); diffuse experiments may have similar difficulties as well.

- Relative-value comparison: the error is "well above" another selected error value -- a likely choice might be the next highest -- once again indicating that one single component is deviant.

- Absolute-value comparison: the error is above an absolute lower limit. For periods of very low disturbances and/or noise, when the errors are very low, this prevents any small spikes from causing false alarms.

These tests serve to establish an envelope that, when some or all of the thresholds are exceeded simultaneously, can be used to trigger a "warning function." This "warning function" is not necessarily intended to be used as a positive failure isolation, but instead as a less cluttered interpretation of the error values. It can be likened to warning the flight control system that something may be wrong and needs more detailed attention. It may be that these simple tests are not sufficient for single or even simultaneous failure errors; in that case more robust algorithms would need to be identified. In any case, the initial design for each algorithm is as follows:[8]

4.6.1 Parity and Actuator Model Test Warning Functions

The parity and actuator test warning functions are simple; the component suspicion is based primarily on which error is highest. A possible command structure for this is:

```
IF  (HIGHEST_ERROR>MV_THR*MEAN(ERRORS))&
    (HIGHEST_ERROR>RV_THR*2ND_HIGHEST)&
    (HIGHEST_ERROR>A_THR),
   ASSERT WARNING_FUNCTION;
```

where mv_thr, rv_thr and a_thr represent the mean-, relative- and absolute-value thresholds respectively.

---

[8] These algorithms were arrived at primarily through examination of the nature of the errors that each mechanism will produce. The experimental results which are discussed in chapter 5 did help to identify strong and weak points in each design, but was used almost exclusively to determine the specific threshold values for each warning function, and not its basic implementation.

4.6.2 Failure Detection Filter Warning Functions

The failure detection filter residuals are quite different in nature from the parity test errors and therefore require a different warning function structure. The full mechanism process is described here.

- Two sets of detection filters are designed (call them A, B and C, D) as described above; A and B use sensors 1-4/5-8 and C and D employ the odd/even ones.
- Four residuals are computed for each filter set. For AB, these residuals are(refer to Figure 4-10):

$$EA1 = Y_1 - \hat{Y}_1 \qquad EB1 = Y_1 - \hat{Y}_2$$
$$EA2 = Y_2 - \hat{Y}_1 \qquad EB2 = Y_2 - \hat{Y}_2$$

- For each filter set, residuals EA1 and EB2 are examined. If the magnitude of one is higher than the other by some <u>comparison ratio</u>, there is sufficient suspicion that the high [illegible] has been affected by a failure from its sensor subset. The exact ratio will determine the likelihood of identifying small failures or asserting false alarms. The magnitude of the high residual is also checked to make sure values near zero (i.e. when both filters are performing near-perfectly) do not trigger false alerts. For example, the commands

```
IF (NORM(EA1)>MV_THR*NORM(EB2))&(NORM(EA1)>A_THR),
  FAILURE_SIGNAL='A';
ELSE
  FAILURE_SIGNAL='NONE';
END
```

would be [illegible] and implementation to check for failures in filter A. Here mv_thr is the <u>comparison ratio</u> and a_thr ensures near-zero values are not mistaken for false alarms.

- When a failure signal has been asserted the filter must determine, if possible, which of the four sensors within the infected subset has failed by dot-product comparison. The residual which is constrained to linear signatures (for failures in filter A this would be residual EB1) is compared to the four possible failure directions to ascertain which are closest. The equation

```
PROXIMITY=((EB1/NORM(EB1))*EVENT_VECTOR).^N;
```

would be such an implementation. Here the dot product is computed by simple vector multiplication. Again, N represents to the selectivity of the proximity function.

- A single "warning function," much like those calculated for parity tests, is similarly useful here. The proximity values of the two sensors which fall within the suspected subsets of both filter schemes (i.e. 1 and 3, or 2 and 4, or 6 and 8, etc...) are examined. If one sensor produces high proximity values in both filter sets, the warning function is asserted. For example, if subset AB detects high residuals for sensors 1 through 4 (filter A), and subset CD detects similar results for the odd sensors (filter C), the commands

```
IF FAILURE_SIGNAL_AB(Q)=='A',
 IF FAILURE_SIGNAL_CD(Q)=='C',
  IF  (MAX(A_PROXIMITIES)->1)&(MAX(C_PROXIMITIES)->1),WARNING_FUNC=1;
  ELSEIF  (MAX(A_PROXIMITIES)->3)&(MAX(C_PROXIMITIES)->3),WARNING_FUNC=3;
  ELSE WARNING_FUNC=0;
```

would assert warning functions for the appropriate sensor 1 or 3.

4.6.3 Power Spectral Density Warning Functions

PSD tests have the task of detecting measurements whose frequency components are either much higher or lower than the others. The basic structure of the algorithm is similar to that of the parity tests; so too is the warning function generator. Its commands resemble the following:

```
IF      (ERROR_SMALLEST<ERROR_2NDSMALLEST-A_THR)&
        (ERROR_SMALLEST<MV_THR*MEAN(ERRORS)),
   ASSERT WARNING_FUNCTION(SMALLEST);
ELSEIF  (ERROR_LARGEST>ERROR_2NDLARGEST+A_THR)&
        (ERROR_LARGEST>MV_THR*MEAN(ERRORS)),
   ASSERT WARNING_FUNCTION(LARGEST);
```

where the thresholds establish a two-directional envelope that detects large and small errors. Here a_thr combines the relative- and absolute-value comparisons into one threshold.

4.6.4 Extension of Warning Functions to Simultaneous Failures

What of faults like hydraulic system failures, which may affect several high-speed engine actuators at once? Such simultaneous failures must be accounted for; however, the same principal methods can still be used, with the addition of a more developed warning function algorithm. Once again, the distributed nature of the actuation system serves to be an advantage: the warning function as designed for single failures can be modified into a distributed warning function routine which performs identical tests on several or all of the errors, examining each in turn with relative, mean-value and absolute comparisons.

The only difference in multiple warning function design resides in the choice of errors used in each comparison test. Clearly, for relative-value comparisons, the next highest value cannot be used since it may be evidence of another failure. The mechanism could be made to lift those errors which it views as "large" and examine each signature with respect to the rest of the "small" errors.

Such an algorithm is illustrated for a two-failure local parity tests:

```
IF  (ERROR(I)>MV_THR*MEAN(ALL ERRORS EXCEPT 2 HIGHEST))&
    (ERROR(I)>RV_THR1*ERROR_ABOVE)&
    (ERROR(I)>RV_THR2*3RD_HIGHEST)&
    (ERROR(I)>A_THR),
  ASSERT WARNING_FUNCTION(I);
```

The commands look complicated, but the approach is essentially identical. The top two error values are removed from the mean-value comparison to decouple it from failed signals; two relative-value thresholds are identified, one for the third highest (taking the place of the second highest in the 1-failure algorithm) and another for the value immediately above ERROR(I) to ensure that two signals -- ERROR(I) and the one immediately above it -- must be equally distanced from the background noise. In simpler terms, this algorithm triggers when two error signals of reasonable magnitude rise equally while all others remain low.

The extension to simultaneous-failure warning functions for diffuse parity and PSD methods is nearly identical. Failure detection filters, on the other hand, are complicated enough in a single-failure detection methodology, since residuals are so difficult to constrain to a line. It may be that many sets of detection filters each with different fault containment regions could detect multiple failures. However, such a design would be far more complex than any parity or PSD mechanism; and so, for the purposes of this thesis, parity and PSD detection methods must be relied on to identify simultaneous failures.

4.7 Summary

So far, several detection mechanisms have been proposed: **local expectancy, diffuse expectancy**, and **power spectral density examination** for sensors; and gain-delay model comparison for actuators. It has been shown that these methods may be able detect both single and simultaneous failures, manifesting arbitrary behavior; the mechanisms are designed to be flexible, and all that is required is a comprehensive diagnosis of the resulting error outputs. The possible advantages and limitations of each approach have been discussed, but all of that is clearly academic without thorough testing and experimentation to glean realistic results and conclusions.

All of the above algorithms were written as MATLAB subroutines on the MacIntosh IIcx and thoroughly tested by simulating failures in actual flow data. The relevant program code is included in Appendix A.

Chapter 5

Analysis and Results

5.1 Establishing Requirements

The discussion so far has centered primarily on general detection philosophies, rather than delineating specific requirements to be met. The most vital characteristic of each [illegible] mechanism for the compressor is its response time. Of utmost importance is how soon [illegible] system must react to correct failures before stall is unavoidable.

This response time requirement is distinct from the time between the point-of-fault-inception and the point-of-stall. After all, there is a certain level of flow distortion at which catastrophic compressor failure is irreversible. This critical point, after which stability cannot be recovered, is what must be determined. Existing time response information from compressor study has been determined through sudden and complete failures of the controller or by reducing the uncontrolled compressor's flow coefficient until stall occurs. It is unlikely that failures of single components in the compressor will affect the plant in the same way, as the system may retain some measure of influence on the dynamics. On the other hand, it has been shown that some failures may cause the controller to destabilize the system, which may produce more stringent time responses.

It is risky to quantify the "critical response time" as a measure of the maliciousness of failure; yet one must have some figures to work with. Simulating a catastrophic, or near-worst-case failure close to the closed-loop stall line may give some hint of the minimum response time demanded. Exactly this type of experiment was conducted on the GTL compressor with two purposes in mind: 1) testing several fault modes thought to be malicious to illuminate a near-worst-case failure; and 2) determining how long that failure must exist until stall is irrecoverable.

This qualification process is illustrated in Figure 5-1.

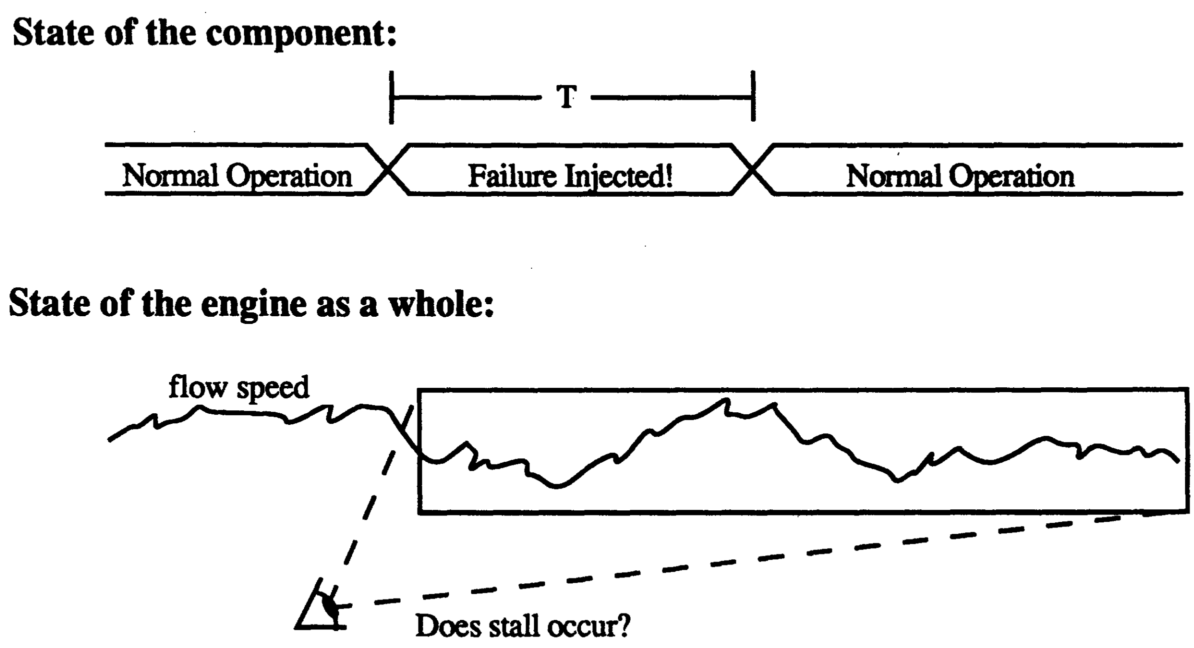


Figure 5-1 Description of Failure Injection Algorithm

First, the controller is allowed to reach steady operation. A particular sensor or actuator measurement is failed within the control software for a period of time T, and the compressor operating point is observed during and after the failure injection. Since the compressor's flow velocity is dynamic, each test is repeated many times to ensure that no chance combination of random fluctuation and failure compound to stall the compressor. Then, if no stall occurs, the failure duration T is increased and the experiment repeated. Response times are recorded when stall is observed or when it becomes obvious that the compressor remains stable under any failure duration.

Prior to approaching the compressor test stand, it soon became clear in computer simulations that large **pegged failures** were the most catastrophic to Fourier coefficient recomposition. These sudden deviations from the ambient to the maximum measurable flow velocity had the greatest impact on controller accuracy. With Byzantine philosophy in mind, however, the decision was made to test a number of failure modes of both sensors and actuators. The injected failures are described here:

pegged failures at the maximum possible values (as in Figure 3-5);

zero failures;

and **random noise failures** at standard deviations above the ambient flow perturbations.[9]

[9] The actuators are sensitive to sudden changes in command control (e.g. -15° to +15° in one sample) so the noise was low-pass filtered at 50Hz with a Chebyshev type I filter using the Direct Form II Transposed Standard Difference Equation.

The compressor was operated with first mode control as close to the stall line as possible; thus, any errors in the controller would have been most likely to throw the compressor into sudden stall. The flow coefficient at stall was measured at ϕ=.396,[10] and all of the failure tests were conducted as close to this point as possible, usually at ϕ=.410 (which would put the compressor well inside the "stall margin" region of safety, usually placed at 10-12% above the stall line).

Surprisingly, no single actuator failure of any kind for any duration was sufficient to stall the compressor. Figure 5-2 shows the test results from pegged sensor failures; stall was seen to occur somewhere between 250 and 330 samples (500-660 milliseconds) after injection.

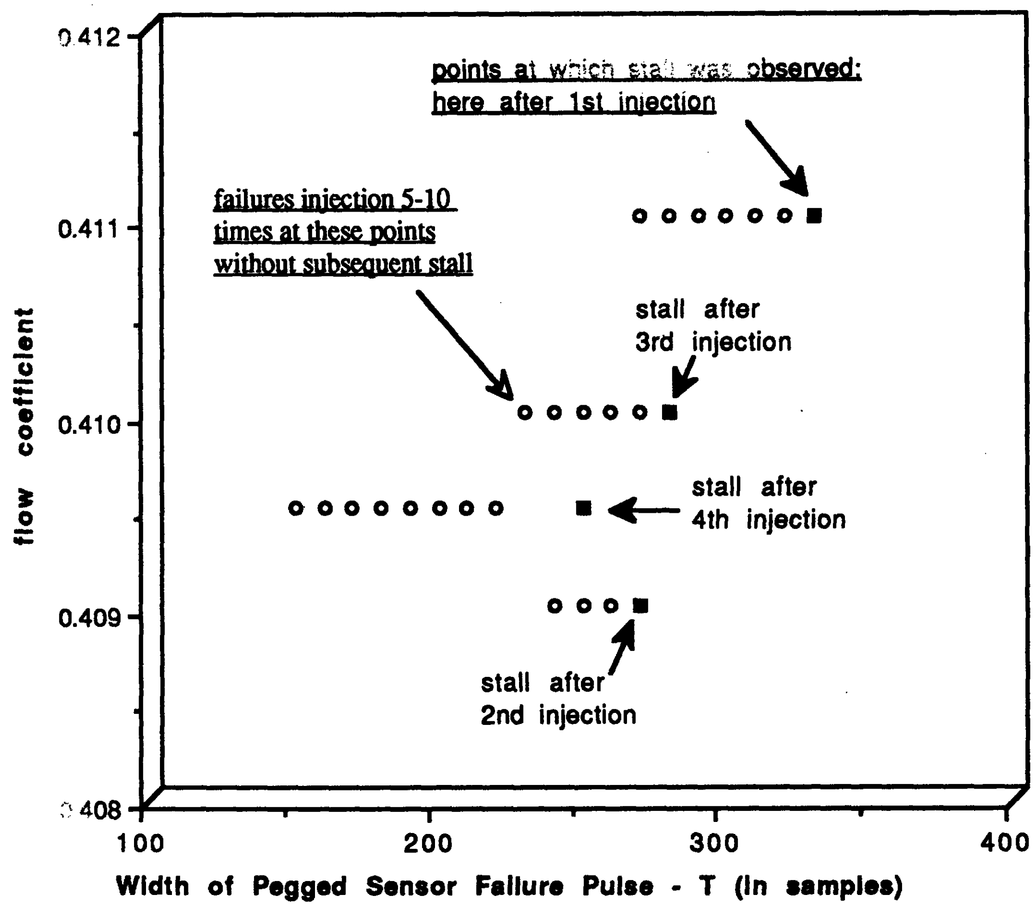


Figure 5-2 Failure Tests Performed with Various (Sensor) Failure Pulse Widths

---

[10] Note that this differs slightly from the first-mode stall coefficient stated earlier, as the engine will function differently each day, due in part to ambient temperature and pressure.

Already improvement in performance with distance from the stall line (upwards in flow coefficient) can be made out, though clearly the data pool is too small to be quantified.

As predicted, no other sensor failure mode was as damaging as the maximum pegged case. Small **pegged faults** at ± 5 meters per second induced stall after 20,500 samples during one experiment; thereafter, stall could not be induced. No stall was observed under any duration of faults generated as random processes at the same standard deviation of the ambient disturbance magnitude. With confidence, random errors of this kind can be eliminated as the worst of the tested failures. In like manner, **zero failures** induced no stall for any duration.

A 500 millisecond minimum response time puts the detection requirements well above the established 20 ms time-to-deep-stall which was identified with no control in [Moo88]. As a conservative estimate, 400 milliseconds or 200 samples shall be used as the base requirement.

If a detection mechanism can be found that will identify failures before this base requirement, one can qualify the mechanism as being at least 'promising' in a Byzantine Fault Resilient capacity. This conclusion follows from the basis upon which Byzantine Fault Resilience was defined in section 2.2: that no failure of any nature will affect the performance of the compressor, in this thesis defined as the point-of-stall.

5.2 Actuator Failure Detection

First the actuator state space model shall be examined, as it is potentially the most easily solved. With the prediction that the blade estimates are reasonably accurate, experiments are first performed with small moving windows on the order of 10-20 samples. Though at times the apparent deviations from the actual motor output can be significant, it is important to be sure that no alarms are triggered unless the errors are very large, or persist over an extended period of time. How much the error is to be weighed must be considered. First, a very soft failure is examined. In this experiment, the output of actuator 4 is shifted slightly (by 10% of the largest deflection in this run) at sample 200 to simulate a **bias error**. Figure 5-3 shows the relative deviation of each component from the gain-delay model, with a moving window size of 20 samples:

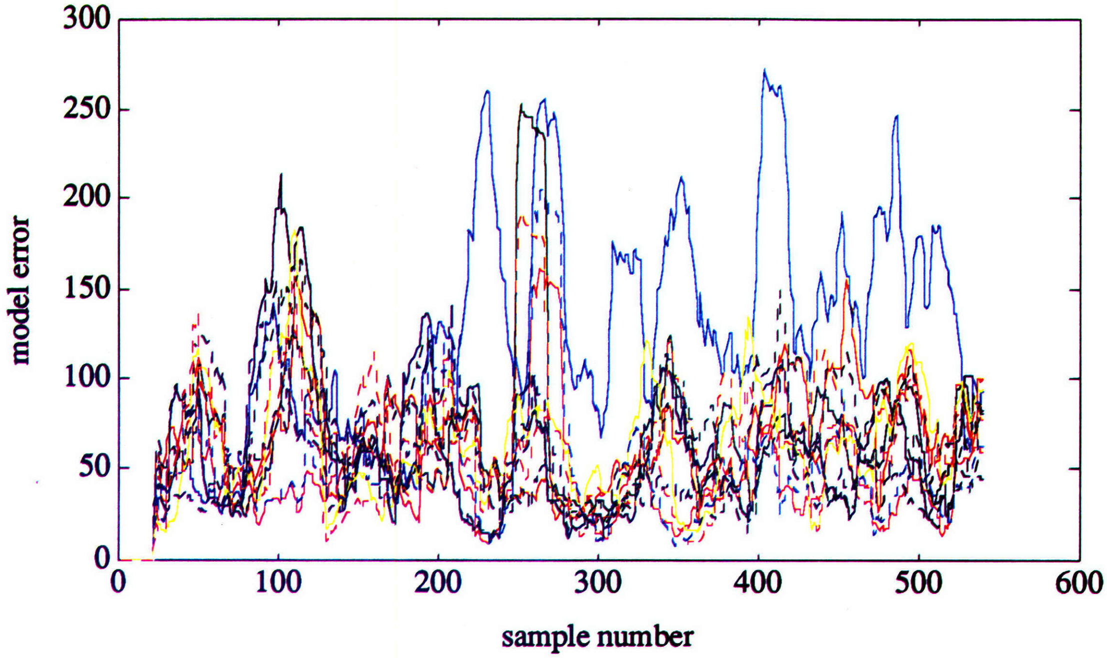


Figure 5-3 Soft Actuator Failure (Shifted Mode) Comparison Errors

Though the signals are quite noisy, it is quite clear among the spikes and jumps of the controller's normal operation that at 200 seconds (and beyond) one error signal (number 4, marked by the light blue line) stands out by itself. At approximately 260 seconds there are some large errors from another source, possibly rough command controls sent to the actuators, but in this case several errors rise at once, as opposed to one clear spike.

Some detailed thought must be given to choosing threshold values to minimize false alarms, missed detections, and the like. It is best to directly examine actual flow data to get the robust results. For the actuator failure simulated above, mean-value comparisons might view the velocity data as illustrated in Figure 5-4, here contrasting the fourth component's error with the overall average (of the other traces).

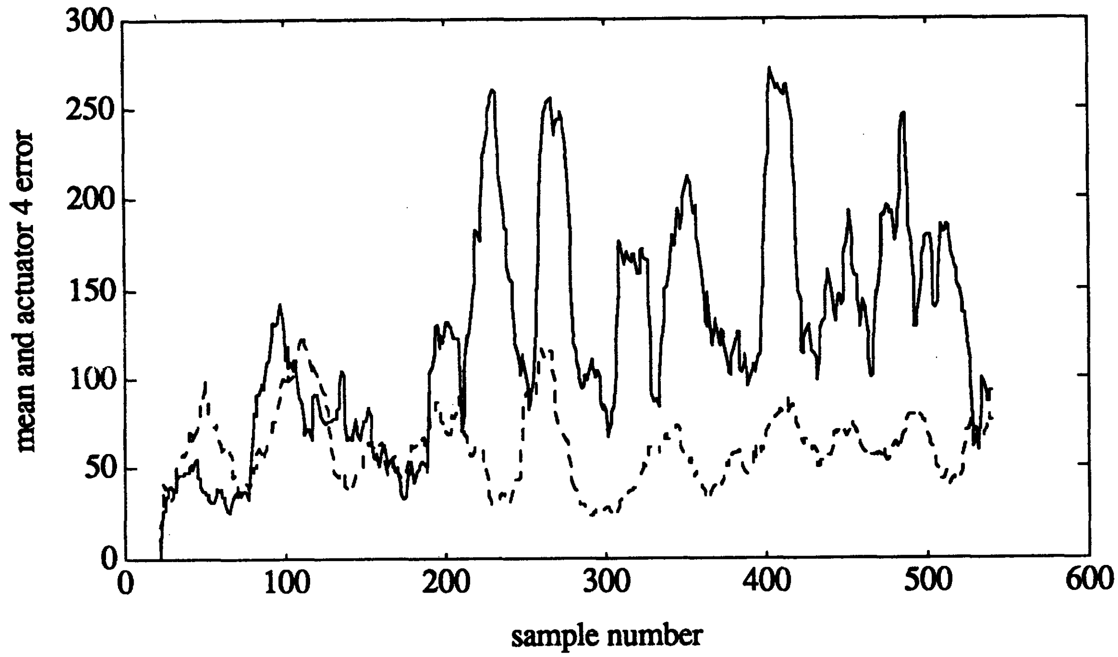


Figure 5-4 Soft Actuator Failure, Mean (solid line) and Actuator 4 Error (dashed line) Displayed

Figure 5-5 illustrates how relative-value comparison views the velocity errors, here contrasting error 4 to the next highest.

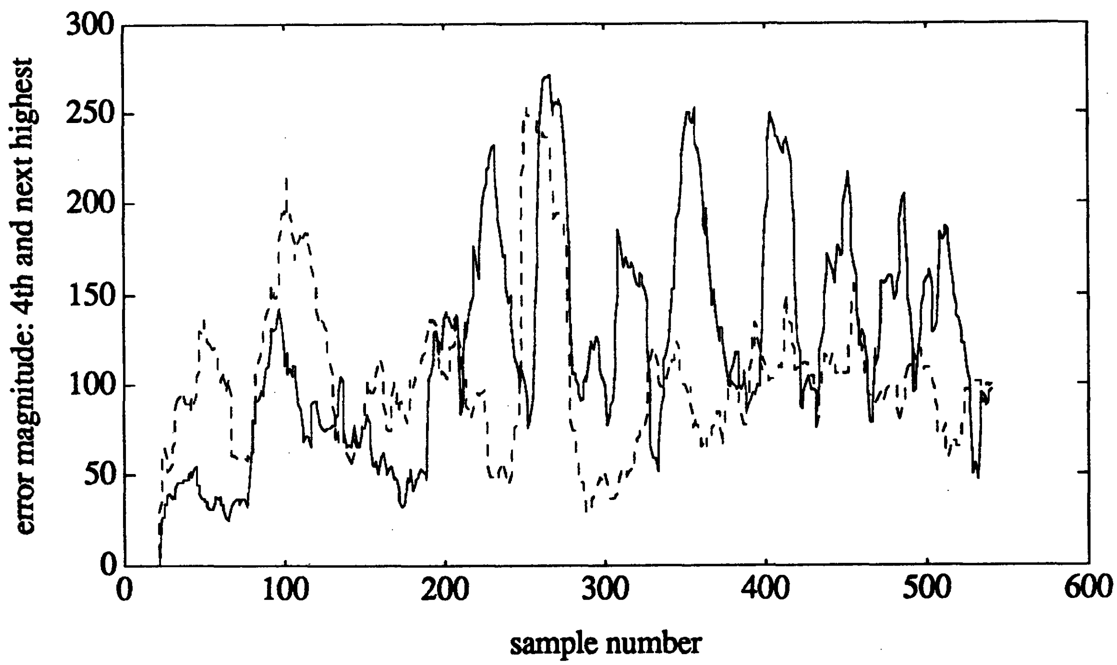


Figure 5-5 Error Values of Actuator 4 (solid line) and the Next Highest Value (dashed line)

A similar threshold can be identified, which should be less than the mean-value limit (since a higher ratio would not only be too stringent, but make the mean-value comparison unnecessary; basic algebra). A likely range in this experiment might be 1.5:1 to 2:1.

