Advanced Detection, Isolation, and Accommodation of Sensor Failures – Real-Time Evaluation

Walter C. Merrill, John C. DeLaat, and William M. Bruton
NASA Lewis Research Center, Cleveland, Ohio

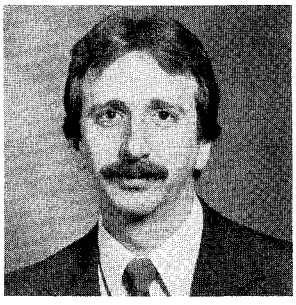
The objective of the Advance Detection, Isolation, and Accommodation program is to improve the overall demonstrated reliability of digital electronic control systems for turbine engines by detecting sensor failures using analytical redundancy. A computer algorithm has been developed to accomplish this task. In this paper, results of a real-time hybrid computer evaluation of the algorithm are presented. Minimum detectable levels of sensor failures for an F100 engine control system are determined. Also included are details about the microprocessor implementation of the algorithm, as well as a description of the algorithm itself.

Introduction

THE objective of the Advanced Detection, Isolation, and Accommodation (ADIA) program is to improve the overall demonstrated reliability of digital electronic control systems for turbine engines by detecting, isolating, and accommodating sensor failures using analytical redundancy. This paper discusses the results of the test bed evaluation of an analytical redundancy-based algorithm developed as part of the ADIA program.

Over the past 35 years, hydromechanical implementations of turbine engine control systems have matured into highly reliable units. However, there is a trend toward increased engine complexity. This increased complexity is required to

Walter C. Merrill was born in Oberlin, Ohio, in November 1948. His education includes a B.S. from the General Motors Institute in 1972, and an M.S. from Auburn University in 1971, both in electrical engineering. In 1975, he received a Ph.D. in engineering science from the University of Toledo. Since 1975, he has been a researcher at the NASA Lewis Research Center. He was responsible for the Advanced Detection, Isolation, and Accommodation program and currently is a senior research engineer in the Advanced Controls Technology Branch. At present, he is applying advanced concepts to reusable rocket engine control systems. Other interests include advanced control concepts for turbine engines, integrated control, and system identification. Dr. Merrill has also analyzed and designed photovoltaic power system controls. He is a Senior Member of the IEEE.



John C. DeLaat is an electronics engineer in the Advanced Controls Technology Branch at the NASA Lewis Research Center in Cleveland, Ohio. He received B.S. and M.S. degrees in electrical engineering from the University of Washington in 1980, and from the University of Toledo in 1986, respectively. With NASA since 1977, his work has included the design of several microcomputer systems for the implementation of advanced control algorithms. His current work involves the implementation of distributed, multicomputer systems for integrated aircraft/propulsion control. His interests include multiprocessing, computer communication networks, analytical redundancy, and control system architectures.

William M. Bruton was born in Detroit, Michigan, in October 1933. He received a B.S. in mathematics from Michigan State University in 1955. Since 1963, he has been employed by NASA at the Lewis Research Center where he has served as manager of the Lewis Analog/Hybrid Simulation Facility. His primary area of interest has been the development of real-time computer simulations in the fields of jet engines, wind turbines, and wind tunnels.

Received April 17, 1987; revision received Nov. 13, 1987.

meet ever-increasing engine performance requirements. Consequently, engine control has become increasingly complex. Because of this complexity trend and the revolution in digital electronics, the control has evolved from a hydromechanical to a full-authority digital electronic (FADEC) implementation. These FADEC-type controls have to demonstrate the same or improved levels of reliability as their hydromechanical predecessors.

Thus, in an effort to improve the overall reliability of the digital electronic control system, various redundancy management techniques have been applied to both the total control system and individual components. One of the least reliable of the control system components is the engine sensor. In fact, some type of sensor redundancy will be required to achieve adequate control system reliability. One important type is analytical redundancy. Analytically redundant systems can have cost and weight savings over other redundancy approaches such as hardware redundancy. Additionally, future engine systems with demanding mission reliability and flight safety requirements will require redundant information not only in the sensor subsystem, but in computers and actuator interfaces as well. Analytically redundant systems such as ADIA utilize the full onboard computational capability to extract information redundancy from dissimilar sensors. This allows maximum system reliability with minimum hardware replication and computer interfaces. This will, in turn, lead to weight and cost savings.