Absolute-value comparison is easily demonstrated without a graph; if the measurement error ventures above a minimum level, an alert is asserted (again, avoiding near-zero clutter which plagues ratios).

5.2.1 Single Failure Results

Threshold coefficients were selected by trial-and-error both to minimize false alarms and maximize the number of correct warnings. The mean-value comparison was set at 2 to 1; relative-value comparison at 1.5 to 1; and absolute value comparison at, say, 50 (though for this case the absolute-value condition may not be necessary since the errors rarely approach zero). If any of the error values ever exceeds all of these conditions, a trigger will be set to indicate which component is beyond the collective "warning" threshold. Here is the resulting warning function:

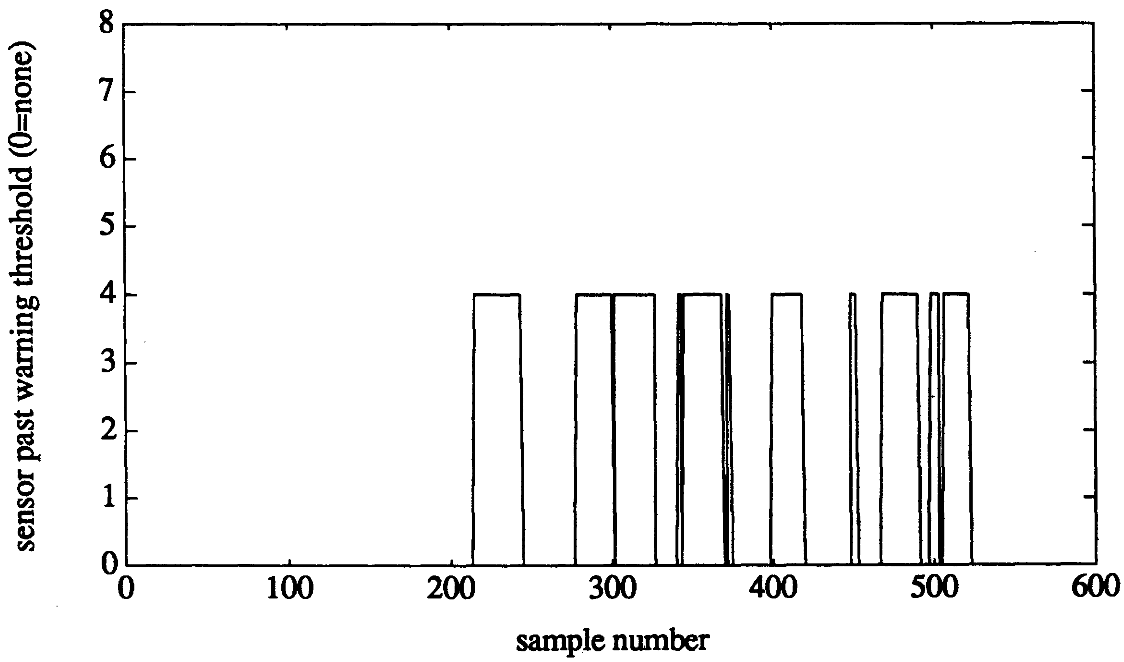


Figure 5-6 Warning Function Graph for an Actuator Failure

The magnitude of the function indicates which sensor the warning algorithm believes to be faulty, i.e. a magnitude of 4 indicates suspicion of actuator 4, etc. Remarkably, no false alarms were reported, despite the noise at 260 samples, though one can easily see a "no-alert" gap in that area. The warnings are relatively coherent starting just 12 samples (24 milliseconds) after the failure injection, and seem to be able to tolerate normal operation noise before 200 samples. With the minimum response time required (before 400 samples) in mind, and considering also the compressor's proven resistance to any actuator failures, this performance can easily be called 'promising.'

A "harder" failure test may prove useful, say, replacement of a signal with random noise at the same standard deviation of a typical actuator signal, again at 200 seconds with a 20-sample moving window. Here are the error traces and the "warning function" respec... with the same coefficients as before):

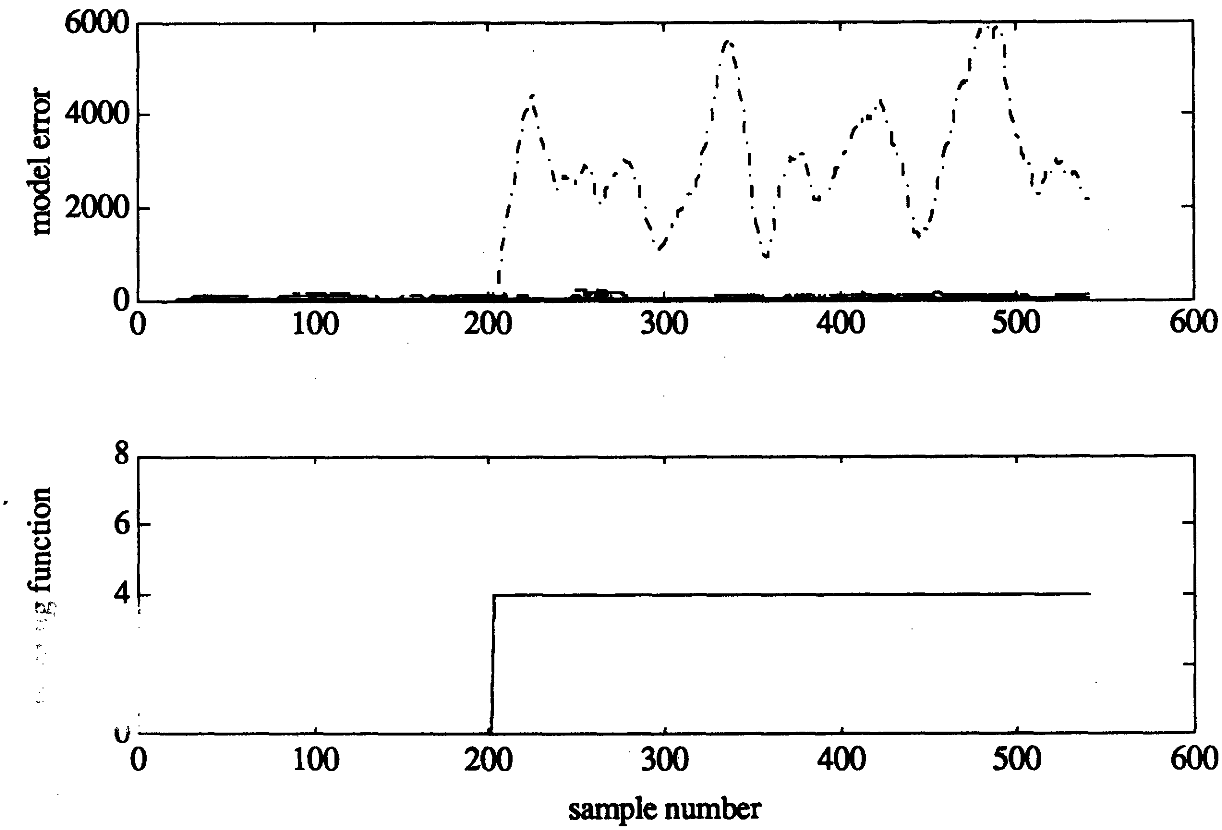


Figure 5-7 Pegged Failure Error (top) and Interpreted Warning Function (bottom)

It is brutally obvious that the failure is detected immediately, in fact, two samples (4 milliseconds) after it occurs. Even the disturbances at 260 seconds seem paltry in comparison to the error generated by random noise.

5.2.2 Multiple Simultaneous Failures

In the following experiment, where three actuators (numbers 4, 7 and 10 -- fully a quarter of the distributed system!) suffer simultaneous bias errors, a distributed three-warning-function as implemented in chapter 4 identifies all failures with ease (center), whereas the single warning function(bottom) manifests considerable confusion, unable to decide which of the three errors is worst from one sample to the next (see Figure 5-8).

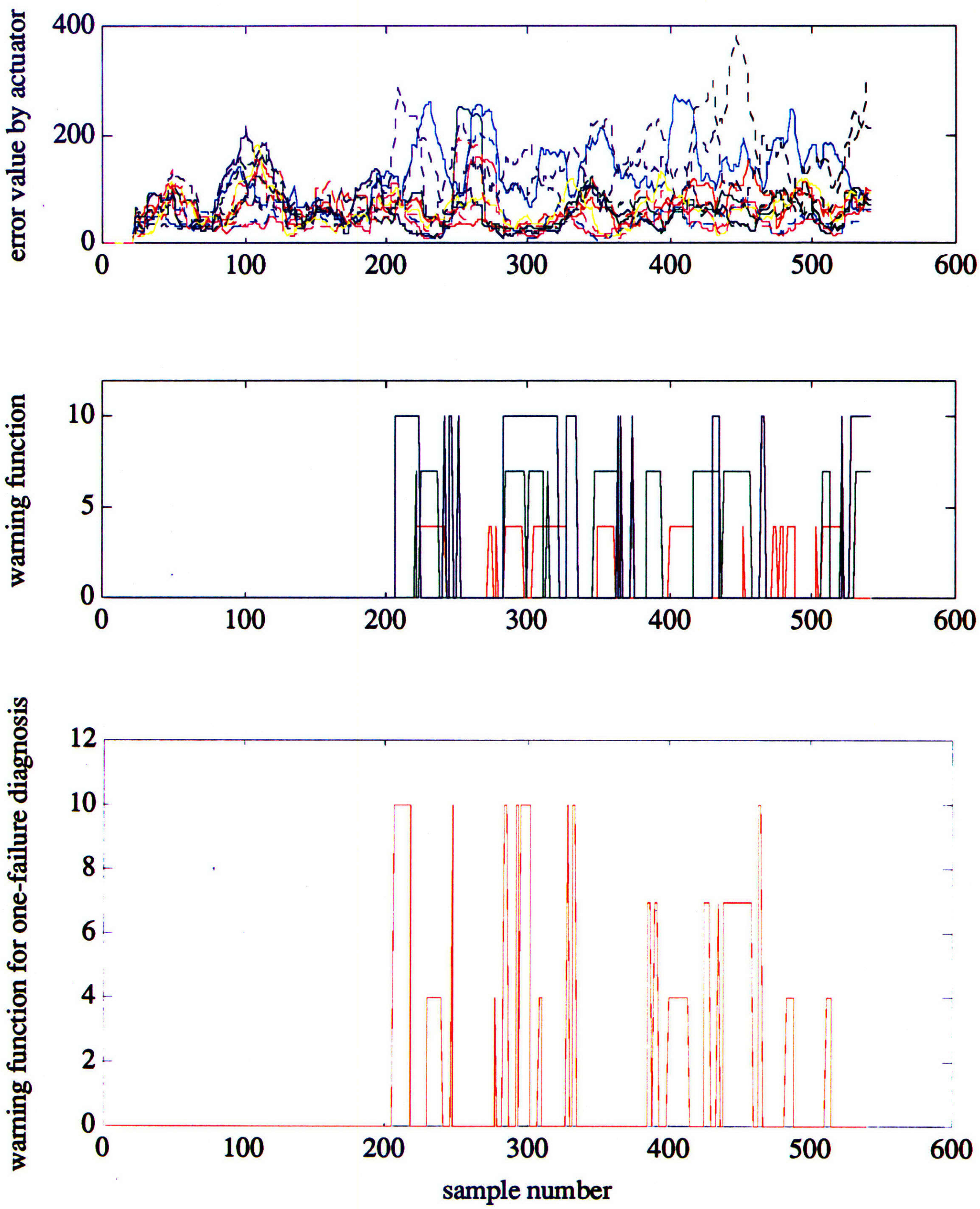


Figure 5-8 Three-Actuator Failures with Multiple- (middle) and Single-failure (bottom) Warnings

Each color of the distributed warning function represents a separate but concurrently computed element of the function as described in chapter 4; in the graph above, lines of the three colors can easily be discerned rising to the 4, 7 and 10 value magnitudes, again with no false alarms near 260 samples.

Once again, the performance is impressive, even for this soft failure which is all but invisible in the blade inputs and outputs. It may be that the simplex warning function could identify each failure sequentially (there are alerts in the last of the three graphs at each of the three actuator locations, 4 , 7 and 10) and remove them from the control loop; however, implementing a distributed warning function algorithm involves only slightly more processing at considerably higher reliability and speed (all three failures are identified by 218 samples, whereas nearly 200 samples pass before actuator 7 is triggered by the simplex algorithm).

Many other modes, both single and simultaneous were tested exhaustively with [illegible] results (they are not included here). At this [illegible] with ample confidence [illegible]des that actuator failure detection can be computed w[illegible] reliability approaching [illegible], given the availability of the actuator outputs.

5.3 Sensor Mechanism Results

The moving window size, or the mean-value, relative-value and absolute-value thresholds can be altered in successive tests to determine the effect each has on reliable detection, much like the process described for actuators above. Conducting many such tests at an experimental level provides clear indications of where best to place each variable.

It is impractical to include or discuss individually the hundreds of such tests that were conducted in the researching of this report. The following pages contain the results of a few of these tests representing a selective group of simulations which were chosen to i[illegible] the major issues at play in local, diffuse parity relation and power spectral [illegible] They represent the typical response of each mechanism to failures, not the "bes[illegible]" Once again, all tests used actual flow data, not "predicted" measurements, with simulated failures injected among the measurements. While acquiring the actuator inputs, outputs and, most importantly, flowfield estimates, the compressor was running with first harmonic mode control at a flow coefficient of Φ=.397, which is not only unstable in the open-loop system, but quite close to the stall point of the compressor with control (at Φ=0.387, thus the operating point is well within the typical 10-12%

stall margin). Also, the measurements are not steady; there are significant perturbations in the overall flow, though not enough to imply instability.

It is consequently believed that test results using this data should be "reasonably" close to worst-case first mode control conditions. Naturally, local and diffuse parity relations were tested at other flow coefficients as well, with similar or better results (due to greater stall margin and fewer perturbations). They are not included here. It is believed that any conclusions drawn from filtering this data set should be sufficient to demonstrate a robust "demonstration and validation" of the concepts defined herein which, after all, is the scope of this report.

Each of the graphs presented in this chapter contains two plots: in the upper plot, the moving window error for each sensor is displayed (the absolute magnitude is meaningless; only the relative size is important). The colored lines on each plot represent, for sensors 1 through 8 in order: red, green, dark blue, light blue, pink, yellow, dotted black and dotted red.[11] The lower plot is the "warning function" which is generated directly from the moving window error, based on the comparison ratios outlined in section 4.6. Again, the magnitude of the warning function represents the sensor it considers erroneous (0 indicating a vote of confidence). Unless otherwise specified for parity tests, the mean-value ratio is 2:1, the relative-value ratio 1.5:1, and the absolute-value constant .2 for local parity tests and .07 for the diffuse tests (these values were chosen after extensive iteration, much like the actuator thresholds were, and effect a minimum of false alarms while maximizing the probability of detection). For the PSD tests, both minimum and maximum thresholds are employed (again, defining a bidirectional comparison band rather than a unidirectional limit). The PSD mean-value threshold is 1:5, and conversely 5:1, the relative-value 1:4 and 4:1, and absolute-value threshold at 75. All single-point failures were injected at sensor 4 starting at 200 seconds; simultaneous failures were all injected at sensors 4 and 5 or 4 and 8 (i.e., adjacent and opposite positions). Once again, these are simulations that are maintained for all of the tests to standardize the relative outputs of the detection mechanism and best analyze their respective advantages. These results are not meant to be an exhaustive validation of byzantine fault coverage so much as a reasonably detailed look at capturing the flavor of each detection mechanism's response to different failures and slight changes in its algorithmic structure.

[11] Some of the simultaneous failure experiments used data from a 12-sensor refit of the compressor. The colors corresponding to these traces, after the eighth dotted red line, are dotted green, dotted dark blue, dotted light blue, and dotted pink respectively. Refer to the colormap key at the beginning of the report.

5.3.1 Experiments with Single Noisy Failures

In the following eight tests, random noise at the same standard deviation of the other signals was injected at sensor 4 at 200 seconds. Figure S-1 shows the response of a local parity test sampling three adjacent-to-site sensors with a moving window size of 20 samples.

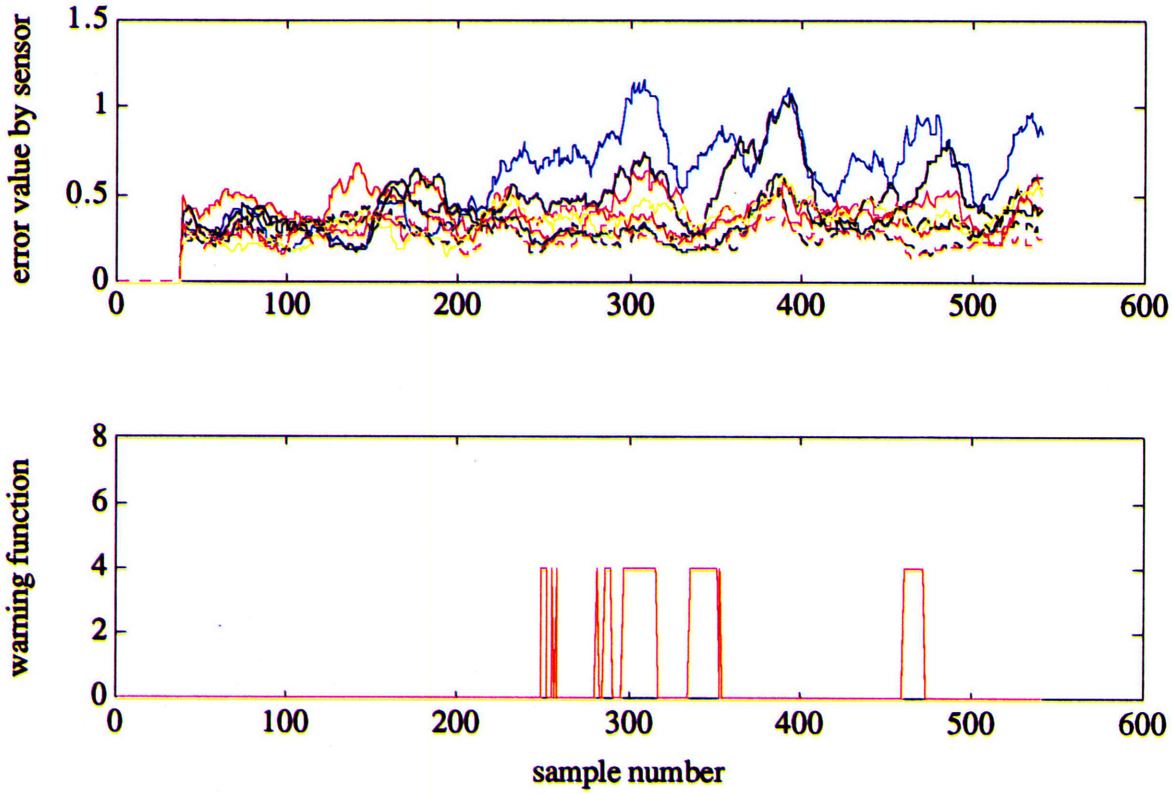


Figure S-1 Three-Poll 20-sample Local Parity Test with Noisy Failure

Shortly after 200 seconds, the light blue line (sensor error 4) noticeably creeps up beyond the background noise of the other signals and maintains its height, roughly, to the end of the test run. The warning function triggers first at sample 248, 96 milliseconds after the injection. Throughout the run there are instances where other signals interfere, generating a warning function that looks sporadic and glitchy. This is primarily due to error propagation into the adjacent comparisons; the dark blue, pink, and yellow lines representing nearby sensors 3, 5, and 6 are clearly prominent. This is a further indication that sensor 4 is the cause of the error. The warning function, however, is a simple mathematical model and is not intelligent enough to notice this. However, it is an important detail.

Figures S-2 and S-3 show identical tests with 4 and 5 adjacent sensors sampled respectively, flexibility that was built into the algorithm. The first graph does show improvement; a warning is triggered at sample 236, 12 samples earlier, and the response in the warning function is more noticeable; there are still glitches and periods of no-detection, but near inception, at least, the error is much clearer. The error is also higher, evidence that the predictions of section 4.3.1 were at least partially correct; the failure spike is more well-defined.

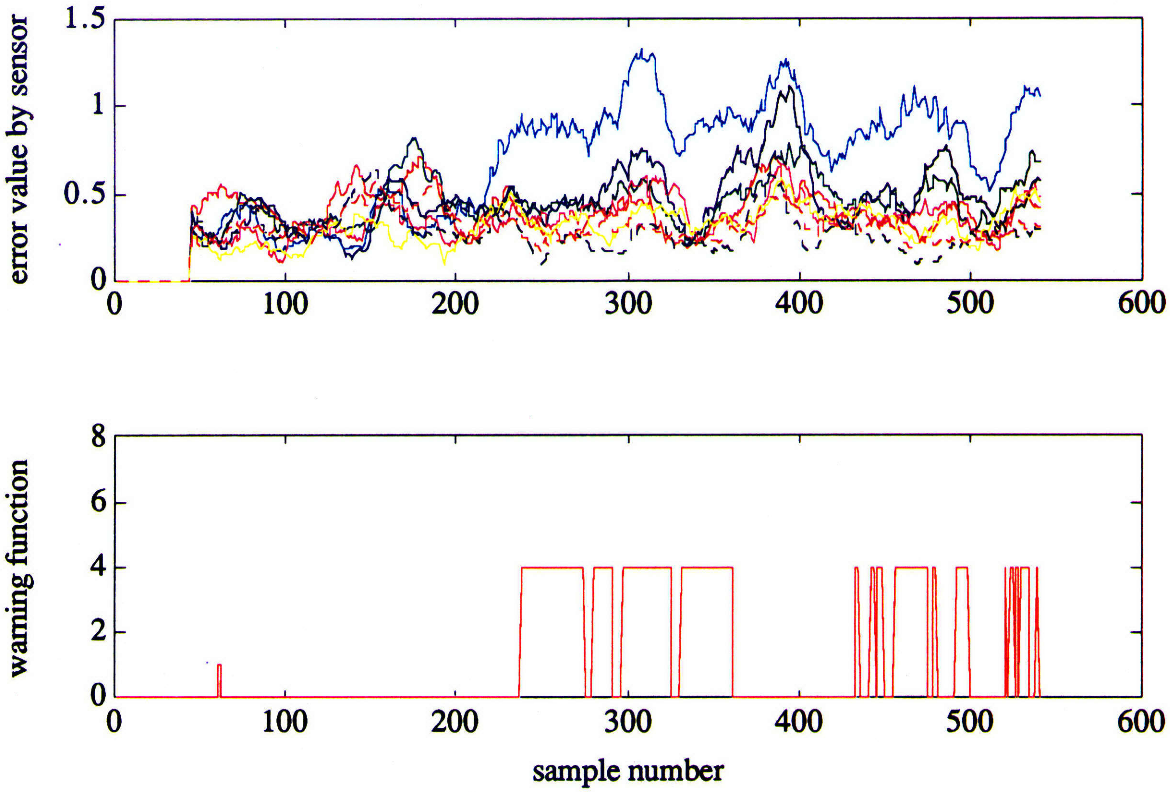


Figure S-2 Four-Poll 20-sample Local Parity Test with Noisy Failure

In figure S-3, however, there is a marked degradation of performance as one more component is included in the comparison. The difference in the error plot is slight, but sensor 4's error can be discerned propagating into five of the other seven comparison tests, and one sees many of those errors rising to obscure error 4, giving a warning function that is much less reliable. The errors do not trigger until sample 263, but there is no continuous warning; the most obvious indication of failure does not occur until sample 338. Just as predicted, sampling too many sensors can be dangerous, and here dominates the response characteristic. One may argue, however, that this may be due to the nature of the comparison envelope. Altering the comparison ratios may provide a superior response. However, this is

not the case. Lowering any of the ratio values does produce a slightly better warning function, but at the cost of many false alarms. A four-sensor polling algorithm seems to be the best choice.

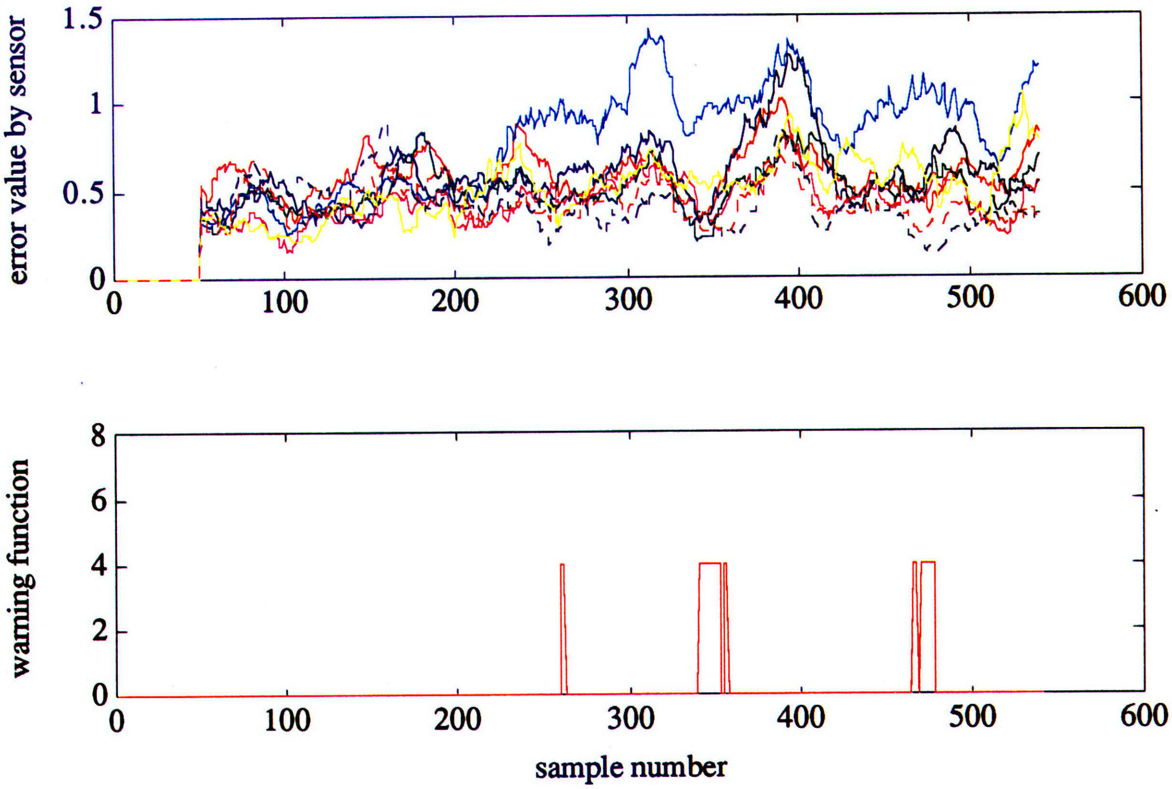


Figure S-3 Five-Poll 20-sample Local Parity Test with Noisy Failure

Other compressors or distributed system architectures may exhibit different performance when altering the number of polled sensors; the driving requirements for this choice are primarily the amount of background compressor noise and the size of the distributed system.

Perhaps shrinking the window size may give a shorter time response; figure S-4 illustrates the same test as S-1 performed with a moving window of 10 samples instead of 20:Indeed, a warning is indicated early, at sample 235, but here as well there are false warnings, though none as "wide" in time as that at samples 235-250. Perhaps an intelligent system could mask such false alarms by examining the width of the warning pulse to eliminate glitches, but test S-2 triggers just as early with **no** false alerts and a less glitchy output.

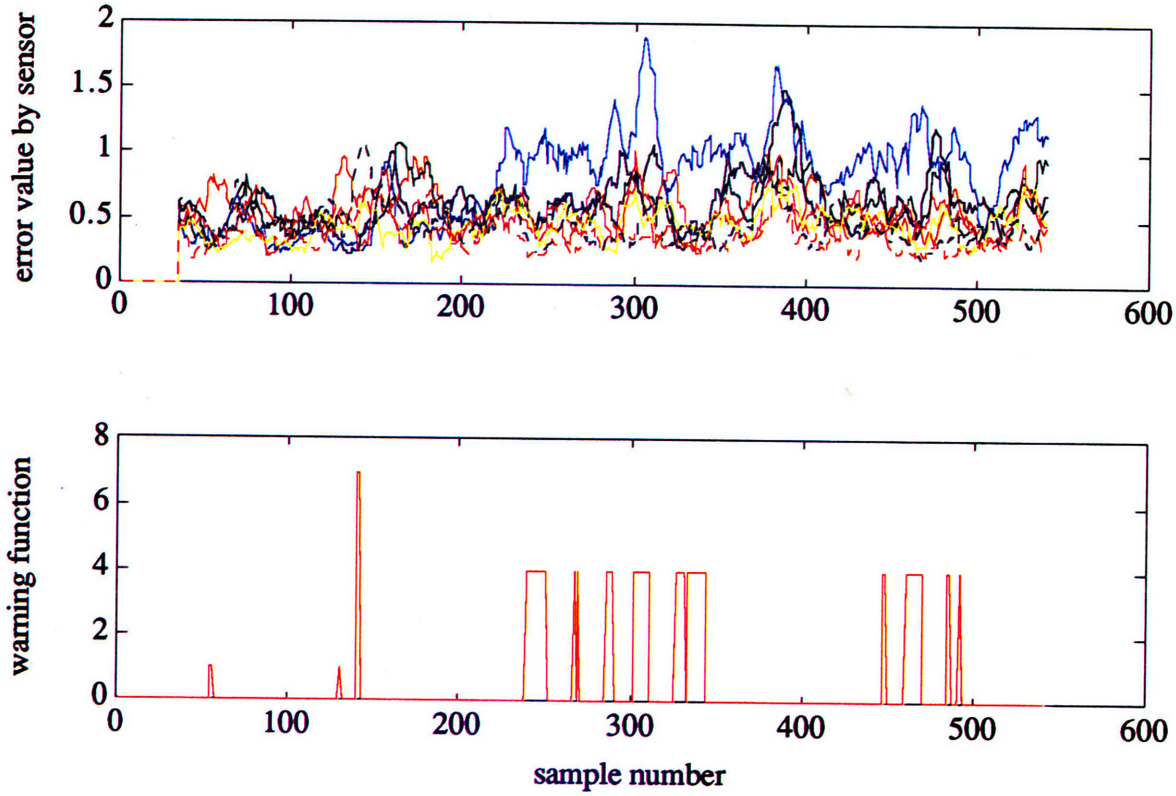
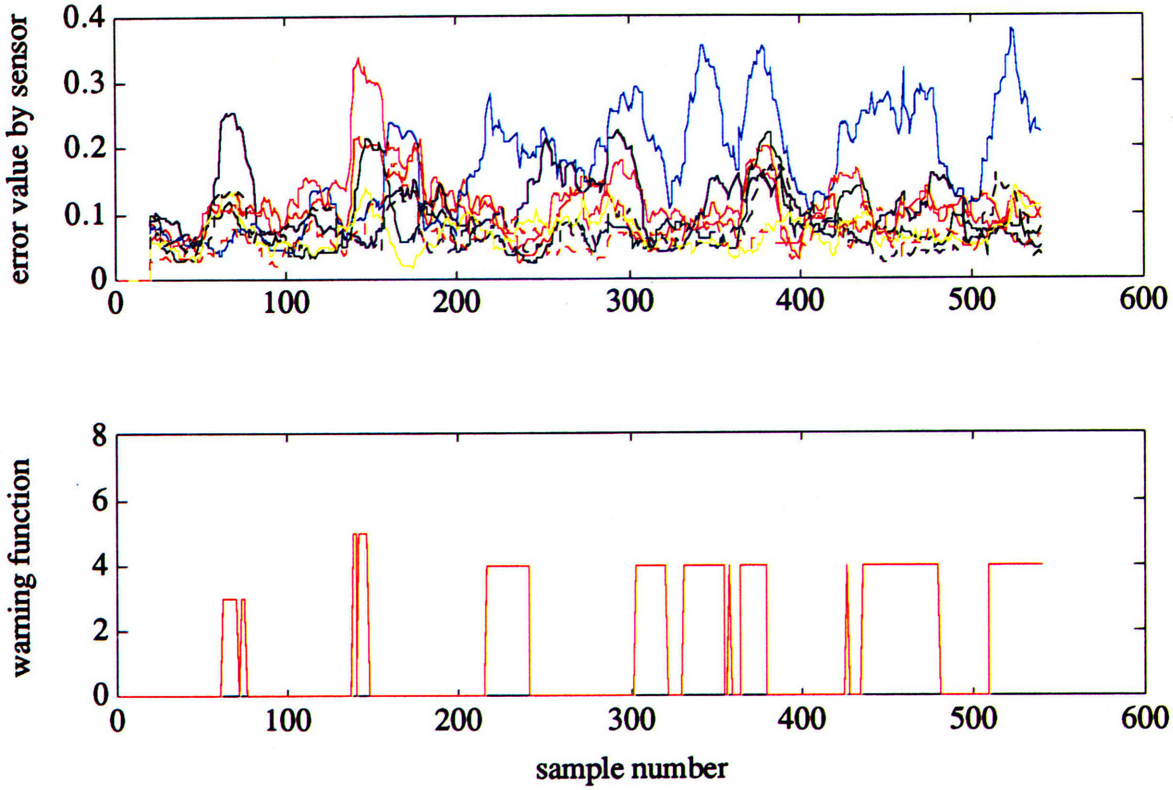
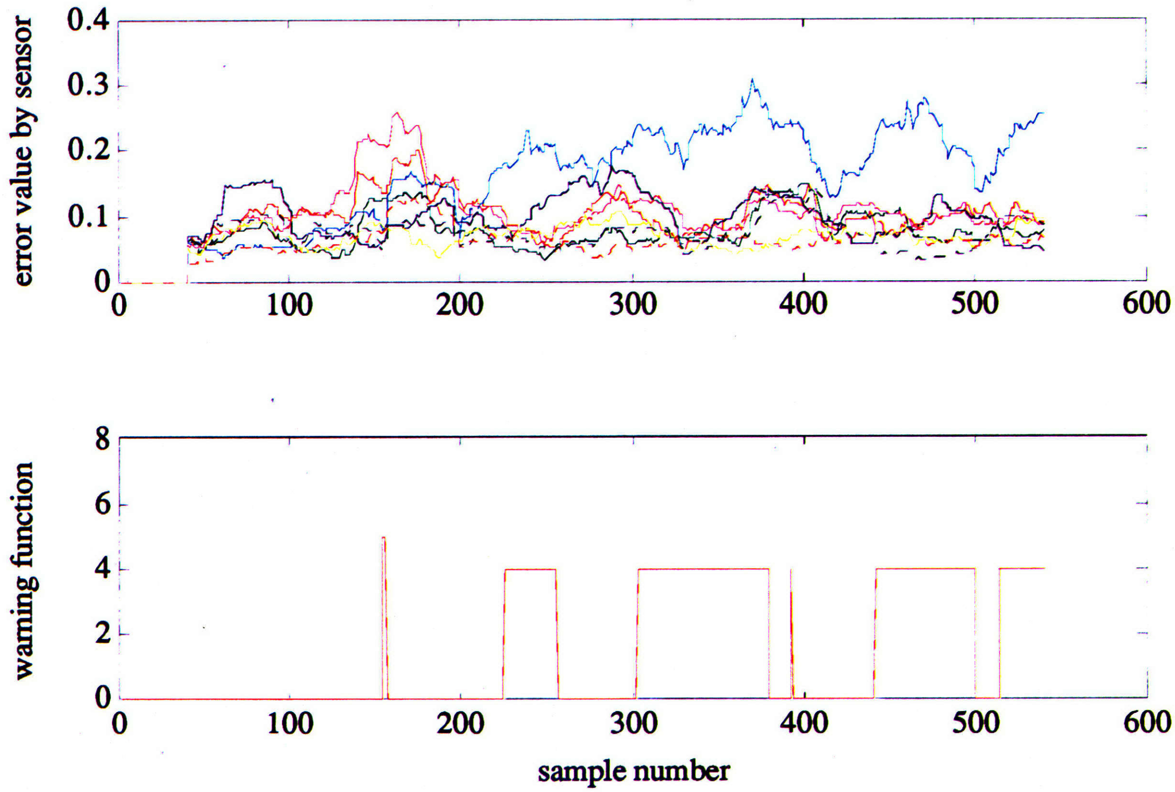


Figure S-4 Four-Poll 10-sample Local Parity Test with Noisy Failure

Next the diffuse parity mechanism makes an attempt with the identical measurements. Figures S-5 and S-6 were produced from the diffuse parity test with identical inputs and comparison ratios (as outlined above). In Figure S-5 there are several false alerts triggered prior-to-failure, implying a higher sensitivity to fluid disturbances. Indeed, at the same locations on the upper plot of S-5 there are two noticeable jumps in error magnitude at 75 and 150 seconds, whereas there are no such deviations in the local parity tests. However, S-5 also shows a correct and smooth fault alert from 214 to 242 samples, a response time which excels the best of the local detections by 21 samples! The coherency of the subsequent warnings to the end of the run is also comparable to that of S-2. In S-6 the false alerts are eliminated or reduced to single-sample spikes with a 40-sample moving window; the failure is first detected at sample 224, again beating the response time of S-2.





Figures S-5 and S-6 Sensor Diffuse Parity Test, 20 and 40 sample moving windows respectively, noisy failure

The failure detection filters employ several error signals at once; Figures S-7A and B illustrate the residuals generated by each filter set. Hopefully, residuals EA1 in the 1to4/5to8 set and EB2 in the odd/even set should rise after 200 samples to imply errors in sensors 2 or 4. Even a cursory glance, however, shows that no such indications are present. Any errors are indistinguishable from normal operation and provide no useful failure information.

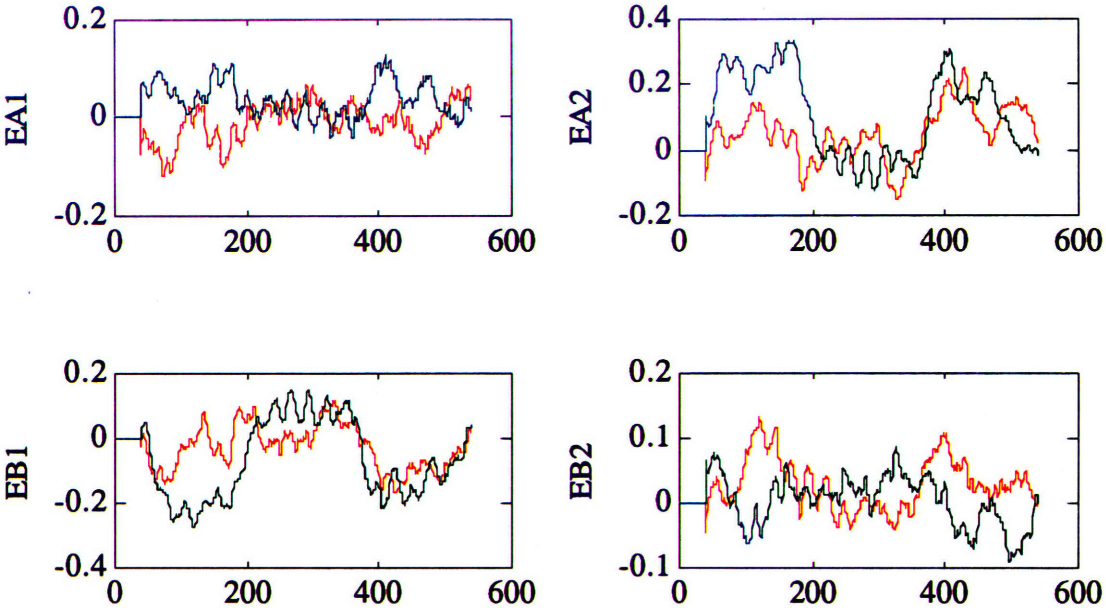


Figure S-7A Residuals for 1to4/5to8 Filter Set, Noisy Failure

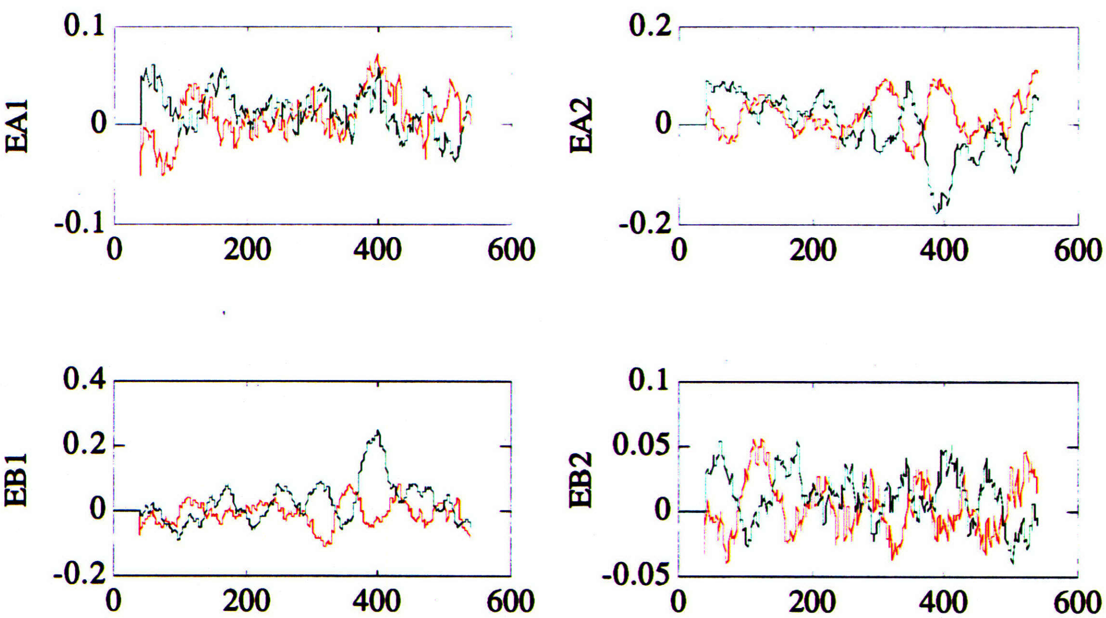


Figure S-7B Residuals for Odd/Even Filter Set, Noisy Failure

This is not really surprising; failures such as random ones with high-frequency components tend to be the most difficult assessments for detection filters. Faults of this type are very similar to the random noise from modelling error, and stay within the background error envelope; furthermore, any directional information is difficult to attain as a rapidly changing residual might seem to skip around the unit circle of event vectors, instead of maintaining a linear direction.

S-8 depicts the results from the 128-point PSD test which seems to have trouble identifying the error quickly, primarily because the PSD of the flow perturbations wander low enough to approach that of the noise. When the perturbations start to appear, however, the error rises immediately and the warning function gives a smooth alert from 410 to 495 samples.

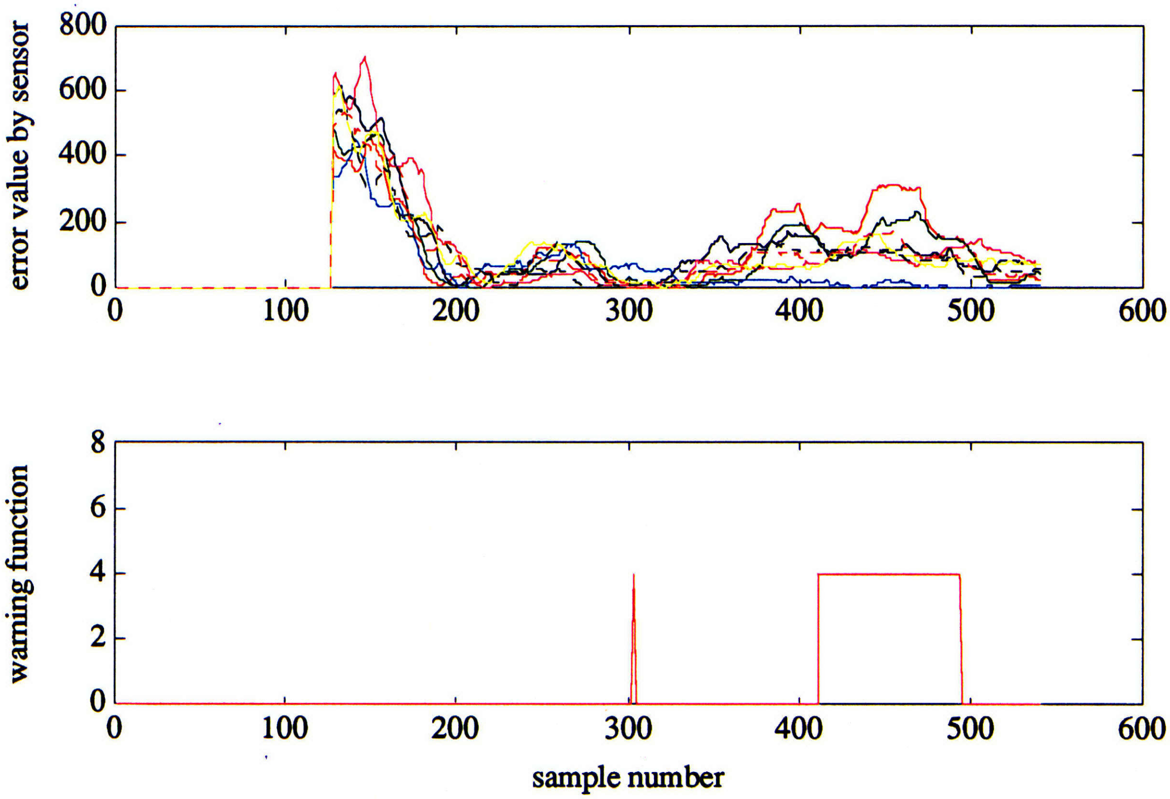


Figure S-8 Power Spectral Density Test, 128-point FFT, noisy failure

This test shows, as predicted, how susceptible PSD's can be to incomplete and time-varying inputs. After 200 samples, the PSD algorithm is calculating with two very distinct and concatenated velocity shapes -- one random, and one corresponding to typical compressor operation -- and is clearly confused. When the noise completely dominates the PSD moving window, however (after 200+128 = 328 samples), the ten-Hertz component stays

very low, as it should for random noise. This further illustrates the danger of large-point PSD's, as they can become confused with time-varying signals (which may or may not contain failures). This 128-point PSD performs relatively well here.

5.3.2 Observations on Mechanism Spheres of Influence

Perhaps the most important illustration of performance so far lies not in **how well** each one performs, but **where** each one performs well. From even a cursory glance it is clear ... method functions fundamentally differently, having its own particular regions of ...rmance. This implies that each also has the capacity to detect failures in ...lind spot." For example, near sample 380, all of the local tests are starting to ...nifest significant error for several components, and the warning function sets itself to zero, unable to decide which sensors, if any, are faulty. However, the diffuse parity test shows an error in sensor 4 which is easily identified over the other signals, and produces a strong warning trigger in samples 365-378, where no such alerts exist in runs S-1 through S-4. Conversely, local test S-2 manifests an alert block at 500 samples, where little or no activity occurs in the diffuse tests. The PSD experiment, empty of alerts before 400 samples, picks up from 400 to 500 where the parity warnings are sporadic, and overlaps some significant gaps in the other detection algorithms.

Though at this stage, all three methods are starting to show some promise in reliably detecting failures, they are also establishing that the way they each respond to errors is fundamentally different. One may be confused, another certain; the fact is they have the capacity -- if combined intelligently -- to mask each other's "blind spots" so their disadvantages can be minimized.

5.3.3 Experiments with Simultaneous Noisy Failures

As in actuator failures, multiple simultaneous failures must be addressed as well. For a local parity test the worst conceivable failure mode, mathematically speaking, is two or more adjacent failures, which contribute equally to several sensor polls. Referring back to Figure 4-4, this would cause a band-shaped error, as opposed to two clear spikes, to rise up in local parity error signals. With the addition of noise in a mesh-type graph of Figure 4-4, this could be nearly impossible to classify in a warning function.

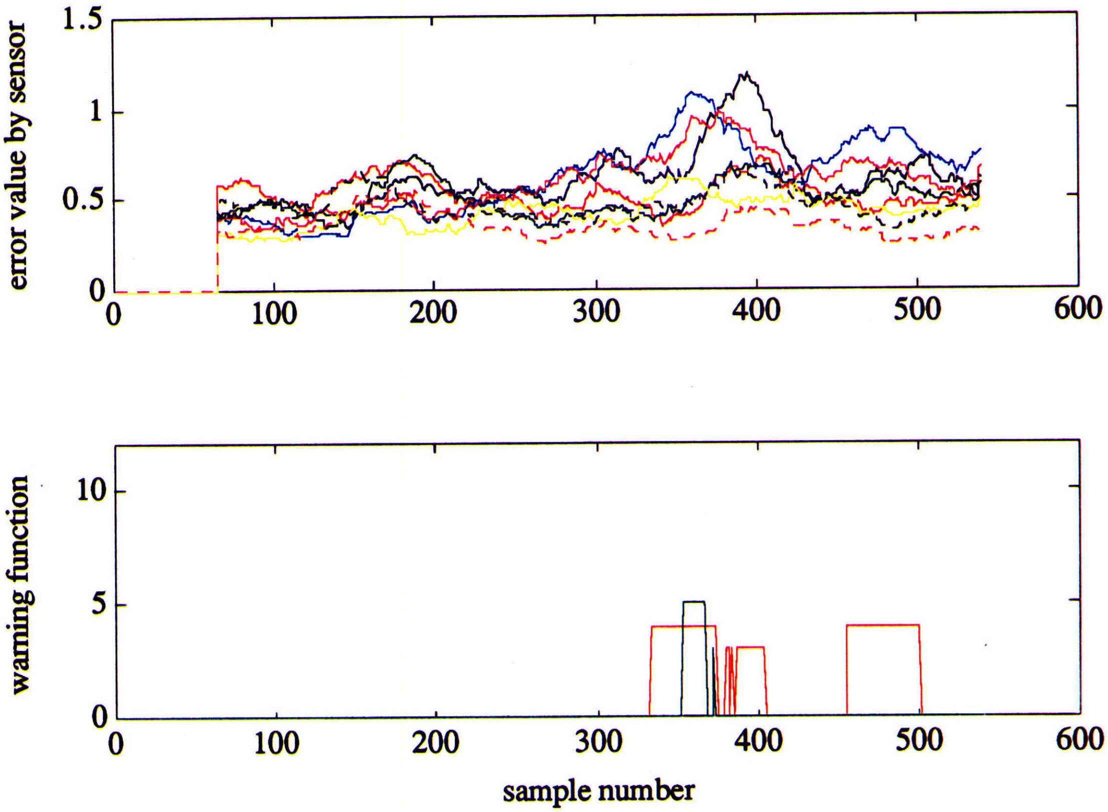


Figure S-9 Four-Poll Local Parity Test, Two Adjacent Simultaneous Noisy Failures

Indeed, in Figure S-9, which represents one of the cleanest iterations of the warning function, the local parity test has substantial trouble detecting failures of sensors 4 and 5 correctly. One false alert is prominent at 400 samples, where compressor noise and the two adjacent failures combine to lift sensor 3's error estimate (which includes sensors 4 and 5 in its polling) above all others. Diffuse parity tests suffer from the same problem, namely, multiple errors corrupting too many of the parity relations for the warning function to discern the state of the distributed system.

Multiple actuator failures were easy to detect since the model error mechanism performs no polling; failures in one component cannot corrupt another's error estimate. The most obvious solution to the multiple failure problem for sensors is a larger distributed system. The larger the measurement pool, the less susceptible these mechanisms should be to compressor noise and cross-correlating fault infection. This improvement is balanced by an increase in complexity and the mean time between failures: the more sensors, the higher the probability of experiencing failures. Again, another tradeoff has been illustrated.

In Figure S-10 (and all following simultaneous failure graphs) the compressor has been refitted with twelve sensors instead of eight. These twelve-sensor plots demonstrate results from a different set of measurements than those previously provided, and can be correlated to the eight-sensor data only slightly. However, the data contained in S-10 and others was taken immediately before a full compressor stall under first-mode control, and like Figure S-9, contains substantial disturbances in the compressor flow which prove demanding to the detection mechanisms.

Figure S-10 shows two noisy failures of sensors 4 and 5 occurring at sample 100.

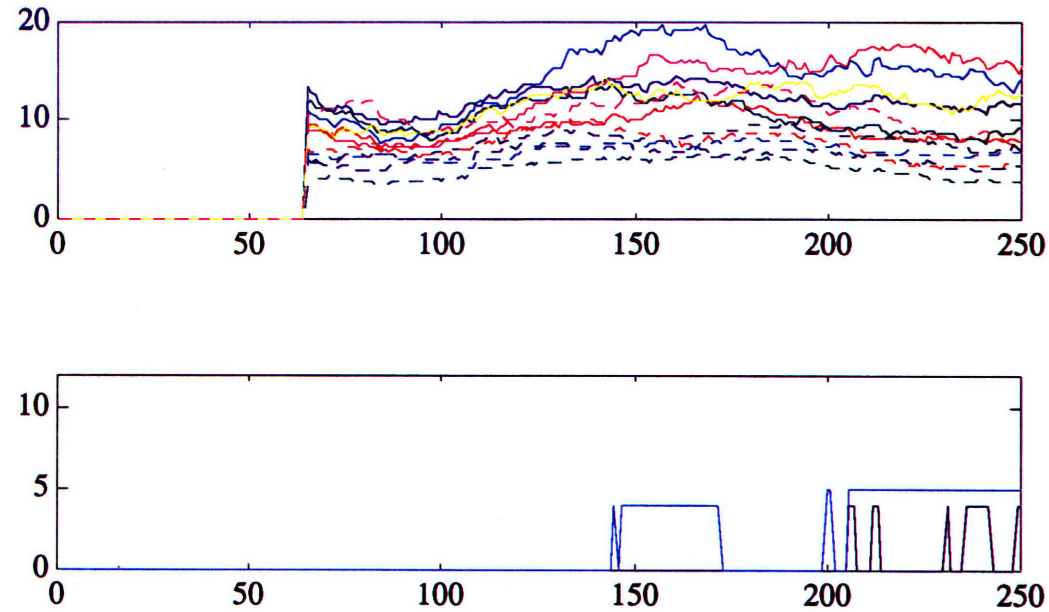


Figure S-10 Four-Poll Local Parity Test with Simultaneous Adjacent Noisy Failures

The error traces do prove confusing to the eye, but the multiple failure warning function retains the capacity to detect the failures by examining each error <u>and</u> the others affected by its polling region.