Considerable work has been accomplished in the application of analytical redundancy to improve turbine engine control system reliability. Reference 1 surveys these accomplishments and defines several technology needs. These needs include 1) the ability to detect small magnitude (soft) failures, 2) real-time implementations of algorithms capable of detecting soft failures, 3) a comparison of algorithm complexity vs performance, 4) a full-scale demonstration of a soft failure detection capability, and 5) an evaluation of the pseudolinearized modeling approach. The ADIA program addresses all of these technology needs.

The ADIA program is organized into four phases: development, implementation, evaluation, and demonstration. References 2–5 describe the development and implementation phases. In the development phase[2,3] the ADIA algorithm was designed using advanced filtering and detection methodologies. In the implementation phase,[4,5] this advanced algorithm was implemented in microprocessor-based hardware. A parallel computer architecture (three processors) was used to allow the algorithm to execute in a time frame consistent with stable, real-time operation. This report describes the evaluation phase. In this phase, an evaluation of algorithm performance was obtained using a real-time hybrid computer engine simulation. The objectives of the evaluation are to validate the algorithm for sensor failure detection, isolation, and accommodation effectiveness; document algorithm performance; validate the algorithm's real-time implementation; and establish a data base for the demonstration phase of the ADIA program. The ADIA algorithm demonstration will be conducted on a full-scale engine in the NASA Lewis Research Center's altitude test facility.

This paper begins with a description of the test bed system used in the evaluation of the ADIA algorithm. Next, a description of the ADIA algorithm is given, followed by a description of the implementation hardware. The results of the evaluation are then presented. Finally, conclusions and recommendations for further work are given.

Test Bed System Description

The algorithm was evaluated using a test bed system. The test bed system consists of the engine system simulation, the multivariable control, and the ADIA algorithm. The algorithm will be described in the next section. The test bed system is shown in Fig. 1.

Engine System

The engine system simulation consists of an F100 turbofan engine, the control actuators, and sensors. The F100 engine is

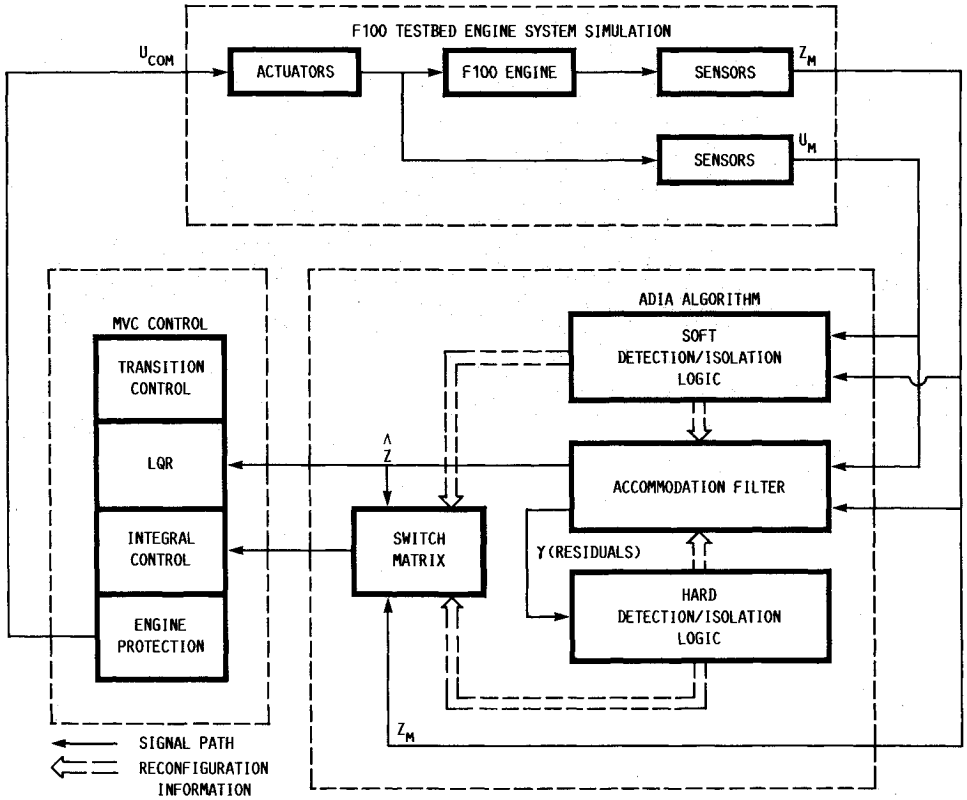


Fig. 1 F100 test bed system

a high-performance, low-bypass ratio, twin-spool turbofan engine. The engine has four controlled inputs, five sensed outputs, and four sensed environmental variables. These variables are defined as follows.