The diffuse parity test seems to function better as well, though it takes longer to reach a consensus than the local parity test. Figure S-11 shows the results from the same measurement set.

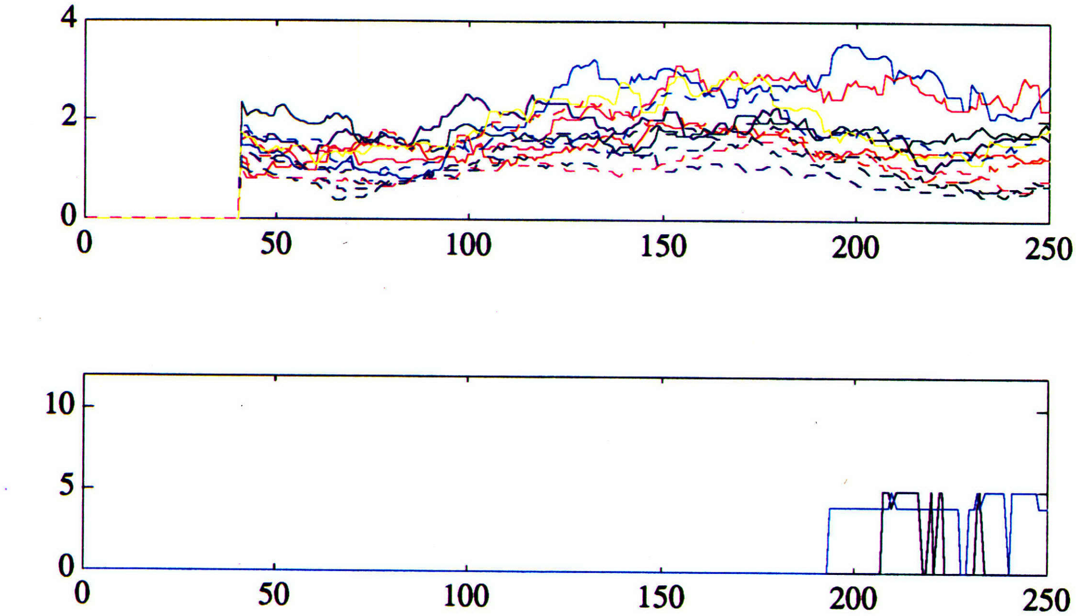


Figure S-11 Diffuse Parity Test with Simultaneous Adjacent Noisy Failures

It is the PSD mechanism that proves to be the most robust in this example, identifying both failures within 68 samples. Its performance is much more consistent than either parity test.

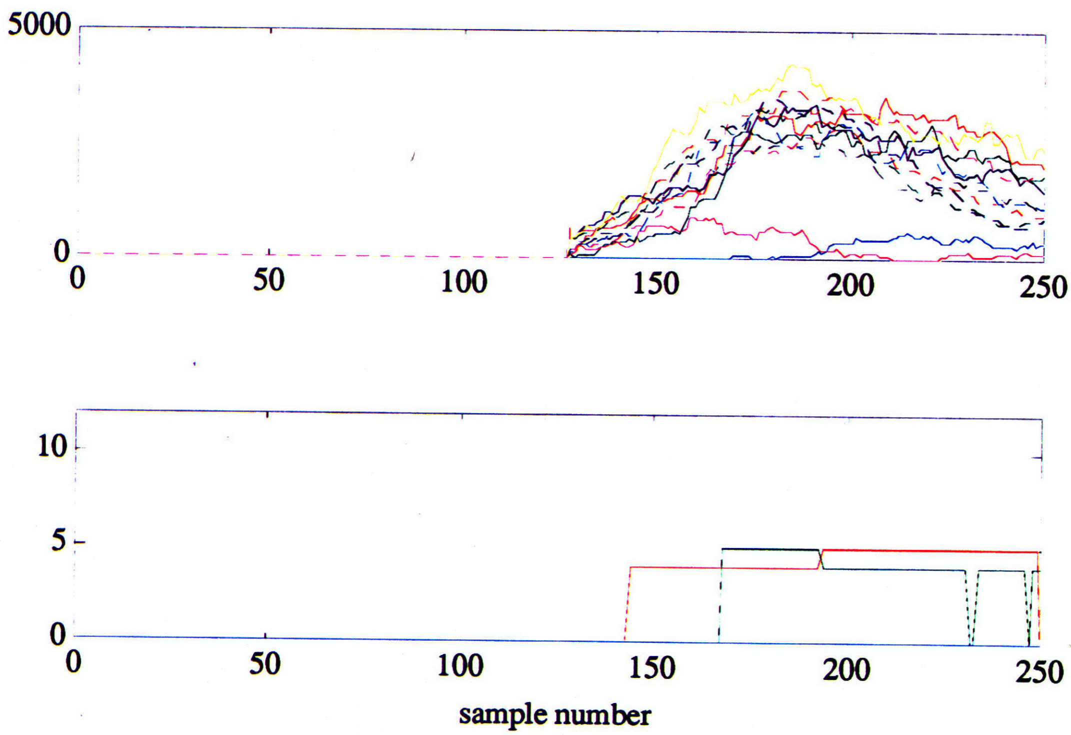


Figure S-12 PSD Test for Simultaneous Adjacent Noisy Failures

Changing the location of the failures produces similar results; a distributed warning function must still cope with several error signals which violate one or more of the comparison tests. In experiments with two failures opposite each other in position, local parity tests perform slightly better (manifesting two relatively distinct "error spikes" as in Figure 4-4); diffuse tests perform nearly the same; while PSD analysis again exceeds both in response time and coherency.

5.3.4 Experiments with Single Pegged Failures

No conclusions are easily validated from a single failure mode; there are infinitely more faults to be explored. Tests S-13 through S-18 were conducted while injecting a **pegged failure** at a value of .7 meters per second, well below the maximum ≈ 1.1 meters per second velocity perturbation observed in the measurements. S-12 depicts the same test as S-1: three adjacent sensors sampled at a 20-sample moving window size.

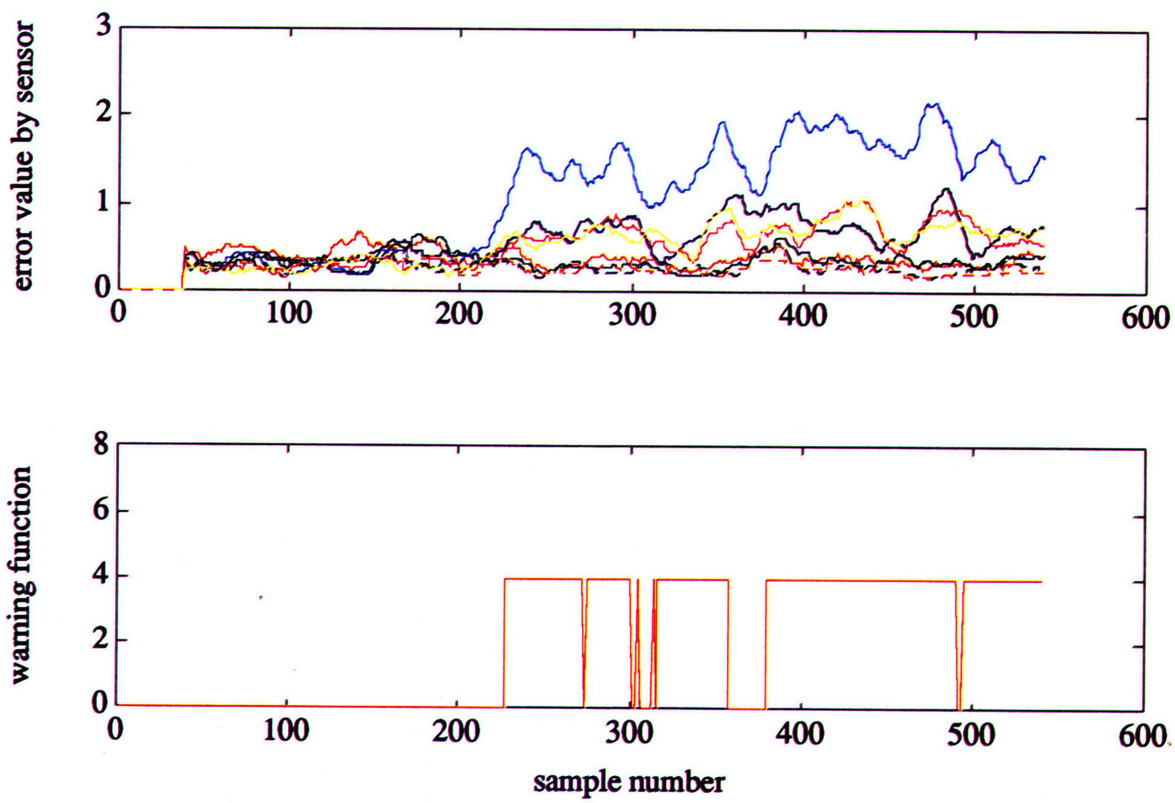


Figure S-13 Three-Poll 20-Sample Local Expectancy Parity Test for Pegged Failure

Obviously this failure mode is far easier to detect than a random noise failure; the errors are considerably higher and the warning function smoother. Again, the familiar
dark blue, pink and yellow lines from the three adjacent sensors rise as well, further evidence incriminating sensor 4 as its errors corrupt the surrounding polls. These adjacent errors, on the whole, seem about a third the size of sensor 4's, as they should. Further changes in the warning function algorithm to detect these coupled signals would no doubt improve the alerts at the bottom plot to almost a straight line (this is quite easily concluded from examining the error signals themselves). S-14 shows a reduced 10-sample window for the same test, at reduced reliability (with a false alert) and no improvement in response time.

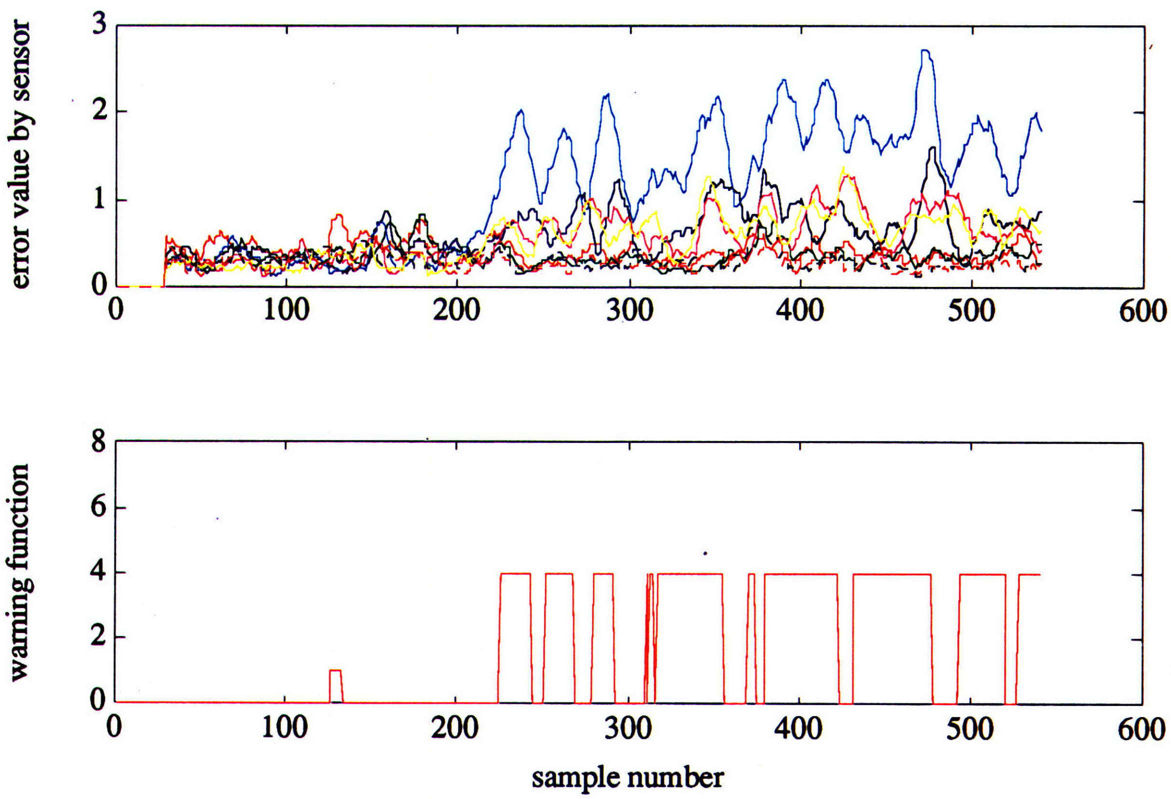
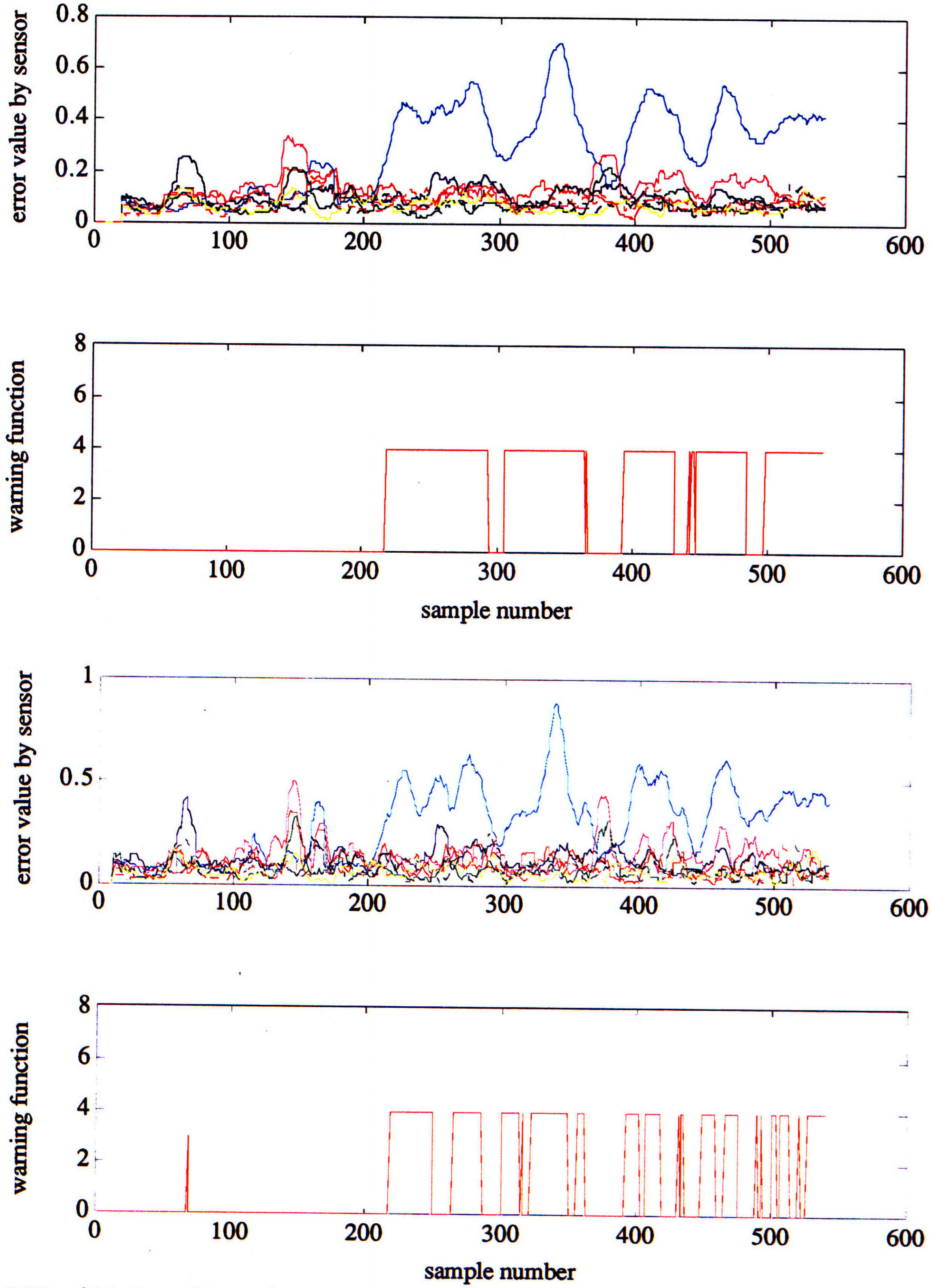


Figure S-14 Three-Poll 20-Sample Local Parity Test with Pegged Failure

S-15 and 16 are the diffuse tests for the pegged error at 20 and 10 window widths respectively. The relative- and mean-value tolerances have been adjusted to mask false alarms near 75 samples (false warnings are not being ignored, though, and were shown in previous graphs).



Figures S-15 and 16 Pegged 20- and 10-Sample Diffuse Parity Tests; mean-value ratio = 2.3; relative-value ratio = 2.3

These plots show similar performance with better response time, much like the noisy failure results; once again, there is evidence of blind spots in one method that are covered by another. These are most noticeable near samples 390 and 440 of the 20-window experiments.

Next, the **failure detection filter** makes an attempt. Figures S-17A and B show the four output residuals from each of the detection filters (refer to sections 4.4.2-4.4.3):

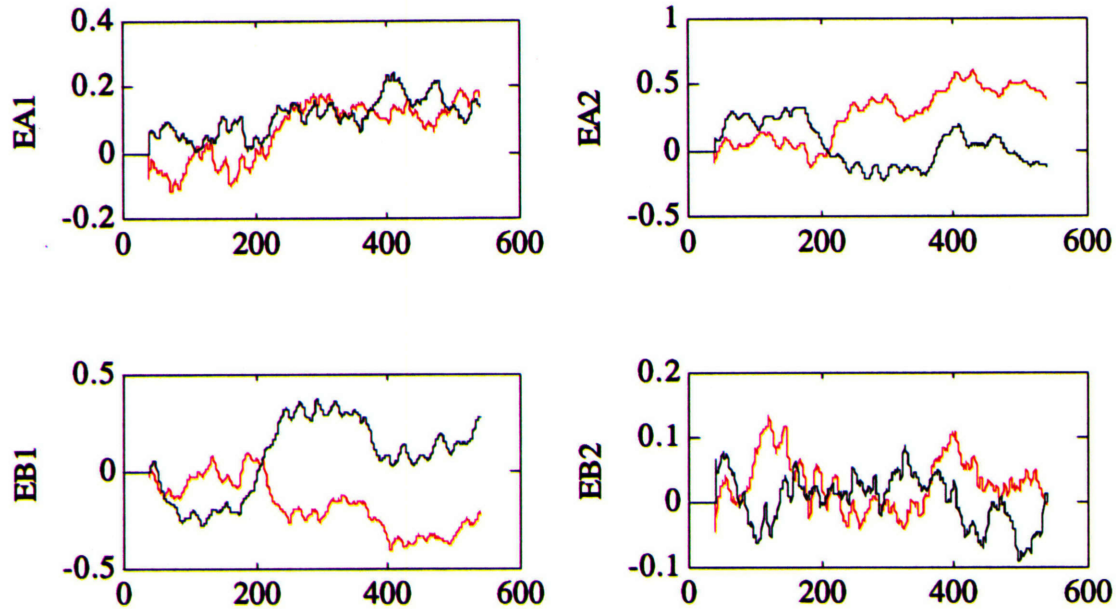


Figure S-17A Residuals for 1to4/5to8 Filter Set, Pegged Failure

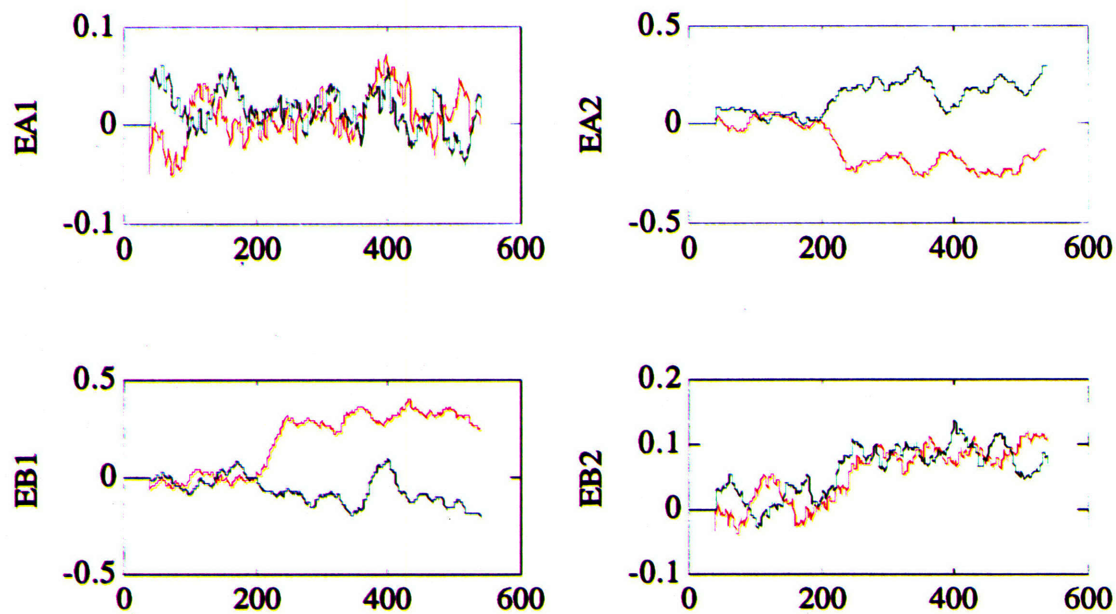


Figure S-17B Residuals for Odd/Even Filter Set, Pegged Failure

Fortunately, the residuals EA1 of S-17A and EB2 of S-17B rise as hoped, identifying that sensors 2 and 4 are indeed under suspicion. Examining these residuals, and the corresponding values which imply failure directions for each set produces the following failure signals (each of which is, again, a restatement of residuals EA1 and EB2) and proximity values:

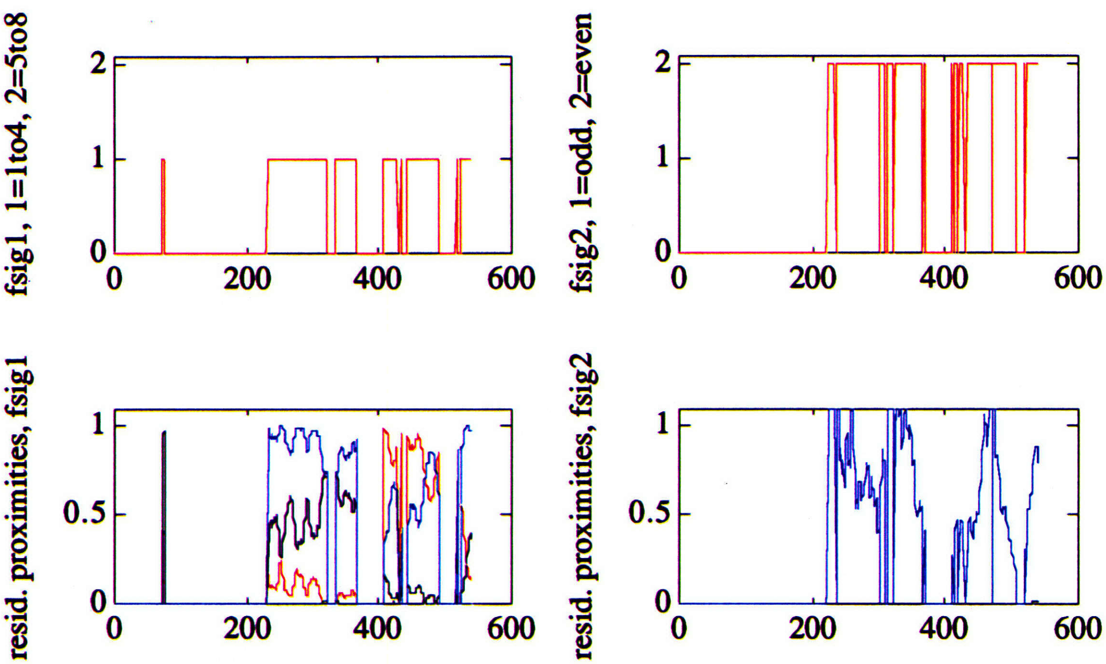


Figure S-17C Failure Signals and Proximity Functions, Pegged Failure

Both failure signals seem to agree that something is wrong shortly after 200 samples. The odd/even set (right column) reacts faster and is quite sure the failure lies along sensor 4 or 8's direction. This latter conclusion is clear from the light blue line at bottom right which dominates the graph; the dark blue (corresponding to sensors 2 or 6) is hardly visible from the cosine filtering.

The 1to4/5to8 filters also reacts quickly, but cannot make up its mind throughout the test whether the failure is in sensor 4 (light blue line at bottom left) or sensor 1 (red line). This degraded performance is perfectly logical, since the odd/even plant models approximate the fluid dynamics with superior accuracy. Clearly, however, the light blue line at bottom left remains as one of the top two proximities, and this characteristic leads to a well-behaved warning function when combined with the second filter set. The false alarms present in the right-hand column of S-17B are masked easily, and the function shape looks very promising for failure detection filters, at least for pegged failure identification (see Figure S-17D).

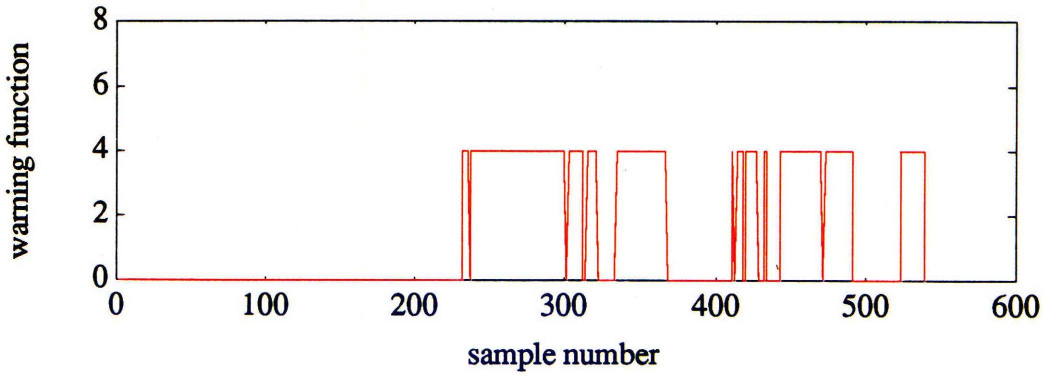


Figure S-17D Detection Filter Warning Function, Pegged Failure

The PSD experiment in S-18 reacts quickly as well, first manifesting warnings at 210 samples, well earlier than the first noisy-failure warning in S-7.

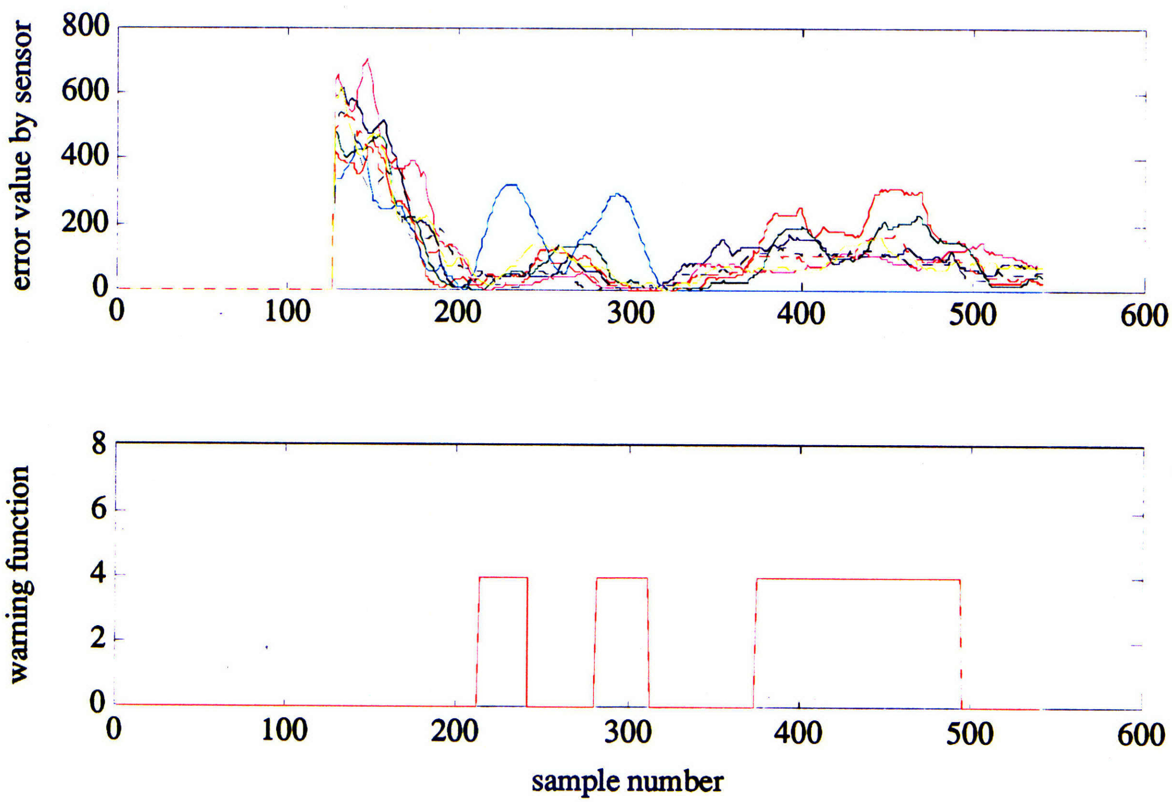


Figure S-18 Power Spectral Density Test, 128-Point FFT,Pegged Failure

5.3.5 Experiments with Simultaneous Pegged failures

Experiments with simultaneous failures are similarly related to single faults as in the noisy failure tests. In Figure S-19 (failures, again, of two components, here sensors 4 and 10 -- light blue and dotted dark blue) the local parity test has some trouble with false

alarms, manifesting two at 45 and 210 samples; whereas the diffuse test results contain a continuous false alert from 218 to 234 samples in Figure S-20.

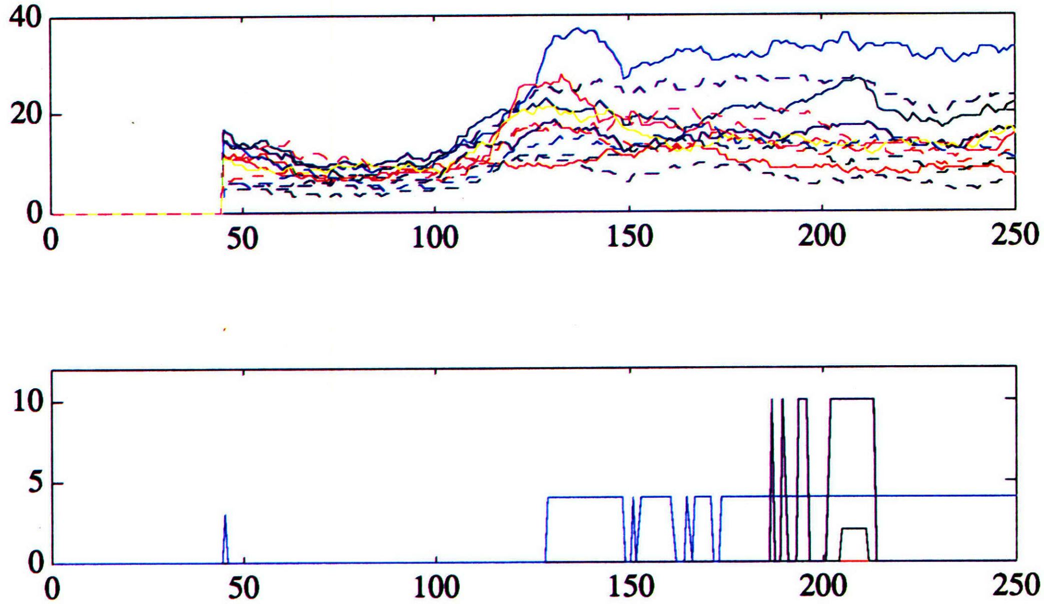


Figure S-19 Four-Poll 40-Sample Local Parity Test with Simultaneous Opposite Pegged Failures

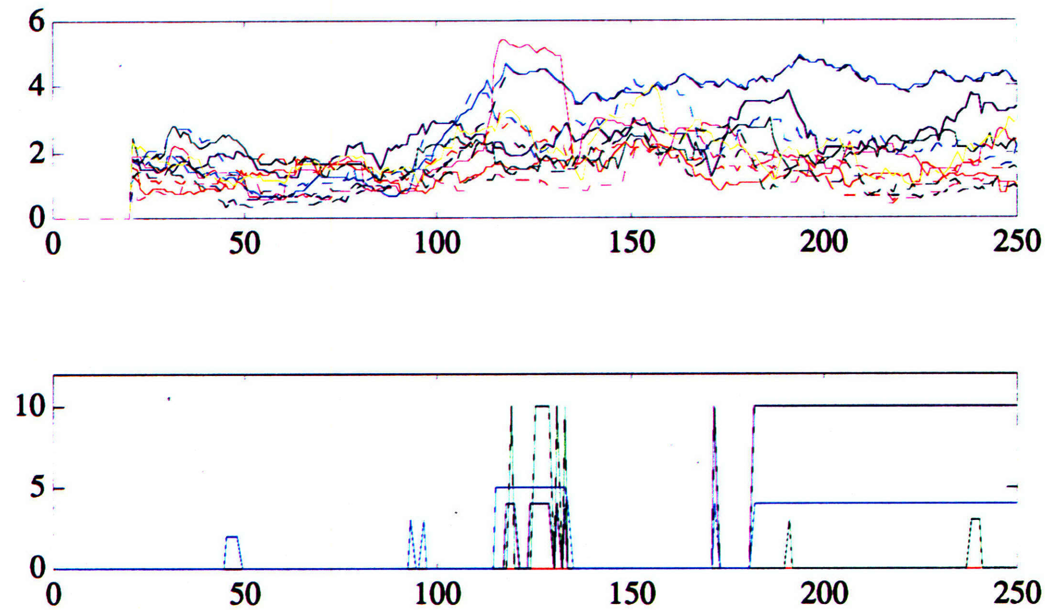


Figure S-20 20-Sample Diffuse Parity Test for Simultaneous Pegged Failures

And once again, the power spectral density test excels both parity mechanisms, manifesting consistent warnings with no false alarms (see Figure S-21).

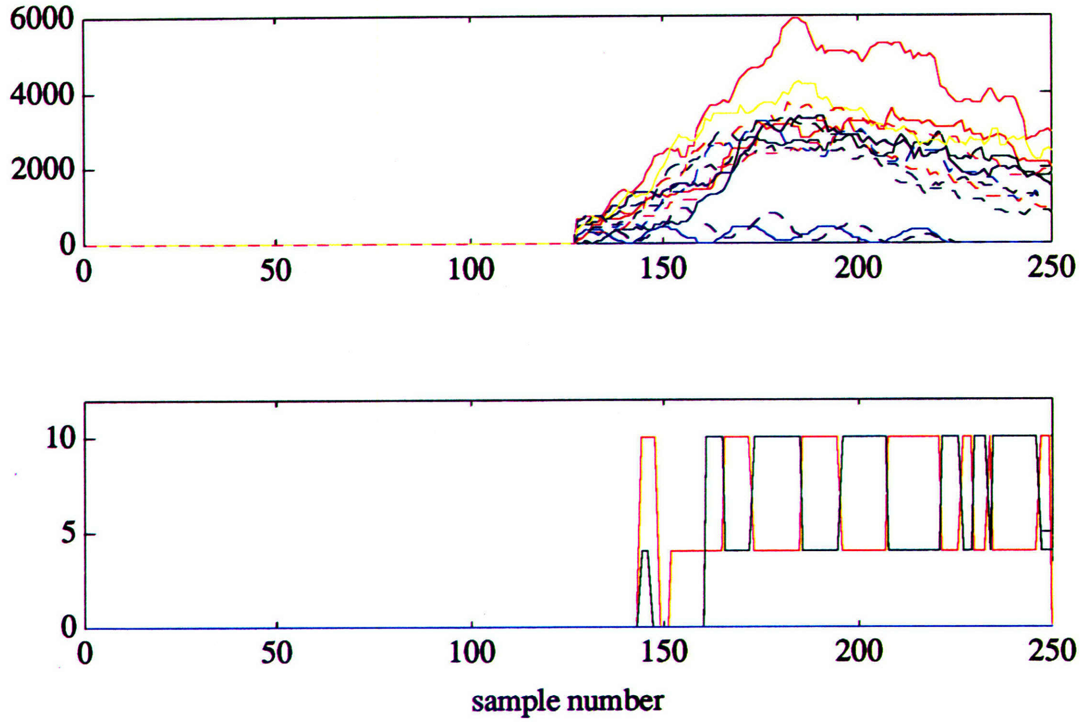


Figure S-21 PSD Test for Simultaneous Pegged Failures

5.3.6 Experiments with Single Zero Failures

Finally, figures S-22 through S-25 show each mechanism's response to failures to zero in sensor 4. In the parity tests, the familiar light blue line which dominated the tops of all previous plots is nowhere to be seen; on the contrary, it seems to sit at the bottom of the pile, hovering lower than most others. Neither of the two narrow warning spikes triggered in S-22 and S-23 correspond to sensor 4 degradations. Altering the moving window size does not help either; S-24 employs a 100-sample window size but still no significant errors are apparent.[1]

The failure detection filter performs badly as well; its residuals resemble those for the noisy failure and have not been included. However, the PSD test in S-25 seems to cope quite well with this failure mode, manifesting smooth warnings from samples 375 to 500. For simultaneous failures the results scale similar to those for noisy and pegged failures, and offer little additional information (they are not included here).

1 In this test the compressor had been refitted with 12 sensors as discussed previously, and is implied by the top bound of the warning function.

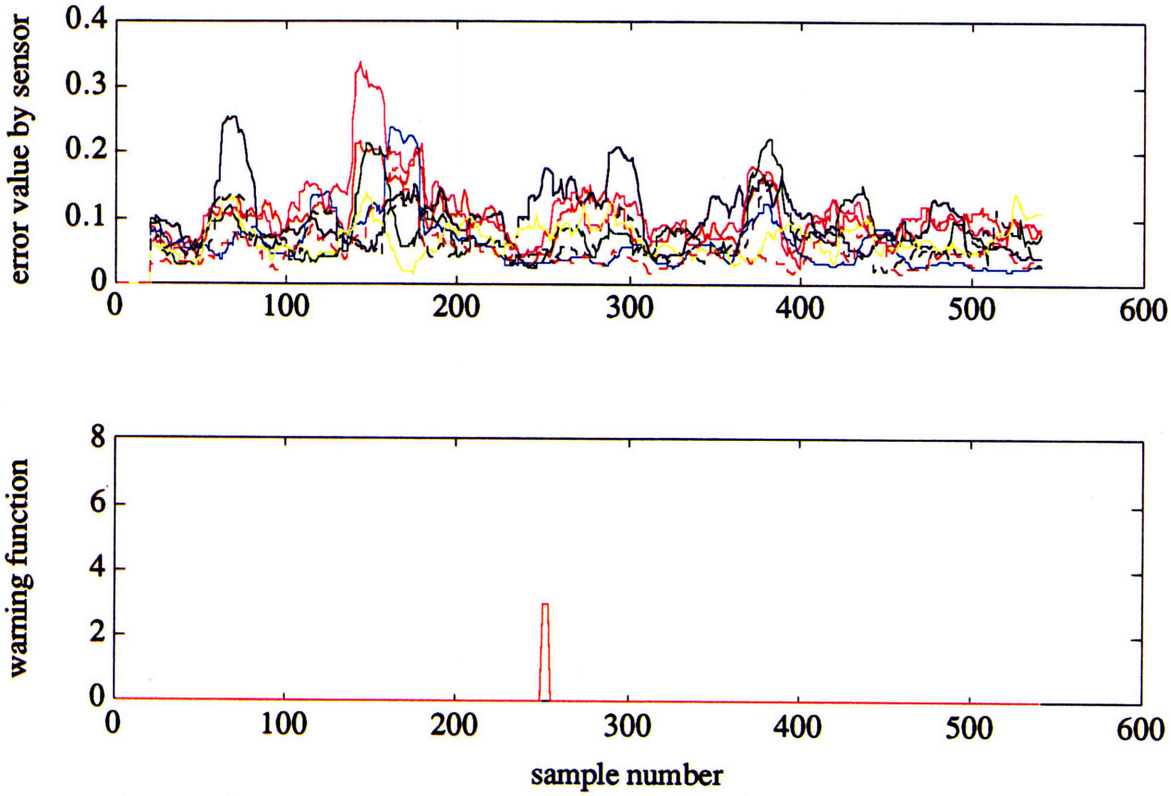


Figure S-22 20-Sample Diffuse Parity Test; mean-value ratio=2.3, relative-value ratio =2

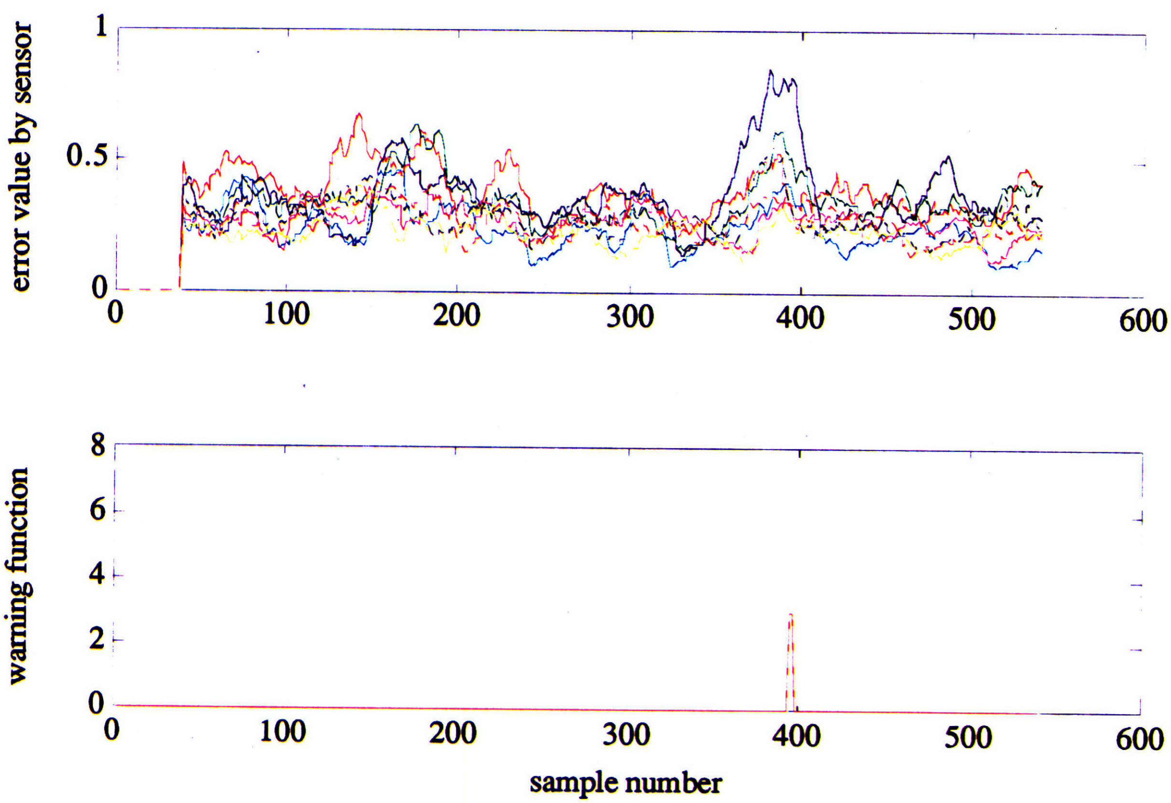


Figure S-23 20-Sample Local Parity Test with Zero Failure

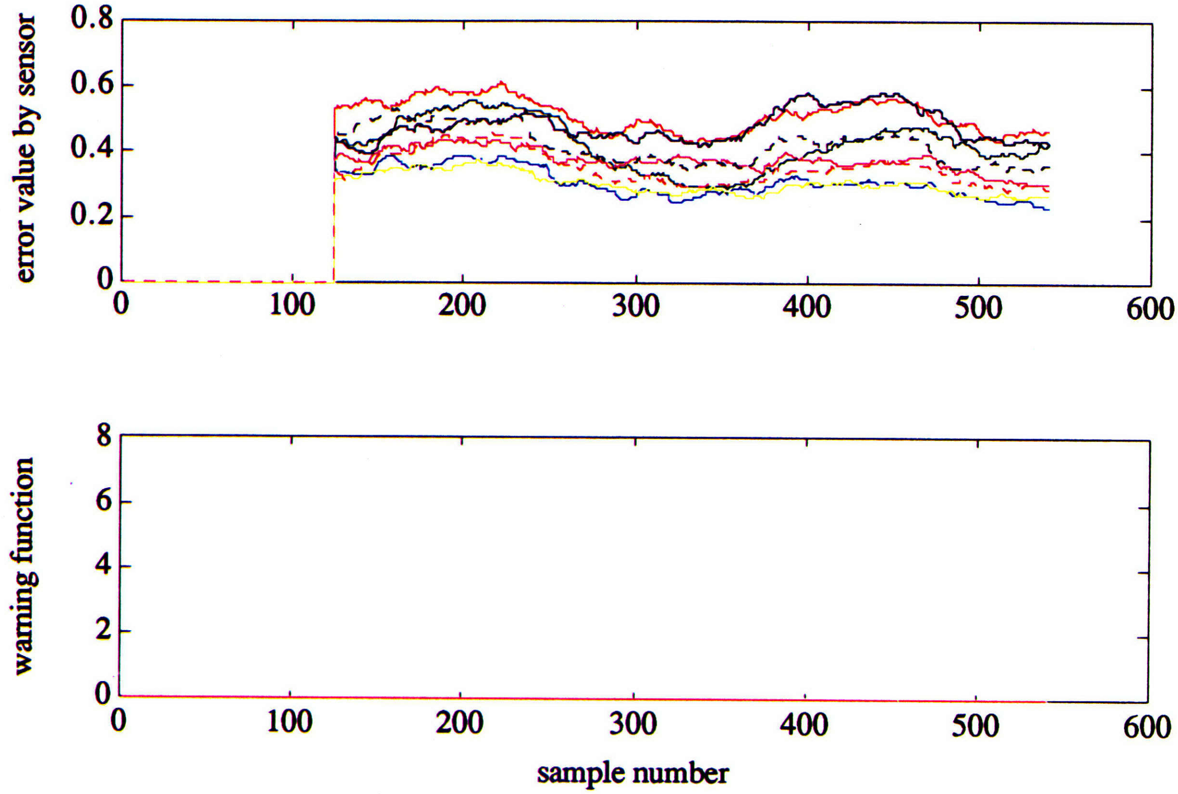


Figure S-24 Four-Poll 100-Sample Local Parity Test for Zero Failure

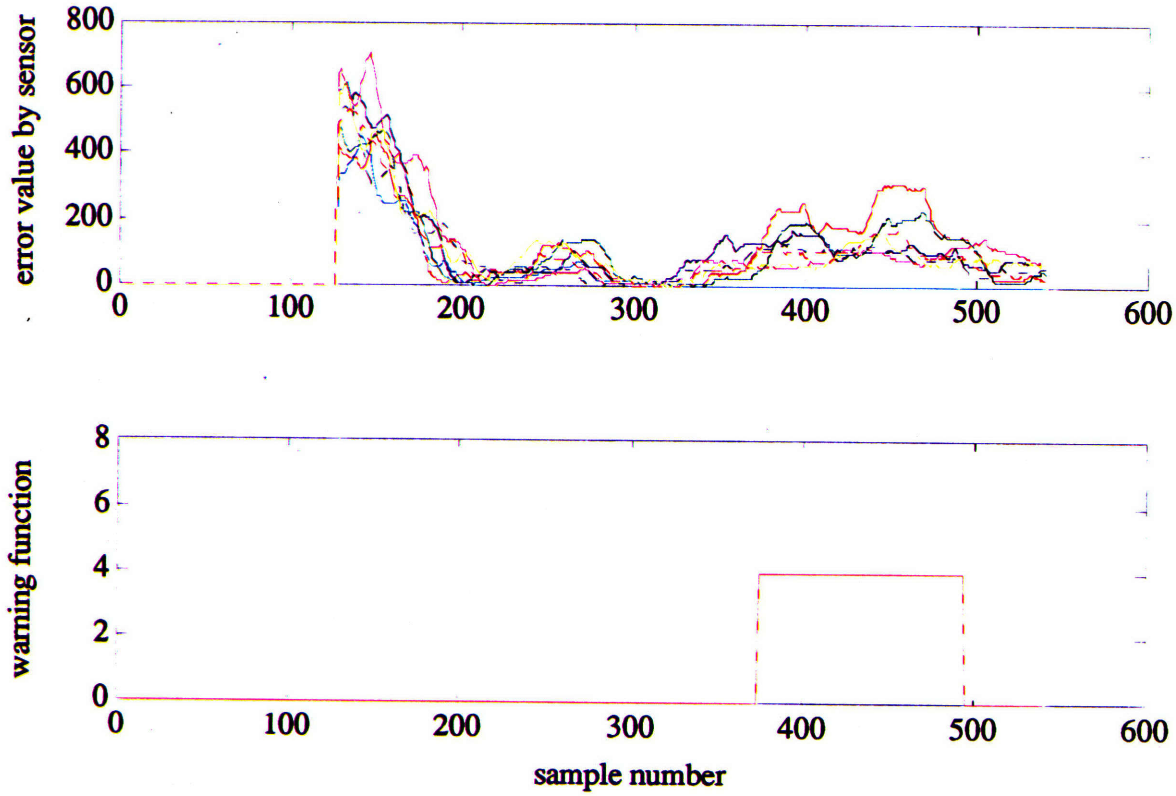


Figure S-25 Power Spectral Density Test, 128-point FFT, Zero Failure

Chapter 6
Conclusions

6.1 Summary of Experimental Results

General observations from the test results approximate closely the performance predicted in designing the detection mechanisms. Direct input-output modelling for each actuator resolves blade FDI to a trivial task. High-speed engine designers that seek to include rotating stall controllers or, more generally, any type of moveable stator vanes would be greatly encouraged to make the actuator outputs readily accessible to the controller, facilitating robust and easily implemented FDI.

Local Parity Test performance scales roughly with the maliciousness of the fault ("maliciousness" here defined in terms of how radically the compressor stability was affected, i.e. pegged faults among the worst, random failures somewhat less damaging, and failures to zero relatively harmless). Experimenting with the number of sensors polled soon revealed the tradeoff between response time and reliability: few polls distorted the adjacent sensor errors too much as the faults infected each estimate too readily; whereas many polls threw the warning function into confusion as the disturbances' nature changed across the domain of the distributed system. Four polls seemed to work best, identifying a noisy failure within 72 milliseconds and a pegged one within 54, impressively below the required 400 ms response time. "Moving window" widths below 20 samples began to show evidence of easy triggering of false alarms; 20 samples seemed to give the best performance, as higher-width moving windows decreased the response time.

Diffuse Parity Test performance scaled similarly with fault maliciousness but, on the whole, responded roughly two thirds quicker than the local mechanisms, at 48 milliseconds for noisy failures and 32 for pegged ones. 40-sample moving window tests had better performance than 20-sample experiments with no significant increase in response time. The diffuse methodology seems to be more susceptible to false alarms; this is most likely due to stall cells and other perturbations affecting only part of the compressor, throwing off global estimates of the Fourier harmonics.

For zero failures, however, neither parity test seemed equipped to detect errors with any reliability, no matter how many sensors were polled or window-sizes attempted.

In general, parity tests have the most difficulty detecting errors that remain within the "operating envelope" of the velocity data. This "envelope" was best demonstrated in Figure 4-6, where the flow speed traces superimpose when viewed in short fragments. Random noise has a better chance, on the average, of exhibiting a direction orthogonal to the predicted flow field, whereas zero failures hover closer to the velocity envelope and defy detection. Pegged errors, on the other hand, remain outside of the velocity envelope for long periods of time and can be identified well within the minimum time response.

Failure Detection Filters work very well in identifying those failures which manifest large errors, but have significant trouble with others such as noisy or zero errors. The performance of these filters depends almost exclusively on their ability to detect consistent deviations along one direction; failures which shift rapidly become very confusing. Furthermore, in cases where the compressor model can closely maintain its estimates, a more likely happening for random and zero failures, the residuals may never rise out of the background noise. The key advantage of this approach, however, is its use of a coherent plant model which is not affected by the operating conditions at any particular point. Wind gusts or sudden changes in throttle would seem to be least likely to affect the failure detection filter.

Power Spectral Density mechanisms seemed the most reliable in detecting failures overall, though often beaten in speed by parity testing. Zero errors showed up quite clearly, easily outdoing any parity test attempts. All failures manifested smooth warning functions with few false alarms or glitches. However, this approach can be limited by computation bandwidth and only approximates the distribution of a narrow band of frequencies. Failures such as bias errors or sudden shifts in phase, whose only components lie outside the frequencies examined, will not be detected. Therefore, a PSD methodology as detailed here will not be sufficient for Byzantine Fault Resilience by itself.

In simultaneous failure experiments parity test performance is considerably degraded, unable to ascertain which component has failed. Expanding the warning function algorithm to cope with multiple large errors helps somewhat to resolve this problem, though in many cases false alarms arise. Power spectral density tests continue to show great promise, at least for failures which do not mimic the high first mode 10-Hertz frequency components of the compressor system.

On the whole, it appears that the mechanisms are useful in identifying arbitrary failures; however, trusting any sole warning function can be dangerous, as each has its "blind spots." All of the failures that were injected in the study of these approaches were detected by one or more algorithms within the required response time of 400 milliseconds, though none of the detection mechanisms has the individual capacity to achieve Byzantine Fault Resilience in tolerating all possible faults.

6.2 Tying Detection Mechanisms Together

The primary advantage in designing several detection mechanisms concurrently is the overlap that is possible with a selection of algorithms, as opposed to the dangerous (and non-Byzantine resilient) "blind spots" evident in employing only one. Measurements incorporated into distinct but conspiring detection mechanisms can cope with different types of failures, whose basic nature can be radically different. Each approach may have, in its domain, a set of failures which it detects best.

After some thought it becomes clear that from first principles this <u>must</u> be the approach taken to distributed system FDI. Some of the compressor components may be experiencing radically different environmental conditions; nevertheless, they are all used equally to determine a comprehensive estimate of the plant dynamics. This coupling leads directly to the conclusion that any smart fault detection must examine the system from both a local perspective, alert for temporary disturbances affecting fragments of the system which may act like failures, and a diffuse one as well, tracking global trends on the lookout for single-point variations and slow degradations.

In other words, a distributed architecture requires a <u>distributed</u> detection methodology. Failure detection could be assigned to multiple mechanisms with the output of each collectively alerting the controller to deviations from predicted responses. This concept is most easily demonstrated by forming a collective warning function from each mechanism's individual output. Since most tests discussed herein all used the same data, their warning functions can be superimposed for each failure. For the single noisy failure tests the function is given in Figure 6-1.

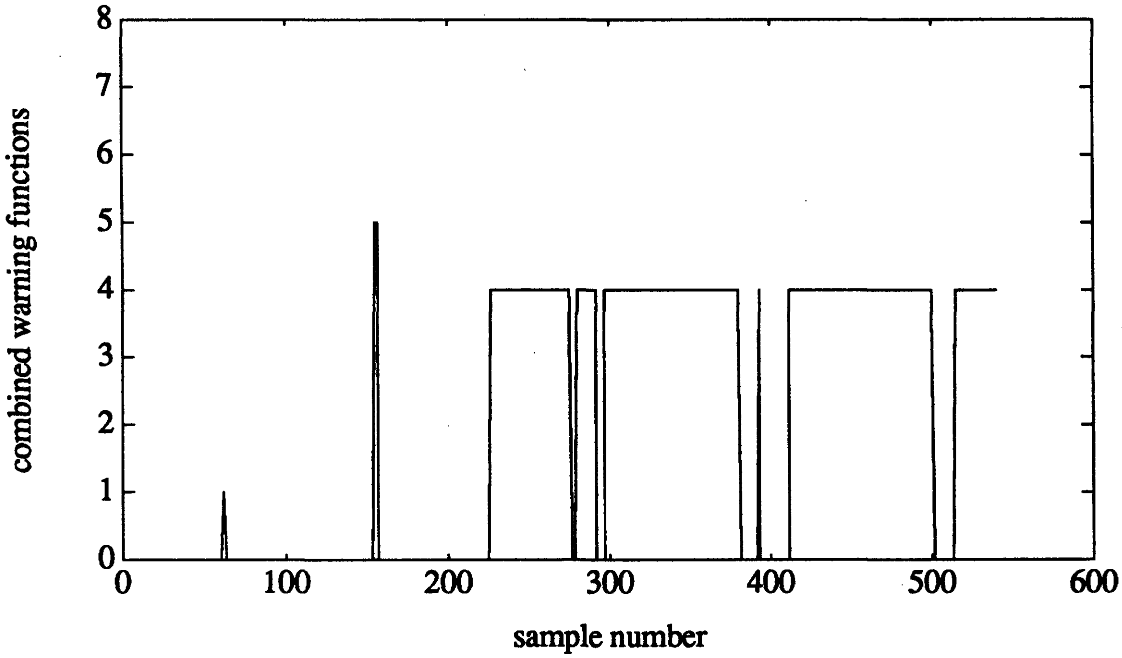


Figure 6-1 Combined Warning Functions of Local, Diffuse and Power Spectral Density Tests for a Single Noisy Failure

Here all three warning functions overlap, both with correct and false alerts, into a single curve.

The combined output for the pegged error is shown in Figure 6-2:

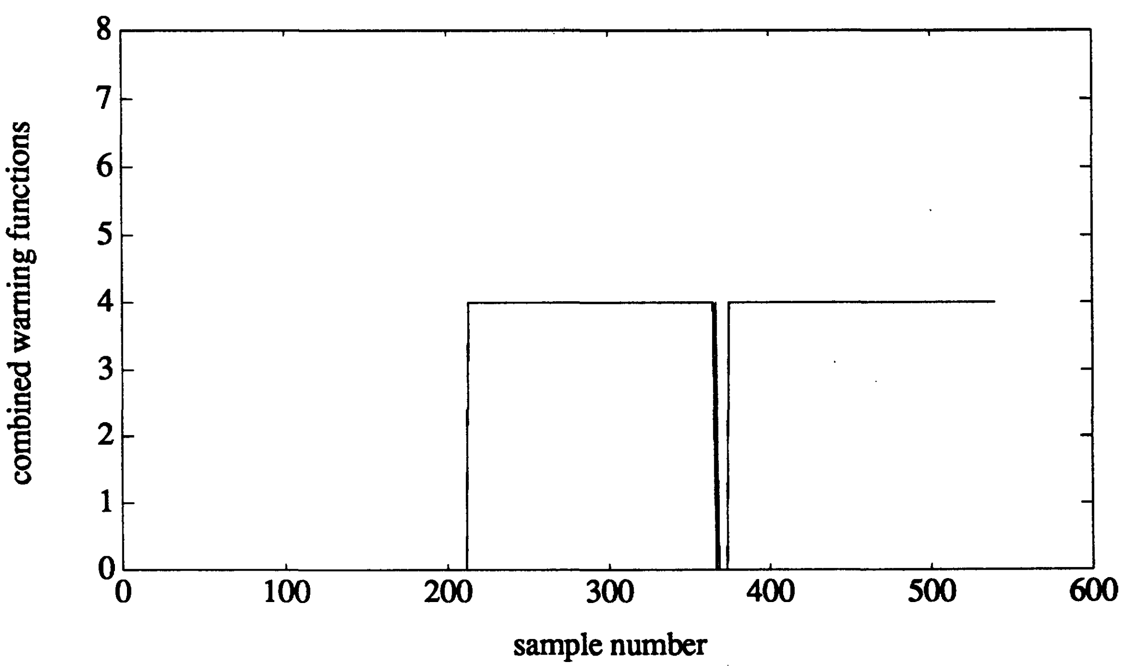


Figure 6-2 Combined Warning Functions of Local, Diffuse and Power Spectral Density Tests for Pegged Failure

The performance of the combined warning functions is clearly superior than any individual analysis, and seems to minimize each's limitations by overlapping blind spots and manifesting no false alarms.

These are clearly the simplest of methods for coming to a group consensus of sensor failure; a better algorithm might announce faults when two out of three mechanisms agree on its existence. For multiple failures such a process must be conducted, due to the frequency of false alarms present in most of the experiments. Figure 6-3 shows the end product warning function for the simultaneous pegged case, which impressively manifests no false alarms whatsoever. To progress from several mechanisms with at best glitchy outputs to this smooth curve in a single simple step strongly supports the advantage of combining several detection mechanisms, especially for difficult-to-detect multiple failures. Note that the response is well within the required 200 sample limit.

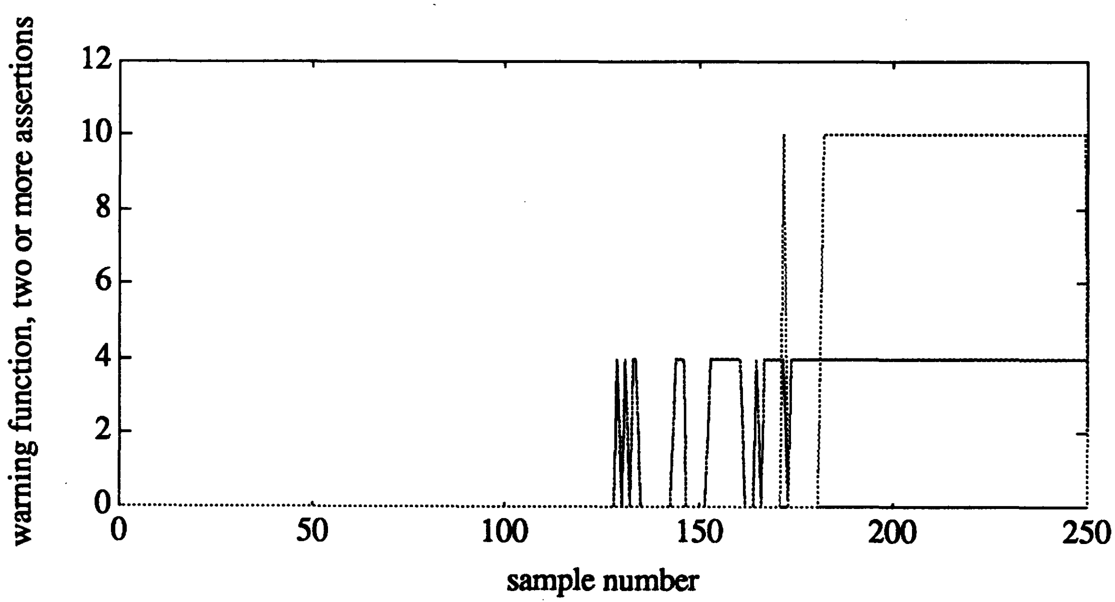


Figure 6-3 Combined Warning Functions for Simultaneous Pegged Failures

Once again, this function interleaving involves a very simple algorithm. Clearly, there may be far superior ones which must be considered for a flight-qualifiable controller.

6.3 Creating an Expert System

A multi-algorithm intelligent methodology with the capacity to synthesize more thorough warnings is best described as an expert system. In the case of fault detection and isolation, an expert system can aid in changing thresholds, making decisions, and establishing confidence factors. The role of an expert system, as a rule of thumb, may include any of the following techniques [Gai89]:

- Adjusting the number of samples based on the severity of the maneuver, thus varying the computation required to the desired confidence level;
- Lowering the probability of false alarm for heavy maneuvers;
- Updating confidence levels based on consecutive/repeated tests;
- Making judgements (when possible) when failure warnings conflict;
- Changing the probability of error for isolation involving residuals with confusing event vectors (failure directions);
- Ascertaining the assurance of performance under reconfiguration;
- Computing the amount of time left to restructure the system before the errors become unrecoverable.

The ultimate goal of an expert system would be to provide a level of diagnostic performance comparable to that of an experienced domain expert. One diagnostic process which is perfectly suited to this thesis has been adapted from [Die89] and is illustrated in Figure 6-4.

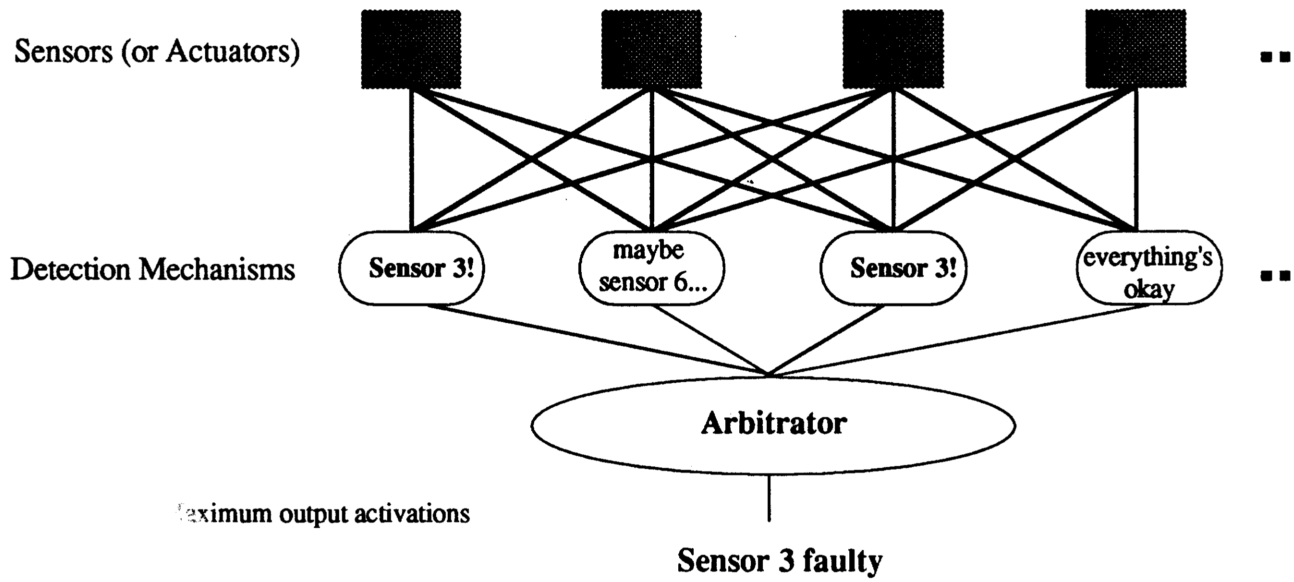


Figure 6-4 Diagnostic Tree Fault Identification

An expert system for the engine compressor would rely on classifiers such as the mean-, relative- and absolute-value comparisons which determine the fault condition as indicated by the particular sensor or detection mechanism. The output of each classifier is then entered into an arbitrator, which determines the output of the diagnostic system as a whole. In the voting examples outlined previously, the arbitrator would simply add all of the mechanism outputs together, or form a two-out-of-three vote. This design combines the advantages of multiple detection mechanisms while allowing arbitrator weights to mask their limitations; it is also suited well to parallelization which may be necessary to achieve the throughput required for multiple fault identification schemes. There are other assessments that networks like this can perform, such as fault severity, but this is one of the most relevant examples suited to the engine compressor. Clearly, however, these techniques have the power to categorize on the basis of qualitative features, such as flowfield shape, as well as quantitative features, such as magnitude.

Despite the promising performance of a unified, distributed detection methodology, none of the mechanism results, individual or combined, are sufficient to guarantee complete Byzantine Fault Resilience, even considering the hundreds of such tests that were performed herein (only several of which have been described). Such a validation of a detection system would require more scrutiny by injecting any conceivable failure at many varied operating conditions. On the other hand, further study in the mathematical domain may instead be used to establish probabilities of detection and isolation, or guarantee fault resilience. Model validation is one of the main requirements for the design of robust controllers in general; these methods have clearly proved worthy of further investigation should a distributed

architecture be implemented. Furthermore, this methodology is not necessarily limited by low-speed jet engine assumptions, so that control of multi-stage and high speed machines can be similarly examined with identical issues in mind.

Recommendations for Future Work

The compressor in the Gas Turbine Laboratory currently runs under three-mode control; in this report only one-mode control was discussed in any detail. Including the higher harmonic control would require extensions in each detection mechanism, and likely complicate error signal threshold selection. However, since second and third mode [illegible]e into play, robust fault detection may require their incorporation into some [illegible]tion methods.

[illegible] effort was devoted to reconfiguration, but it merits some mention at this point. Simple and elegant procedures for reconfiguring actuator and sensor failures amidst the distributed system exist. A likely scheme for actuators would be shutting down the power to the blade or attempting to align it with the average IGV angle. On the other hand, sensor measurements corrupt the controller algorithm directly and must be removed from the controller's flowfield estimation. However, recomposing Fourier Harmonics with partial information has already been discussed in section 4.3.2 and can be used in the same way to calculate the control inputs. The row of the full-state DFT matrix which specifies the failed sensor's influence on the Fourier harmonics is set to zero, and the other rows are scaled to compensate for the lost input. The detection algorithms must consequently be altered to cope with a missing measurement.

For successive failures, these procedures can be repeated each time to prevent a component from corrupting the controller. This is especially important as opportunities for compressor maintenance may be sporadic, if available at all, and the compressor may suffer repeated damage during a single mission. These factors, along with the minimum number of working components required for stability, must be considered when specifying the size of the distributed architecture. A three-mode controller needs at least eight flow sensors to avoid spatial aliasing; engine designers may require the compressor to function when, say, three sensors are inoperative; so the system is established as utilizing at least eleven velocity sensors. Furthermore, if a working component is surrounded by two disfunctional ones, FDI using local parity tests may be unreliable due to a "dead zone" in adjacency, requiring that the working sensor be taken off-line as well. Issues like this will clearly drive the design of the distributed system as equally as requirements for single-failure detection and isolation.

Appendix A
MATLAB Programs

The following programs constitute the primary simulation tools that were used for this thesis and are provided for reference. The first line of each subroutine, and their names when used elsewhere, are written in **bold**. Comments have been written in after percent signs (which MATLAB skips over, naturally) and at the end of some lines. All commands not in **bold face** can be found in any MATLAB package.

A.1 Actuator Detection Mechanism Algorithms

```
FUNCTION  [U,WFUN]=ACTPAR(BCM,BAC,BAND,F_TOL,C_TOL) ;

%
%  THIS FUNCTION PERFORMS A SIMPLE ONE-TO-ONE PARITY TEST FROM ACTUATOR
%  INPUT TO OUTPUT MODELLING EACH BLADE TRANSFER FUNCTION AS A
%  GAIN-DELAY.  COMPUTES A "MOVING WINDOW" OF ERROR
%  VALUES , NAMELY DISCREPANCIES BETWEEN THE ESTIMATED BLADE OUTPUTS
%  (EBAC, BLADE COMMANDS BCM) AND THE ACTUAL BLADE
%  OUTPUTS (BAC).
%

[A,B,C,D]=GETSHIFT(2);                              GENERATE BESSEL FUNCTION
[ROW,COL]=SIZE(BCM);
T=0:.002:(ROW-1)*.002';                             GET INCREMENTED TIME SCALE

IF NARGIN==2,
  BAND=20;                                          SET MOVING WINDOW SIZE
END                                                   IF USER DOES NOT SPECIFY

IF NARGIN<4,                                        SET MEAN-VALUE AND RELATIVE-
  F_TOL=2;                                            VALUE COMPARISON CONSTANTS
  C_TOL=1.5;
END

FOR I=1:COL,
  EBAC(:,I)=SHIFT(LSIM(A,B,C,D,BCM(:,I),T),2);     COMPUTE INITIAL ESTIMATED BLADE
END                                                   OUTPUTS AND SHIFT

ERR=BAC([1:BAND+3],:)-EBAC([1:BAND+3],:);           COMPUTE INITIAL DISCREPANCY
                                                      ERROR

EAB=(ABS(ERR)).^2;                                  GENERATE INITIAL MEAN ERROR

FOR I=1:COL,
  U(BAND+3,I)=SUM(EAB([4:BAND+3],I))/BAND;          TOTAL ERRORS FOR "MOVING
END                                                   WINDOW"

FOR J=BAND+4:ROW,                                   REPEAT ABOVE STEPS IN
ERR(J,:)=BAC(J,:)-EBAC(J,:);                          CONTINUOUS LOOP UNTIL
  EAB(J,:)=(ABS(ERR(J,:))).^2;                        END OF DATA STREAM
  FOR I=1:COL,
```

```
    U(J,I)=U(J-1,I)+(EAB(J,I)-EAB(J-BAND,I))/BAND;
  END
  [US,UI]=SORT(U(J,:));                              SORT ERROR BY SIZE
  IF (US(1,COL)>F_TOL*MEAN(US))
      &(US(1,COL)>C_TOL*US(1,COL-1)),                COMPARE TO
      WFUN(J,1)=UI(1,COL);                             CONSTANTS AND
                                                       COMPUTE WARNING
  END                                                  FUNCTION
END

[RF,CF]=SIZE(WFUN);
IF RF~=ROW,                                          FILL OUT WARNING FUNCTION TO
  WFUN(ROW,1)=0;                                       END OF DATA STREAM
END

HOLD OFF;
CLG;
SUBPLOT(211),PLOT(U),SUBPLOT(212),PLOT(WFUN);        PLOT ERROR AND WARNING
GRID;                                                  FUNCTION
SUBPLOT(111);
```

FUNCTION [A,B,C,D] = GETSHIFT(DEL)

```
% GENERATES DELAY SHIFT MODEL

A=-1000/DEL;
B=1;
C=2000/DEL;
D=-1;
```

A.2 Parity Test Programs

FUNCTION [VEC,SIG]=OPTSHIFT(IN,MAXSHIFT)

```
%
% OPTSHIFT TIME-SHIFTS THE COLUMNS OF MATRIX IN AND COMPARES THEM TO GENERATE
% THE OVERALL ERROR.  COMPUTES THE SHIFT BETWEEN 1 AND MAXSHIFT WHICH RETURNS
% THE MINIMUM ERROR.  USE FOR VECTORS WHICH ARE CONTINUOUSLY SHIFTED WITH
% RESPECT TO EACH OTHER.  (ALSO RETURNS SHIFTED MATRIX.)
%

[ROW,COL]=SIZE(IN);
VAR=[];
SIG=[];
FOR I=1:MAXSHIFT,
  ERR=[];
  FOR J=1:(COL-1),
    FOR K=1:(ROW-I),
      ERR(K,J)=IN(K,J)-IN((K+I),(J+1));              COMPUTE SHIFT ERROR
    END
  END
  SIG(1,I)=SUM(MEAN(ABS(ERR)));                      SUM INDIVIDUAL ERRORS FOR TOTAL
END
[M,IND]=MIN(SIG);                                    SORT ERRORS BY SIZE
```

```
FPRINTF('OPTIMAL SHIFT IS %G AT %G ERROR\N',IND,M);      PRINT SMALLEST

VEC=[];
FOR I=1:COL,
  FOR J=1:(ROW-((COL-1)*IND)),
    VEC(J,I)=IN((J+((I-1)*IND)),I);                      RETURN SHIFTED MATRIX
  END
END
```

```
FUNCTION  [VEC]=SHIFT(IN,SHIFT)

%
%  SHIFTS MATRIX COLUMNS BACKWARD [SHIFT] SPACES WITH RESPECT TO
%  THE PREVIOUS COLUMN.  SHIFTS COLUMNS FORWARD [SHIFT] SPACES.
%

[ROW,COL]=SIZE(IN);

IF COL==1,
  IF SHIFT>=0,
    VEC=[ZEROS(MIN([SHIFT ROW]),1);IN([1:ROW-SHIFT],:)];
  ELSE
    VEC=[IN(-SHIFT+1:ROW);ZEROS(MIN([-SHIFT ROW]),1)];
  END
ELSE
  FOR I=1:COL,
    FOR J=1:(ROW-((COL-1)*SHIFT)),
      VEC(J,I)=IN((J+(I-1)*SHIFT),I);
    END
  END
END
```

```
FUNCTION  [U,WFUN]=DIFFUSE(IN,BAND,F_TOL,C_TOL,A_TOL);

%
%  DIFFUSE PERFORMS A DIFFUSE PARITY TEST ON THE MATRIX IN WITH A MOVING WINDOW OF
%  SIZE BAND AND MEAN-VALUE, RELATIVE-VALUE AND ABSOLUTE-VALUE COEFFICIENTS
%  OF F_TOL, C_TOL AND A_TOL RESPECTIVELY.  RETURNS OVERALL ERROR U AND WARNING
[illegible]UN.


[illegible]==1,
  BAND=20;                                               SET WINDOW SIZE IF USER DOES
END                                                        NOT SPECIFY

IF NARGIN<3,
  F_TOL=2;                                               SET COEFFICIENTS IF USER DOES
  C_TOL=1.5;                                               NOT SPECIFY
  A_TOL=.07;
END

[ROW,COL]=SIZE(IN);

QQ=(COL-4)/2+1;                                          CALCULATE FOURIER MATRIX FOR
M=MAKEFD(COL)*MAKEFR(COL,0);                               DIFFUSE ESTIMATIONS OF
M=(QQ+1)/QQ*(M-DIAG(DIAG(M,0)));                           INDIVIDUAL VARIABLES

ERR=((M-EYE(COL))*IN([1:BAND+1],:)')';                   COMPUTE INITIAL ERROR
```

```
EAB=(ABS(ERR)).^2;                                        COMPUTE ABSOLUTE ERROR

FOR I=1:COL,
  U(BAND+1,I)=SUM(EAB([1:BAND+1],I))/BAND;                INITIAL MOVING WINDOW ERROR
END

FOR J=BAND+2:ROW,                                         REPEAT THE ABOVE STEPS FOR
ERR(J,:)=((M-EYE(COL))*IN(J,:)')';                          EACH SAMPLE
  EAB(J,:)=(ABS(ERR(J,:))).^2;
  FOR I=1:COL,
    U(J,I)=U(J-1,I)+(EAB(J,I)-EAB(J-BAND,I))/BAND;
  END
  [US,UI]=SORT(U(J,:));                                   SORT ERROR TO GET MAXIMUM
                                                          COMPUTE WARNING FUNCTION
                                                          ALSO CHANGE M MATRIX IF
                                                            NEEDED TO REMOVE
                                                            FAULTY COMPONENT

  IF (US(1,COL)>F_TOL*MEAN(US))&(US(1,COL)>C_TOL*US(1,COL-1))&(US(1,COL)>A_TOL),
      WFUN(J,1)=UI(1,COL);
%       M=MAKEFD(COL)*MAKEFR(COL,UI(1,COL));
%       M=4/3*(M-DIAG(DIAG(M,0)));
  END
END

[RF,CF]=SIZE(WFUN);
IF RF~=ROW,
  WFUN(ROW,1)=0;                                          FILL OUT WARNING FUNCTION
END                                                         SO PLOTS WILL OVERLAP

HOLD OFF,CLG,SUBPLOT(111);
SUBPLOT(211),PLOT(U),SUBPLOT(212),AXIS([0 ROW 0 COL]),PLOT(WFUN);  PLOT
GRID,XLABEL('SAMPLE NUMBER');


FUNCTION  [S,WFUN]=LOCAL(IN,BAND,F_TOL,C_TOL)

%
%  LOCAL PERFORMS A LOCAL PARITY TEST WITH THREE ADJACENT SENSORS POLLED.
%  SEE DIFFUSE FOR DETAILS.
%

SHIFT=6;                                                  TIME SHIFT TO BE USED (OPTSHIFT)

IF NARGIN==1,
  BAND=20;                                                SPECIFY WINDOW SIZE IF USER
END                                                         DOES NOT
IF NARGIN<3,
  F_TOL=2;                                                SPECIFY COMPARISON COEFS
  C_TOL=1.5;                                                IF USER DOES NOT
END

[ROW,COL]=SIZE(IN);

FOR J=1+SHIFT:BAND+SHIFT,
  FOR I=1:COL,
    ERRP(J,I)=IN(J,I)-IN((J-SHIFT),MOD(I-1,COL));         COMPUTE ADJACENT
    ERRF(J,I)=IN(J,I)-IN((J+SHIFT),MOD(I+1,COL));           POLLS
```

```
    ERRFF(J,I)=IN(J,I)-IN((J+2*SHIFT),MOD(I+2,COL));
  END
END
PS=SUM(ABS(ERRP([SHIFT:BAND+SHIFT],:)).^2)/BAND;      COMPUTE ADJACENT
FS=SUM(ABS(ERRF([SHIFT:BAND+SHIFT],:)).^2)/BAND;        ERROR ESTIMATES
FFS=SUM(ABS(ERRFF([SHIFT:BAND+SHIFT],:)).^2)/BAND;

FOR J=BAND+SHIFT+1:(ROW-2*SHIFT),
  FOR I=1:COL,
    ERRP(J,I)=IN(J,I)-IN((J-SHIFT),MOD(I-1,COL));   REPEAT THE ABOVE STEPS FOR
    ERRF(J,I)=IN(J,I)-IN((J+SHIFT),MOD(I+1,COL));     EACHSAMPLE
    ERRFF(J,I)=IN(J,I)-IN((J+2*SHIFT),MOD(I+2,COL));
  END
  PS=PS+(ABS(ERRP(J,:)).^2-ABS(ERRP(J-BAND-1,:)).^2)/BAND;
  FS=FS+(ABS(ERRF(J,:)).^2-ABS(ERRF(J-BAND-1,:)).^2)/BAND;
  FFS=FFS+(ABS(ERRFF(J,:)).^2-ABS(ERRFF(J-BAND-1,:)).^2)/BAND;

  FOR K=1:COL,
    S(J,K)=PS(:,MOD(K+1,COL))+FS(:,MOD(K-1,COL))+FFS(:,MOD(K-2,COL));

                                                    GET TOTAL ERROR
  END
  [SS,SI]=SORT(S(J,:));                                         SORT BY SIZE
  IF (SS(1,COL)>F_TOL*MEAN(SS))&(SS(1,COL)>C_TOL*SS(1,COL-1)),       COMPARE
      WFUN(J,1)=SI(1,COL);                                      DEFINE WFUN
%       M=MAKEFD(8)*MAKEFR(8,SI(1,COL));            RECOMPUTE M MATRIX TO
%       M=4/3*(M-DIAG(DIAG(M,0)));                    REMOVE FAULTY
                                                      COMPONENTS
    END
  END
END

S=[ZEROS(12,8);S];                                  !!!  ADD DELAY TO ERROR OUTPUT
WFUN=[ZEROS(12,1);WFUN];                              TO COMPENSATE FOR NON-REAL
                                                      TIME SHIFTING ALGORITHM !!!
[RF,CF]=SIZE(WFUN);
IF RF~=ROW,
  WFUN(ROW,1)=0;                                    FILL OUT WFUN SO PLOTS OVERLAP
END

HOLD OFF,CLG,SUBPLOT(111);
SUBPLOT(211),PLOT(S),SUBPLOT(212),AXIS([0 600 0 8]);PLOT(WFUN);           PLOT
GRID;XLABEL('SAMPLE NUMBER');


FUNCTION  [S,WFUN]=LOCAL4(IN,BAND,F_TOL,C_TOL)

%
%  LOCAL4 PERFORMS A LOCAL PARITY TEST WITH FOUR ADJACENT COMPONENTS POLLED.
%  SEE LOCAL FOR ADDITIONAL DETAILS; THE PROGRAMS ARE VIRTUALLY IDENTICAL.
%

SHIFT=6;

IF NARGIN==1,
  BAND=20;
END
IF NARGIN<3,
  F_TOL=2;
```

```
  C_TOL=1.5;
END


[ROW,COL]=SIZE(IN);

FOR J=1+2*SHIFT:BAND+2*SHIFT,
  FOR I=1:COL,
    ERRPP(J,I)=IN(J,I)-IN((J-2*SHIFT),MOD(I-2,COL));  ADDITIONAL POLL ADDED HERE
    ERRP(J,I)=IN(J,I)-IN((J-SHIFT),MOD(I-1,COL));
    ERRF(J,I)=IN(J,I)-IN((J+SHIFT),MOD(I+1,COL));
    ERRFF(J,I)=IN(J,I)-IN((J+2*SHIFT),MOD(I+2,COL));
  END
END
PPS=SUM(ABS(ERRPP).^2)/BAND;                            ADDITIONAL ERROR ADDED HERE
PS=SUM(ABS(ERRP).^2)/BAND;
FS=SUM(ABS(ERRF).^2)/BAND;
FFS=SUM(ABS(ERRFF).^2)/BAND;

FOR J=BAND+2*SHIFT+1:(ROW-2*SHIFT),
  FOR I=1:COL,
    ERRPP(J,I)=IN(J,I)-IN((J-2*SHIFT),MOD(I-2,COL));
    ERRP(J,I)=IN(J,I)-IN((J-SHIFT),MOD(I-1,COL));
    ERRF(J,I)=IN(J,I)-IN((J+SHIFT),MOD(I+1,COL));
    ERRFF(J,I)=IN(J,I)-IN((J+2*SHIFT),MOD(I+2,COL));
  END
  PPS=PPS+(ABS(ERRPP(J,:)).^2-ABS(ERRPP(J-BAND-1,:)).^2)/BAND;
  PS=PS+(ABS(ERRP(J,:)).^2-ABS(ERRP(J-BAND-1,:)).^2)/BAND;
  FS=FS+(ABS(ERRF(J,:)).^2-ABS(ERRF(J-BAND-1,:)).^2)/BAND;
  FFS=FFS+(ABS(ERRFF(J,:)).^2-ABS(ERRFF(J-BAND-1,:)).^2)/BAND;

  FOR K=1:COL,
    S(J,K)=PS(:,MOD(K+1,COL))+FS(:,MOD(K-1,COL));
    S(J,K)=S(J,K)+PPS(:,MOD(K+2,COL))+FFS(:,MOD(K-2,COL));
  END
  [SS,SI]=SORT(S(J,:));
  IF (SS(1,COL)>F_TOL*MEAN(SS))&(SS(1,COL)>C_TOL*SS(1,COL-1)),
      WFUN(J,1)=SI(1,COL);
%       M=MAKEFD(8)*MAKEFR(8,SI(1,COL));
%       M=4/3*(M-DIAG(DIAG(M,0)));
    END
  END
END

S=[ZEROS(12,COL);S];                                   NO CHANGE IN REAL-TIME
WFUN=[ZEROS(12,1);WFUN];                                 ADJUSTMENT NECESSARY SINCE
                                                         LOCAL4 LOOKS FURTHER BACK
[RF,CF]=SIZE(WFUN);                                      IN TIME
IF RF~=ROW,
  WFUN(ROW,1)=0;
END

HOLD OFF,CLG,SUBPLOT(111);
SUBPLOT(211),PLOT(S),SUBPLOT(212),AXIS([0 ROW 0 COL]);PLOT(WFUN);
GRID;XLABEL('SAMPLE NUMBER');


FUNCTION  [S,WFUN]=LOCAL5(IN,BAND,F_TOL,C_TOL)
```

```
%
%  LOCAL5 PERFORMS A LOCAL PARITY TEST WITH FIVE ADJACENT COMPONENTS POLLED.
%  SEE LOCAL FOR ADDITIONAL DETAILS; THE PROGRAMS ARE VIRTUALLY IDENTICAL.
%

SHIFT=6;

IF NARGIN==1,
  BAND=20;
END
IF NARGIN<3,
  F_TOL=2;
  C_TOL=1.5;
END


[ROW,COL]=SIZE(IN);

FOR J=1+2*SHIFT:BAND+2*SHIFT,
  FOR I=1:COL,
    ERRPP(J,I)=IN(J,I)-IN((J-2*SHIFT),MOD(I-2,COL));
    ERRP(J,I)=IN(J,I)-IN((J-SHIFT),MOD(I-1,COL));
    ERRF(J,I)=IN(J,I)-IN((J+SHIFT),MOD(I+1,COL));
    ERRFF(J,I)=IN(J,I)-IN((J+2*SHIFT),MOD(I+2,COL));
    ERRFFF(J,I)=IN(J,I)-IN((J+3*SHIFT),MOD(I+3,COL));      ADDITIONAL STATE ADDED
END
END
PPS=SUM(ABS(ERRPP([SHIFT:BAND+SHIFT],:)).^2)/BAND;
PS=SUM(ABS(ERRP([SHIFT:BAND+SHIFT],:)).^2)/BAND;
FS=SUM(ABS(ERRF([SHIFT:BAND+SHIFT],:)).^2)/BAND;
FFS=SUM(ABS(ERRFF([SHIFT:BAND+SHIFT],:)).^2)/BAND;
FFFS=SUM(ABS(ERRFFF([SHIFT:BAND+SHIFT],:)).^2)/BAND;  ADDITIONAL ERROR ADDED HERE

FOR J=BAND+2*SHIFT+1:(ROW-3*SHIFT),
  FOR I=1:COL,
    ERRPP(J,I)=IN(J,I)-IN((J-2*SHIFT),MOD(I-2,COL));
    ERRP(J,I)=IN(J,I)-IN((J-SHIFT),MOD(I-1,COL));
    ERRF(J,I)=IN(J,I)-IN((J+SHIFT),MOD(I+1,COL));
    ERRFF(J,I)=IN(J,I)-IN((J+2*SHIFT),MOD(I+2,COL));
    ERRFFF(J,I)=IN(J,I)-IN((J+3*SHIFT),MOD(I+3,COL));
  END
  PPS=PPS+(ABS(ERRPP(J,:)).^2-ABS(ERRPP(J-BAND-1,:)).^2)/BAND;
  PS=PS+(ABS(ERRP(J,:)).^2-ABS(ERRP(J-BAND-1,:)).^2)/BAND;
  FS=FS+(ABS(ERRF(J,:)).^2-ABS(ERRF(J-BAND-1,:)).^2)/BAND;
  FFS=FFS+(ABS(ERRFF(J,:)).^2-ABS(ERRFF(J-BAND-1,:)).^2)/BAND;
  FFFS=FFFS+(ABS(ERRFFF(J,:)).^2-ABS(ERRFFF(J-BAND-1,:)).^2)/BAND;

  FOR K=1:COL,
    S(J,K)=PS(:,MOD(K+1,COL))+FS(:,MOD(K-1,COL));
    S(J,K)=S(J,K)+PPS(:,MOD(K+2,COL))+FFS(:,MOD(K-2,COL));
    S(J,K)=S(J,K)+FFFS(:,MOD(K+3,COL));
  END
  [SS,SI]=SORT(S(J,:));
  IF  (SS(1,COL)>F_TOL*MEAN(SS))&(SS(1,COL)>C_TOL*SS(1,COL-1)),
      WFUN(J,1)=SI(1,COL);
%       M=MAKEFD(8)*MAKEFR(8,SI(1,COL));
%       M=4/3*(M-DIAG(DIAG(M,0)));
    END
  END
```

```
END

S=[ZEROS(18,8);S];                                ADDITIONAL ADJUSTMENT FOR REAL
WFUN=[ZEROS(18,1);WFUN];                            TIME MATCH WAS NECESSARY AS
                                                    LOCAL5 LOOKED FORWARD IN
[RF,CF]=SIZE(WFUN);                                 TIME
IF RF~=ROW,
  WFUN(ROW,1)=0;
END

HOLD OFF,CLG,SUBPLOT(111);
SUBPLOT(211),PLOT(S),SUBPLOT(212),AXIS([0 600 0 8]);PLOT(WFUN);
GRID;XLABEL('SAMPLE NUMBER');


FUNCTION  A=MOD(X,Y)

%
%  MOD PERFORMS AN ADJUSTED MODULUS FUNCTION TO REWRAP THE COMPRESSOR
%  ANNULUS TOGETHER IN THE PARITY TESTS.  FOR 8 SENSORS, (8+1) BECOMES SENSOR
%  1, (8+2) BECOMES SENSOR 2, (1-2) BECOMES SENSOR 7, ETC.
%

A=REM((Y+REM(X,Y)),Y);
IF A==0,
  A=8;
END


FUNCTION  [FD]=MAKEFD(N)

%
%  THIS FUNCTION GENERATES A FOURIER DECOMPOSITION MATRIX  TO CONVERT
%  VELOCITIES TO FOURIER HARMONIC COEFFICIENTS.  THE MATRIX IS
%  [COS(THETA) SIN(THETA)] WHICH IS 2 BY N;  THETA ANGLES ARE N VALUES
%  EQUALLY SPACED FROM 0 TO 2PI.   OPPOSITE OF MAKEFR.

Q=2*PI/N;
R=2*PI*(N-1)/N;
TH=0:Q:R;
FD=[COS(TH') SIN(TH')];


FUNCTION  [FR]=MAKEFR(N,F)

%
%  THIS FUNCTION USES A LEAST-SQUARE FIT TO A LINEAR REGRESSION TO GENERATE
%  THE FOURIER COEFFICIENTS [A;B] WHICH BEST FIT THE  CURVE ACOS(T)+BSIN(T) TO A
%  COLUMN VECTOR OF POINTS OF LENGTH N.  [A;B] EQUALS FR*X, WHERE X IS THE VECTOR.
%  FR, OF COURSE, IS 2 BY N.  OPPOSITE OF MAKEFD.  ALSO GENERATES BEST CURVE FIT FOR
%  ONE MISSING ELEMENT (I.E. A FAILURE).  F=0 INDICATES NO FAILURE.

T=[0:2*PI/N:2*(N-1)*PI/N]';
T=[T([1:F-1],:);T([F+1:N],:)];

A=[SUM(COS(T).^2) SUM(COS(T).*SIN(T));
   SUM(SIN(T).*COS(T)) SUM(SIN(T).^2)];
B=[COS(T)';SIN(T)'];
FR=REAL(INV(A)*B);
```

```
IF F~=0,
  FR=[FR(:,[1:(F-1)]) [0;0] FR(:,[F:(N-1)])];
END
```

A.3 Failure Detection Filter Algorithms

```
FUNCTION  FS=DFGO(V,UT);

% DFGO USES DFDF, GETWFDF, AND ADDWFDF TO GENERATE FAILURE DETECTION FILTER
% ESTIMATES AND WARNING FUNCTION.

[A,B,C,D]=DFDF(V,UT,1,.8);
[WF1,RP1]=GETWFDF(A,B,C,D,1);
[A,B,C,D]=DFDF(V,UT,0,.7);
[WF2,RP2]=GETWFDF(A,B,C,D,0);
FS= [illegible] RP1,WF2,RP2);
[illegible] 111);
[illegible] S([0 600 0 2.1]),PLOT(WF1),YLABEL('FSIG1, 1=1TO4, 2=5TO8');
[illegible] BEL('FSIG2, 1=ODD, 2=EVEN');
AX [illegible] 600 0 1.1]);PLOT(RP1),YLABEL('RESID. PROXIMITIES, FSIG1');
PLOT(RP2),YLABEL('RESID. PROXIMITIES, FSIG2');
AXIS;


FUNCTION  [EA1,EA2,EB1,EB2]=DFDF(VI,UT,OPTION2,LAM);

% DFDF CREATES THE FAILURE DETECTION FILTER RESIDUALS FOR EACH FILTER SET.
% OPTION1 ASSERTED INCLUDES MOVING WINDOW SUMMATION.
% OPTION2, IF 1(0), COMPUTES ESTIMATES FOR 1TO4/5TO8
%     (ODD/EVEN) SCHEMES.

BAND=40;
OPTION1=1;

IF OPTION2,
  M1=MAKEFR(8,[5 6 7 8]);                       COMPUTE THE RECOMPOSITION MTX'S
  M2=MAKEFR(8,[1 2 3 4]);
ELSE
  M1=MAKEFR(8,[2 4 6 8]);
  M2=MAKEFR(8,[1 3 5 7]);
END

Y1=VI*M1';
Y2=VI*M2';
Y=VI*MAKEFR(8,0)';

[A,B,F]=ENGDAT(.397,1);                         GET ENGINE DATA MODEL

[PHI,GAM1]=C2D(A,B,.002);                       CREATE DIGITAL MODEL
[CRAP,GAM2]=C2D(A,F,.002);
GAM=[GAM1+GAM2/.002 -GAM2/.002];

DF=PHI-LAM*EYE(2);                              GENERATE FILTER MATRIX
%DF=(PHI(1,1)-SQRT(LAM^2-PHI(1,2)^2))*EYE(2);
```

```
YH1=DLSIM([PHI-DF*EYE(2)],[DF GAM],EYE(2),ZEROS(2,6),[Y1 UT]);
YH2=DLSIM([PHI-DF*EYE(2)],[DF GAM],EYE(2),ZEROS(2,6),[Y2 UT]);

IF OPTION1,
  EA1(BAND,:)=SUM(Y1([1:BAND],:)-YH1([1:BAND],:))/BAND;
  EA2(BAND,:)=SUM(Y2([1:BAND],:)-YH1([1:BAND],:))/BAND;
  EB1(BAND,:)=SUM(Y1([1:BAND],:)-YH2([1:BAND],:))/BAND;
  EB2(BAND,:)=SUM(Y2([1:BAND],:)-YH2([1:BAND],:))/BAND;

  FOR Q=BAND+1:LENGTH(VI),
    EA1(Q,:)=EA1(Q-1,:)+(Y1(Q,:)-YH1(Q,:)-Y1(Q-BAND,:)+YH1(Q-BAND,:))/BAND;
    EA2(Q,:)=EA2(Q-1,:)+(Y2(Q,:)-YH1(Q,:)-Y2(Q-BAND,:)+YH1(Q-BAND,:))/BAND;
    EB1(Q,:)=EB1(Q-1,:)+(Y1(Q,:)-YH2(Q,:)-Y1(Q-BAND,:)+YH2(Q-BAND,:))/BAND;
    EB2(Q,:)=EB2(Q-1,:)+(Y2(Q,:)-YH2(Q,:)-Y2(Q-BAND,:)+YH2(Q-BAND,:))/BAND;
  END
ELSE
  EA1=Y1-YH1;
  EA2=Y2-YH1;
  EB1=Y1-YH2;
  EB2=Y2-YH2;
END

SUBPLOT(111);
SUBPLOT(221),PLOT(EA1),YLABEL('EA1');
PLOT(EA2),YLABEL('EA2');
PLOT(EB1),YLABEL('EB1');
PLOT(EB2),YLABEL('EB2');


FUNCTION  [WF,RP]=GETWFDF(EA1,EA2,EB1,EB2,OPTION2)

% GETWFDF GENERATES THE FAILURE AND PROXIMITY SIGNALS FROM THE OUTPUT OF DFDF.
% OPTION2 PERFORMS THE SAME FUNCTION AS IN DFDF.

%F_TOL=3;
F_TOL=2;

MC=MAKEFR(8,0);                                         FAILURE EVENT VECTORS
IF OPTION2,
%  A_TOL=.14;
  A_TOL=.02;
  FOR K=1:4,
    M1(:,K)=MC(:,K)/NORM(MC(:,K));
  END
  M2=M1;
ELSE
%  A_TOL=.06;
  A_TOL=.03;
  FOR K=1:4,
    M1(:,K)=MC(:,(2*K-1))/NORM(MC(:,(2*K-1)));
    M2(:,K)=MC(:,2*K)/NORM(MC(:,2*K));
  END
END

FOR Q=20:LENGTH(EA1),
  IF (NORM(EA1(Q,:))>F_TOL*NORM(EB2(Q,:)))&(NORM(EA1(Q,:))>A_TOL),
    WF(Q,1)=1;
    RP(Q,:)=(EB1(Q,:)*M1/NORM(EB1(Q,:))).^4;
  ELSEIF (NORM(EB2(Q,:))>F_TOL*NORM(EA1(Q,:)))&(NORM(EB2(Q,:))>A_TOL),
```

```
      WF(Q,1)=2;
      RP(Q,:)=(EA2(Q,:)*M2/NORM(EB1(Q,:))).^4;
   ELSE
      WF(Q,1)=0;
      RP(Q,:)=ZEROS(1,4);
   END
END

%HOLD OFF;SUBPLOT(111)
%SUBPLOT(221),PLOT(EB1),YLABEL('EB1');
%PLOT(EB2),YLABEL('EB2');
%PLOT(WF);
%PLOT(RP);
```

FUNCTION FS=ADDWFDF(WF1,RP1,WF2,RP2);

```
% ADDWFDF COMPUTES THE DETECTION FILTER WARNING FUNCTION FROM THE OUTPUT OF GETWFDF.
      [illegible] IS 1-4/5-8.  WF2 IS ODD/EVEN. RP'S MUST BE POSITIVE.

               [illegible] WF1),
   [S, [illegible] [RP1(Q,:)]);
   [S,I2]=SORT([RP2(Q,:)]);
   IF WF1(Q)==1,                                          COMBINES FAILURE SIGNALS
      IF WF2(Q)==1,
         IF (ANY(1==I1(3:4)))&(ANY(1==I2(3:4))),FS(Q)=1;END;
         IF (ANY(3==I1(3:4)))&(ANY(2==I2(3:4))),FS(Q)=3;END;
      ELSEIF WF2(Q)==2,
         IF (ANY(2==I1(3:4)))&(ANY(1==I2(3:4))),FS(Q)=2;END;
         IF (ANY(4==I1(3:4)))&(ANY(2==I2(3:4))),FS(Q)=4;END;
      END
   ELSEIF WF1(Q)==2,
      IF WF2(Q)==1,
         IF (ANY(1==I1(3:4)))&(ANY(3==I2(3:4))),FS(Q)=5;END;
         IF (ANY(3==I1(3:4)))&(ANY(4==I2(3:4))),FS(Q)=7;END;
      ELSEIF WF2(Q)==2,
         IF (ANY(2==I1(3:4)))&(ANY(3==I2(3:4))),FS(Q)=6;END;
         IF (ANY(4==I1(3:4)))&(ANY(4==I2(3:4))),FS(Q)=8;END;
      END
   END
END

FS(LENGTH(WF1))=0;
```

FUNCTION [A,B,F]=ENGDAT(PHI,MODE)

```
% THIS FUNCTION GENERATES THE DATA POINTS SIGMARS, OMEGARS,
% BR,BI,AND GI FOR THE GTL COMPRESSOR ENGINE.  USES A LEAST-SQUARE
% CURVE-FIT TO DATA WITHIN THE FUNCTION DEFINITION.

K=4;
IF MODE==1,
   PHI_DAT=[0.3700    0.3900    0.4100    0.4300    0.4750]';
   SRS_DAT=[10.8750    7.0000    3.2500    0.5000   -6.6638]';
   WRS_DAT=[48.7500   52.0100   55.2500   58.8800   67.4858]';
   BR_DAT=[-3.3000   -3.1400   -3.0200   -3.0700   -4.9801]';
   BI_DAT=[0.7600    0.8680    1.0330    1.2500    2.1386]';
   GI_DAT=[-0.0279   -0.0295   -0.0316   -0.0349   -0.0570]';
```

```
ELSE
  PHI_DAT=[.375 .4 .425 .45 .475 .5 .535 .55]';
  SRS_DAT=[1 -3.5 -13 -26 -38 -51 -75 -93]';
  WRS_DAT=[123 138 143 152 157 164 172 181]';
  BR_DAT=[-7.3 -8 -8.9 -10 -11.3 -12.6 -14.4 -16.4]';
  BI_DAT=[1 1 .6 .2 -.4 -1 -2.15 -3.75]';
  GI_DAT=[-.0575 -.0385 -.0418 -.044 -.047 -.050 -.055 -.065]';
END

STEP=(MAX(PHI_DAT)-MIN(PHI_DAT))/LENGTH(PHI_DAT);
IF (PHI<(MIN(PHI_DAT)-STEP) | PHI>(MAX(PHI_DAT)+STEP)),
  FPRINTF('ENGDAT:  PHI MAY NOT BE ACCURATE -- OUT OF FIT RANGE.\N');
END

[CS,Y,T]=LSQR(PHI_DAT,SRS_DAT,K);
[CW,Y,T]=LSQR(PHI_DAT,WRS_DAT,K);
[CBR,Y,T]=LSQR(PHI_DAT,BR_DAT,K);
[CBI,Y,T]=LSQR(PHI_DAT,BI_DAT,K);
[CG,Y,T]=LSQR(PHI_DAT,GI_DAT,K);

SRS=CS(1,1);
WRS=CW(1,1);
BR=CBR(1,1);
BI=CBI(1,1);
GI=CG(1,1);

FOR N=1:K
  SRS=SRS+CS((N+1),1)*PHI^N;
  WRS=WRS+CW((N+1),1)*PHI^N;
  BR=BR+CBR((N+1),1)*PHI^N;
  BI=BI+CBI((N+1),1)*PHI^N;
  GI=GI+CG((N+1),1)*PHI^N;
END

A=[SRS -WRS;WRS SRS];
B=[BR -BI;BI BR];
F=[0 -GI;GI 0];

%OUT=[SRS;WRS;BR;BI;GI];


FUNCTION  [C,Y,T]=LSQR(U,B,N)

% FOR THE POINTS (U,B) LSQR GENERATES THE LEAST-SQUARE
% CURVE-FIT OF Y TO B FOR THE POLYNOMIAL Y=C1+C2*U+C3*U^2...
% OF ORDER N, N BEING THE HIGHEST ORDER U TERM.  ALSO GENERATES
% OUTPUT Y(T) FOR T=100 POINTS, PLOTS OF Y AND U, AND WARNS
% IF THE APPROXIMATION IS INNACURATE.

EDGE=1;

S=SIZE(U);
S=S(1,1)-1;

A=U.^0;
FOR E=1:N,
  A=[A,U.^E];
END
```

```
% CON=COND(A);
% IF CON>100,
%    FPRINTF('LSQR:  RESULTS MAY BE INACCURATE -- COND(A)= %G\N',CON);
% END

IF N<S,
  C=INV(A'*A)*A'*B;
ELSEIF N==S,
  C=INV(A)*B;
END

IF EDGE==1,
  EDGESIZE=U(2,1)-U(1,1);
ELSEIF EDGE==2,
  EDGESIZE=5*(U(2,1)-U(1,1));
ELSE
  EDGESIZE=0;
END

T1=M[illegible]ZE;
T2=MAX(U)+EDGESIZE;
TSTEP=(T2-T1)/100;
T=T1:TSTEP:T2;
TD=LENGTH(T);
Y=C(1,1)*(ONES(1,TD));
FOR E=1:N,
  Y=Y+C((E+1),1)*T.^E;
END
% PLOT(T,Y,'B',U,B,'O');GRID;PAUSE;
```

A.4 Power Spectral Density Algorithms

```
FUNCTION  [PYY,W]=PSD(IN,T,WMAX,NP)

%
%  COMPUTES POWER SPECTRAL DENSITY FOR VECTOR IN AT SAMPLE RATE T FROM
%  FREQUENCIES 0 TO WMAX.  FFT HAS NP POINTS.
%

[ROW,COL]=SIZE(IN);

IF NARGIN<=3,                                   SPECIFY INPUTS IF USER DOES
                                                  NOT
  NP=256;
  IF NARGIN<=2,
    WMAX=NP/2;
    IF NARGIN==1,
      T=500;
    END
  END
END

IF COL==1,
  Y=FFT(IN,NP);
ELSE
  FOR I=1:COL,
```

```
    Y(:,I)=FFT(IN(:,I),NP);                          COMPUTE FFT
  END
END
PYY=Y.*CONJ(Y);                                      COMPUTE PSD
W=T/NP*2*(0:(NP/2)-1);
PLOT(W(1:WMAX),PYY(1:WMAX,:));
```

FUNCTION [OUT,FSIG]=PSDPAR(IN)

```
%
% PSDPAR CONDUCTS A POWER SPECTRAL DENSITY PARITY TEST UPON VECTOR IN
% AND RETURNS THE ERROR VALUE OUT AND WARNING FUNCTION FSIG.
%

[ROW,COL]=SIZE(IN);
F_TOL=.2;
C_TOL=.25;
A_TOL=75;

Q1=FFT(EYE(128),128);
F128=Q1(3,:);                                        GET 10 HZ PSD VECTOR

FOR J=128:ROW,
  F(J,:)=F128*IN([J-127:J],:);
END
OUT=F.*CONJ(F);                                      COMPUTE PSD

FOR J=16:ROW,
  [OS,OI]=SORT(OUT(J,:));
  IF (OS(1)<OS(2)-50)&(OS(1)<.33*MEAN(OS)),          COMPUTE ERROR AND
    FSIG(J,1)=OI(1);                                   WARNING FN
        ELSEIF (OS(8)>OS(7)+50)&(OS(8)>3*MEAN(OS)),
    FSIG(J,1)=OI(8);
  END
END

IF LENGTH(FSIG)~=ROW,
  FSIG(ROW,1)=0;
END

HOLD OFF;
CLG;
SUBPLOT(211),PLOT(OUT),SUBPLOT(212),AXIS([0 600 0 8])      PLOT
PLOT(FSIG),GRID,XLABEL('SAMPLE NUMBER');
AXIS;
```

A.5 Detection Mechanism Warning Function Generators

FUNCTION WF=GETWFA(S)

```
% GETWFA COMPUTES THE WARNING FUNCTION FOR MULTIPLE ACTUATOR FAILURE TESTS.

[R,C]=SIZE(S);
```

```
FOR J=16:R,
  [SS,SI]=SORT(S(J,:));
  MS=MEAN(SS(1:C-3));
  FOR I=C-2:C,
    IF (SS(I)>2.2*MS)&(SS(I)>2*SS(C-3)),
      WF(J,I-9)=SI(I);
    ELSE
      WF(J,I-9)=0;
         END
  END
END

WF(R,1)=0;

HOLD OFF;
C[illegible]
S[illegible](S),SUBPLOT(212),AXIS([0 600 0 C])
P[illegible]SAMPLE NUMBER');
AXIS;
```

```
FUNCTION  [W1,W2]=GETWFD(S)

% GETWFD COMPUTES THE WARNING FUNCTION FOR THE AUGMENTED SYSTEM OF 12 SENSORS
% FOR MULTIPLE FAILURES ANALYZED WITH DIFFUSE.

F_TOL=1.5;
C_TOL=1.3;
C_TOL2=1.3;
[R,C]=SIZE(S);
CH=C/2;

FOR J=45:R,
  [SS,SI]=SORT(S(J,:));                             SORT BY SIZE
  N1=[MOD(SI(C)+CH,C)];                             IDENTIFY OTHER EFFECTED
  [illegible]MOD(SI(C)+CH,C)];                         COMPONENTS
  [illegible]2)==N1)),C1=SS(C-2);
  [illegible]SI(C-3)==N1)),C1=SS(C-3);
  [illegible](SI(C-4)==N1)),C1=SS(C-4);
  [illegible](C-5);
  END
  IF ~(ANY(SI(C-3)==N2)),C2=SS(C-3);
    ELSEIF ~(ANY(SI(C-4)==N2)),C2=SS(C-4);
    ELSEIF ~(ANY(SI(C-5)==N2)),C2=SS(C-5);
    ELSE C2=SS(C-6);
  END
  IF (SS(C)>F_TOL*MEAN(SS(1:C-2)))&(SS(C)>C_TOL*C1),
   IF (ANY(N1==SI(C-CH-2:C)))&(SS(C)>C_TOL2*SS(N1)),
    W1(J)=SI(C);
   END
  END
  IF (SS(C-1)>F_TOL*MEAN(SS(1:C-2))&(SS(C-1)>C_TOL*C2)),
   IF (ANY(N2==SI(C-CH-2:C)))&(SS(C-1)>C_TOL2*SS(N2)),
    W2(J)=SI(C-1);
   END
  END
END
W1(R)=0;W2(R)=0;
```

```
W1=W1';W2=W2';
HOLD OFF,CLG,SUBPLOT(111);
SUBPLOT(211),PLOT(S);PLOT([W1 W2]);


FUNCTION  [W1,W2]=GETWFL(S)

% GETWFL PERFORMS THE SAME FUNCTION AS GETWFD FOR LOCAL4.

F_TOL=1.7;
C_TOL=1.4;
[R,C]=SIZE(S);

FOR J=45:R,
   [SS,SI]=SORT(S(J,:));
   N1=[MOD(SI(C)+1,C) MOD(SI(C)+2,C) MOD(SI(C)-1,C) MOD(SI(C)-2,C)];
   N2=[MOD(SI(C-1)+1,C) MOD(SI(C-1)+2,C) MOD(SI(C-1)-1,C) MOD(SI(C-1)-2,C)];
   IF ~(ANY(SI(C-1)==N1)),C1=SS(C-1);
     ELSEIF ~(ANY(SI(C-2)==N1)),C1=SS(C-2);
     ELSEIF ~(ANY(SI(C-3)==N1)),C1=SS(C-3);
     ELSEIF ~(ANY(SI(C-4)==N1)),C1=SS(C-4);
     ELSE C1=SS(C-5);
   END
   IF ~(ANY(SI(C-2)==N2)),C2=SS(C-2);
     ELSEIF ~(ANY(SI(C-3)==N2)),C2=SS(C-3);
     ELSEIF ~(ANY(SI(C-4)==N2)),C2=SS(C-4);
     ELSEIF ~(ANY(SI(C-5)==N2)),C2=SS(C-5);
     ELSE C2=SS(C-6);
   END
   IF (SS(C)>F_TOL*MEAN(SS(1:C-2))&(SS(C)>C_TOL*C1)),
     W1(J)=SI(C);
   ELSE
     W1(J)=0;
   END
   IF (SS(C-1)>F_TOL*MEAN(SS(1:C-2))&(SS(C-1)>C_TOL*C2)),
     W2(J)=SI(C-1);
   ELSE
     W2(J)=0;
   END
END

W1=W1';W2=W2';
PLOT([W1 W2]);


FUNCTION  WF=GETWFP(S)

% GETWFP GENERATES THE AUGMENTED WARNING FUNCTIONS FOR PSD TESTS, LIKE GETWFL
% AND GETWFD.

[R,C]=SIZE(S);

FOR J=16:R,
   [SS,SI]=SORT(S(J,:));
   MS=MEAN(SS(4:C-4));
   IF (MS>750),
     FOR I=1:4,
       IF (SS(I)<.33*MS)|(SS(I)>3*MS),
         WF(J,I)=SI(I);
```

```
    ELSE
      WF(J,I)=0;
          END
   END
  END
END

WF(R,1)=0;

HOLD OFF;
CLG;
SUBPLOT(211),PLOT(S),SUBPLOT(212),AXIS([0 R 0 C])
PLOT(WF),XLABEL('SAMPLE NUMBER');
AXIS;
```

Appendix B

Compressor Control Code

This appendix contains the augmented control code for the Gas Turbine Lab's single-stage compressor and is intended to be a reference for further experimentation with the compressor, should any such tests be conducted.

```
      PROGRAM  KIWI

C MAIN PROGRAM FOR ACTIVE CONTROL OF ROTATING STALL.

C THIS VERSION ALLOWS IS A RE-VAMP THAT ALLOWS THE SPACING AND
C NUMBER OF HOT WIRES TO BE ARBITRARY.  THE MATRIX IN SUBROUTINE
C FILEINIT MUST BE CHANGED DEPENDING ON THE HOT WIRE ARRANGEMENT.
C THIS MODE IS ALSO DIFFERENT IN THAT IT ALLOWS Z-CONTROL OF THE
C THIRD SPATIAL MODE.  THERE IS NO CONTROL LAW AVAILABLE FOR THIS MODE

C REVISION 1/28/91 BY J. PADUANO

C REVISION 4/28/91 BY D. SEAL, DRAPER LABS

C VARIABLE DEFINITION
C -------------------------------------------------------------
      PARAMETER (NAX=12)
      IMPLICIT UNDEFINED (A-Z)
      EXTERNAL DASH16, INKEY$, RAN
      REAL*4          RAN
      REAL      YM(2,3), XH(2,3), UC(2,3), UF(2,3), BLDS(12),
     .                PLEVEL,XNMZ,XNMO,XNMT,YNMZ,YNMO,YNMT
      INTEGER         START,ENDM,ENDD, I, J, NTIMES, IAXIS,
     .                IABORT, IOPT, UPDATE, FOPT, FBAND,FDUR
      INTEGER*2      IMODE,PARAM(5),RCODE,
     .                DASH16, KOUNT, INKEY$, KEY
      INTEGER*1      TBLDS(12), EIGHTYHEX(NAX), DUMMY(NAX)
      LOGICAL         FAIL_STATE,QUIT_STATE,SAME
      CHARACTER*80 FNAME
      CHARACTER*1  RESP
      CHARACTER*10 SETUP(5), COMAND

      INCLUDE "D:\NEW_ACRS\COMMONS\PDAT"
      INCLUDE "D:\NEW_ACRS\COMMONS\VELDAT"
      INCLUDE "D:\NEW_ACRS\COMMONS\CAL"
      INCLUDE "D:\NEW_ACRS\COMMONS\BCOMDAT"
      INCLUDE "D:\NEW_ACRS\COMMONS\STRE"

      DATA   SETUP/    'HX;        ','PA0;       ','BG;        ',
     .                   'PD-1;      ','IM;       '/
      DATA EIGHTYHEX/NAX*128/

  120 FORMAT(' ERROR IN A/D : RCODE = ',I10)
```

```
  130 FORMAT(I5)

C BEGINNING OF CODE
C -----------------------------------------------------------

C     CALL INTRO
C     CALL CLS
      CALL FILEINIT

C PARAMETERS FOR THE CALL TO THE A/D -8 CONVERSIONS, INTERNALLY TRIGGERED,
C NON-RECYCLE MODE FOR THE DMA

      IMODE = 21
      PARAM(1)  = NHWS+2
      PARAM(3)  = 1
      PARAM(4)  = 0

C MAIN COMMAND LOOP
    9 CALL SETPHZ
   10 CONTINUE
      IFIRST=.TRUE.

C INITIALIZE COMMAND LOOP STATES AND THE LOW-PASS FILTER VARIABLES

      FAIL_STATE=.FALSE.
      QUIT_STATE=.FALSE.
      YNMZ=0.
      XNMZ=0.
      XNMO=0.
      XNMT=0.
      YNMO=0.
      [illegible]=0.

C [illegible] [illegible]0'S TO ENTER 'POSITION DUMP' MODE, THEN 'INCREMENTAL MODE'
      DO 80 IAXIS=1,NAX
      DO 70 I=1,5
        COMAND=SETUP(I)
        WRITE(*,'(5X,A10,$)') COMAND
        CALL SNDCOM(COMAND,IAXIS)
   70 CONTINUE
      WRITE(*,*) ' '
   80 CONTINUE


C MOTORS ARE AT ABSOLUTE POSITION 0 NOW - INITIALIZE BLADE COMMAND INFO TO
C REFLECT THIS.
      DO 90 I=1,12
      UOLD(I)=0
      ISUM(I)=0
   90 CONTINUE

C START INQUIRIES FOR TYPE OF RUN

       CALL CLS
   91 WRITE(*,'('' NUMBER OF DATA POINTS TO TAKE? '',$)')
      READ(*,130,ERR=91) NTIMES

      PRINT *
      START=0
```

```
      ENDM=0
      ENDD=NTIMES
      IOPT=2

 96   WRITE(*,144)
 144  FORMAT(' SELECT FAILURE TYPE:'/
     .               '  0. NO FAILURE'/
     .               '  1. ACTUATOR ZERO'/
     .               '  2. ACTUATOR RANDOM'/
     .               '  3. ACTUATOR PEGGED'/
     .               '  4. SENSOR ZERO'/
     .               '  5. SENSOR RANDOM'/
     .               '  6. SENSOR PEGGED'/
     .               ' ENTER CHOICE: ',$)
      READ(*,130,ERR=96) FOPT
      IF (FOPT.LT.0 .OR. FOPT.GT.6) GOTO 96

C ASCERTAIN LEVEL TO PEG SENSOR OR ACTUATOR AT (IF A PEG FAILURE)

      IF(FOPT.EQ.3 .OR. FOPT.EQ.6) THEN
 97    WRITE(*,146)
 146   FORMAT(' PEG LEVEL?:',$)
       READ(*,*,ERR=97) PLEVEL
      ENDIF

C ASCERTAIN DURATION OF DESIRED FAILURE

 98   WRITE(*,148)
 148  FORMAT(' FAILURE DURATION?:',$)
      READ(*,130,ERR=98) FBAND
      FDUR=250+FBAND

C INITIALIZE MEAN VELOCITY ESTIMATE BY LETTING THE ESTIMATOR
C RUN FOR ABOUT 5 SECONDS
      CALL VMNINIT
      WRITE(*,'('' HIT <CR> TO START LOOP, D TO SAVE THIS: '',$)')
      READ(*,'(A1)') RESP
      CALL CLS
      IF(RESP.EQ.'D'.OR.RESP.EQ.'D') THEN
      NTIMES=2500
      GOTO 105
      ENDIF

      KOUNT=0
      UC(1,1)=0.
      UC(2,1)=0.

C LOOP-BACK POINT
C ------------------------------------------------------------------
   99 KOUNT=KOUNT+1
      WRITE(*,'(''  .'',$)')
      IABORT=0

C MAIN CONLOOP
C ------------------------------------------------------------------

      DO 100 I=1,NTIMES
      RCODE = DASH16(IMODE, PARAM, VELS, HWCRVS)
      IF (RCODE .NE. 0) WRITE(*, 120) RCODE
```

```
C CHECK TO SEE IF ANY OF THE MOTORS HAS GONE DOWN - IF SO THEN EXECUTE
C A CAREFUL ABORT

       IABORT=MAX(0.,IABORT+VELS(1))
       IF (IABORT .GT. 4000) CALL ABORT(1)
       VELS(1)=IABORT

       CALL DFT_SL(YM,FOPT,I,FDUR,FAIL_STATE,PLEVEL)
       CALL FEEDBACK(YM,UC)
       CALL IDFT(UC,BLDS)

C FAIL BLADE FOUR IF FAILURE IS DESIRED (SENSOR FAILURE IS SIMILAR,
C   AND IS IN DFT.F

       IF (FAIL_STATE.AND.I.GT.250.AND.I.LT.FDUR) THEN
          IF (FOPT.EQ.1) BLDS(4)=0.

C LOW-PASS FILTER RANDOM VALUE TO 50 HZ.  THE NUMERICAL VALUES
C   ARE THE COEFFICIENTS IN THE CHEBYSHEV INCREMENTAL METHOD.

          IF (FOPT.EQ.2) THEN
             XNMZ=28*RAN(5439)-14.
             BLDS(4)=.0301*XNMZ+ .0602*XNMO+.0301*XNMT
     .                                    +1.5223*YNMO-.6427*YNMT
             XNMT=XNMO
             XNMO=XNMZ
             YNMT=YNMO
             YNMO=BLDS(4)
          ENDIF
          IF (FOPT.EQ.3) BLDS(4)=PLEVEL
       ENDIF

       CALL COMMAND(BLDS,TBLDS)
       CALL STORE(I,YM,UC,BLDS,TBLDS)
       IFIRST=.FALSE.
  100  CONTINUE

C END MAIN CONTROL LOOP
C-----------------------------------------------------------------------
       FAIL_STATE=.FALSE.

C ONE KEYSTROKE IS PROCESSED AFTER EACH LOOP COMPLETION, AFTER WHICH
C LOOPING WILL EITHER HALT, CONTINUE UNCHANGED, OR CONTINUE WITH
C FAILURES BEING INJECTED FOR ONE CYCLE

       KEY=INKEY$()

C SIMPLE LOOP BACK OR QUIT CONDITIONS ARE PROCESSED FIRST, TO SAVE TIME
C IF NOTHING TOO FANCY IS BEING DONE - WILL ALLOW ONE ADDITIONAL LOOP
C THROUGH WHICH SHUTS DOWN CONTROL IN THE MIDDLE (AFTER ENDD CYCLES).
       IF (KEY.EQ.0) GOTO 99

C NOW PROCESS KEYSTRIKES WHICH MAY MEAN SOMETHING MORE COMPLICATED
C  LOGICAL VARIABLE FAIL_STATE AND QUIT_STATE ARE PASSED BACK
C  IF THE USER HAS KEYED THE APPROPRIATE BUTTON FOR EACH; THE
C  USER CAN ALSO CHANGE THE FAILURE DURATION TIME (FDUR)

       CALL KEY_SL(KEY,FAIL_STATE,QUIT_STATE,FDUR)
```

```
      IF (QUIT_STATE) GOTO 105

C GO TO TOP OF MAIN CONTROL LOOP  - WITH NEW INSTRUCTIONS, IF THE
C KEYSTRIKE WAS MEANINGFUL
      GOTO 99

C POST PROCESSING
C----------------------------------------------------------------------

  105 CONTINUE
      WRITE(*,*) ' DONE. END FAILURE TIME WAS '
      WRITE(*,*) FDUR

      CALL POWER

C RESET ALL AXES BACK TO NORMAL MODE OF OPERATION (END POSITION DUMP)
      CALL FOLLOW(EIGHTYHEX,DUMMY)
      DO 110 IAXIS=1,NAX
      CALL SNDCOM('PD0;',IAXIS)
      CALL SNDCOM('PA0;',IAXIS)
      CALL SNDCOM('BG;',IAXIS)
  110 CONTINUE

      WRITE(*,*) ' SAVE THIS RUN <Y/N>?'
      READ(*,*) RESP
      IF(RESP.EQ.'N'.OR.RESP.EQ.'N')GOTO 500

      CALL SAVMOR(IOPT)

  500 PRINT *, ' EXECUTE ANOTHER RUN <Y/N>?'
      READ *, RESP
      IF (RESP .NE. 'N' .AND. RESP .NE. 'N') GOTO 10

      WRITE(*,*) ' RESET Z MAG OR PHASE <Y/N>?'
      READ(*,*) RESP
      IF(RESP.NE.'N'.AND.RESP.NE.'N')GOTO 9
      STOP
```

References

[Abl88] T. A. Abler. *A Network Element Based Fault Tolerant Processor.* Master of Science Thesis, Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology, May 1988.

[Arl88] Jean Arlat, Karama Kanoun, and Jean-Claude Laprie. "Dependability Evaluation of Software Fault-Tolerance," *Fault-Tolerant Computing Symposium 18*, June 1988,pp.142-147.

[Avi88] Algirdas Avizienis, Michael Lyu, and Werner Schütz. "In Search of Effective Diversity: A Six-language Study of Fault-tolerant Flight Control Software." *Fault-Tolerant Computing Symposium 18*, June 1988, pp.24-29.

[Avweek89] "Soviets Say Engine Stall Caused Crash of MiG-29 at Le Bourget," *Aviation Week & Space Technology*, 19 June 1989, pp. 32-33.

[Bea71] Richard Vernon Beard. *Failure Accommodation in Linear Systems Through Self-Reorganization*, Doctor of Philosophy Thesis, Massachusetts Institute of Technology, February 1971.

[Bro81] Larry Brock, J. Barton DeWolf, and Albert Hopkins. *A Study of Architectures for Microprocessor Flight Control*, Air Force/Wright Aeronautical Laboratories Report AFWAL-TR-81-3120, October 1981.

[Bur84] Frank Burcham, Jr. and Edward Haering, Jr. *Highly Integrated Digital Engine Control System on an F-15 Airplane*, NASA Technical Memorandum 86040, June 1984.

[Cha88] Steven C. Chapra and Raymond P. Canale. *Numerical Methods for Engineers, 2nd Ed.*, McGraw-Hill, New York, 1988, pp. 402-408.

[Che87] G. T. Chen, E. M. Greitzer, and A. H. Epstein. "Enhancing Compressor Distortion Tolerance by Asymmertic Stator Control," *Proceedings of the 23rd Joint AIAA / SAE / ASME / ASEE Propulsion Conference*, June 1987.

[Che88] Kwang-Ting Cheng, Vishwani D. Agrawal, and Ernest S. Kuh. "A Sequential Circuit Test Generator Using Threshhold-value Simulation." *Fault-Tolerant Computing Symposium 18*, June 1988,pp.24-29.

[Coh89] Gerald C. Cohen and Charles W. Lee. "Methods for Evaluating AIrframe/Propulsion Control System Architectures," *NAECON '89*, pp. 569-575.

[Die89] W. E. Dietz, E. L. Kiech, and M. Ali. "Jet and Rocket Engine Fault Diagnosis in Real Time," *Journal of Neural Network Computing*, Summer 1989, pp. 5-18.

[Gai89] S. A. Gauthier, A. K. Agrawal, and S. C. Shah. "A Real-Time Expert System for Self-Repairing Flight Control," *AIAA Guidance, Navigation and Control Conference*, Boston, August 1989.

[Gar89] Vincent Garnier. *Experimental Investigation of Rotating Waves as a Rotating Inception Indication in Compressors*, GTL Report #198, June 1989.

[Gre86] E. M. Greitzer and F. K. Moore. "A Theory of Post-Stall Transients in Axial Compression Systems" (A two-part article), *Journal of Engineering for Gas Turbines and Power*, Vol. 108, January 1986.

[Har87] Richard E. Harper. *Critical Issues in Ultra-Reliable Parallel Processing,* Doctor of Philosophy Thesis, Massachusetts Institute of Technology, June 1987.

[Har88] Richard Harper, Jaynarayan Lala, and John J. Deyst. *Fault Tolerant Parallel Processor Architecture Overview,* Draper Report CSDL-P-2780, June 1988. (Presented at the 18th Fault-Tolerant Computing Symposium, Tokyo, Japan).

[Has90] Malcolm R. Haskard. "An Experiment in Smart Sensor Design," *Sensors and Actuators,* Vol. A24 No.2, July 1990, pp. 163-169.

[Hut86] S. A. Hutchinson, D. R. Kagey, J. W. Kantowski and M. A. Miller. *"VLSI Fast Fourier Transform Digital Signal Processor Chip,"* Proceedings of the Seventh Digital Avionics Systems Conference, October 1986, pp. 492-495.

[Kim75] C. R. Kime. "Fault tolerant computing: and introduction and perspective." *IEEE Trans. Computers,* C-24(5):457-460, May 1975.

[Kim88] K. H. Kim and J. C. Yoon. "Approaches to Implementation of a Repairable Distributed Recovery Block Scheme," *Fault-Tolerant Computing Symposium 18*, June 1988,pp.50-55.

[Lam62] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems,* Vol. 4 No. 3, July 1982, pp. 382-401.

[Mar85] Alejandro Miguel San Martin. *Robust Failure Detection Filters,* Master of Science Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1985 (under NASA Grant NAG1-126).

[Mes81] Jere Schenck Meserole, Jr. *Detection Filters for Fault-Tolerant Control of Turbofan Engines,* Doctor of Philosophy Thesis, Massachusetts Institute of Technology, June 1985.

[Moo83] F. K. Moore. "A Theory of Rotating Stall of Multistage Axial Compressors" (A three-part article), *Transactions of the ASME*, articles 83-GT-44 through 83-GT-46, January 1983.

[Nap89] Marcello Napolitano and Robert L. Swaim. "A New Technique for Aircraft Flight Control Reconfiguration," *AIAA Guidance, Navigation and Control Conference*, Boston, August 1989.

[Pad90] James Paduano, Lena Valavani, A. H. Epstein, E. M. Greitzer and G. Guenette. *Active Control of Rotating Stall*, December 1990.

[Pad91] James Paduano. *Active Control of Rotating Stall in Axial Compressors, an MIT Gas Turbine Laboratory Presentation*, February 1991.

[Sak91] C. E. Sakamaki. *The Design and Construction of a Data Path Chip Set for a Fault Tolerant Parallel Processor.* Master of Science Thesis, Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology, February 1991.

[Sch84] Richard S. Schabowsky, Jr., and William Stanton. *Interface I Phase I Architecture Selection and Reliability Studies*, Draper Report CSDL-C-5724, June 1984.

[Schal91] Christiaan Mauritz Van Schalkwyk. *Control System Failure Monitoring Using Generalized Parity Relations,* Master of Science Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1991.

[Schm87] Hermann Schmid, Stanley Larimer, and Tahm Sadeghi. "CATS: Computer-Aided Trade Study Methodology," *NAECON '89,* pp. 560-568.

[Shin84] Kang G. Shin and Yann-Hang Lee. "Error Detection Process -- Model, Design, and Its Impact on Computer Performance." *IEEE Trans. Computers,* C-33(6):529-540, June 1984.

an84] Vandervelde, Adams, Gai, Lala, Walker and Desai. *Fault Tolerant Computer Systems (16.322) Course Notes*, Massachusetts Institue of Technology, Fall 1984.