Controlled engine inputs U_{com} and U_m:

WF	Main burner fuel flow
AJ	Exhaust nozzle area
CIVV	Fan inlet variable vanes
RCVV	Rear compressor variable vanes

Sensed engine outputs Z_m:

N1	Fan speed
N2	Compressor speed
PT4	Burner pressure
PT6	Exhaust nozzle pressure
FTIT	Fan turbine inlet temperature

Sensed environmental variables E_m:

P0	Ambient (static) pressure
PT2	Fan inlet (total) pressure
TT2	Fan inlet temperature
TT25	Compressor inlet temperature

Strictly speaking, TT25 is an engine output variable. However, since TT25 sensor failures are not covered by the ADIA logic and TT25 is used only as a scheduling variable in the control (like TT2), it is called an environmental variable.

Multivariable Control System

The multivariable control (MVC) system shown in Fig. 1 is essentially a model following proportional plus integral control. The components of the control are the reference-point schedules, transition schedules, proportional control logic, integral control logic, and engine protection logic. The reference-point schedules generate a desired engine operating point from the pilot's input request. The transition logic generates rate-limited command trajectories for smooth transition between steady-state operating points. The proportional and integral control logic minimize transient and steady-state deviations from the commanded trajectories. The engine protection logic limits the size of the commanded engine inputs. This control is more completely described in Ref. 6. The control modes in this logic normally use fuel to set engine fan speed and nozzle area to set nozzle pressure (engine pressure ratio). However, at those conditions for which limiting is required, fuel flow can be used to limit the maximum FTIT, the maximum PT4, or the minimum PT4.

Evaluation Hardware

The ADIA algorithm was evaluated using a real-time hybrid computer simulation of an F100 engine.[7] A microprocessor-based control computer that implemented the MVC control and ADIA algorithm and included accompanying interface and monitoring hardware[8] and interactive data acquisition software was also used. A personal-computer-based system simulated the required sensor failures.[9]

Algorithm Description

The ADIA algorithm detects, isolates, and accommodates sensor failures in turbofan engine control systems. The algorithm incorporates advanced filtering and detection logic and is general enough to be applied to different engines or other types of control systems.

The ADIA algorithm consists of three elements: 1) hard failure detection and isolation logic, 2) soft failure detection and isolation logic, and 3) an accommodation filter. The algorithm detects two classes of sensor failures, hard and soft. Hard failures are out of range or large bias errors that occur instantaneously in the sensed values. Soft failures are small bias errors or drift errors that accumulate relatively slowly with time.

In the normal or unfailed mode of operation, the accommodation filter uses the full set of engine measurements to generate a set of optimal estimates of the measurements. These estimates ($\hat{Z}$) are used by the control law. When a sensor failure occurs, the detection logic determines that a failure has occurred. The isolation logic then determines which sensor is faulty. This structural information is passed to the estimator. The estimator now removes the faulty measurement from further consideration. The estimator, however, continues to generate the full set of optimal estimates for the control. Thus, the control mode does not have to restructure for any sensor failure.

As shown in Fig. 1, the ADIA algorithm inputs are the sensed engine output variables Z_m, the environmental variables E_m, and the sensed engine input variables U_m. The outputs of the algorithm, the estimates $\hat{Z}(t)$ of the measured engine outputs $Z_m(t)$, are used as input to the proportional part of the control. During normal mode operation, engine measurements are used in the integral control. When a sensor failure is accommodated, the measurement in the integral control is replaced with the corresponding accommodation filter estimate by reconfiguring the interface switch matrix.

Engine Model

The performance of the accommodation filter and the detection and isolation logic are strongly dependent on a model of the engine. The model used has a linear state-space structure. However, nonlinear engine characteristics are incorporated by representing the matrix elements within the linear state-space structure and the base points as nonlinear functions of various engine variables.

$$\dot{X} = F(X - X_b) + G(U - U_b) \tag{1a}$$

$$Z = H(X - X_b) + D(U - U_b) + Z_b \tag{1b}$$

Here, the subscript b represents the base point (steady-state point) and X is the 4×1 model state vector, U the 4×1 control vector, and Z the 5×1 output vector. The F, G, H, and D matrices are the appropriately dimensioned system matrices. The system matrices and the model base points were determined at 109 operating points throughout the flight envelope. Three variables are sufficient to define a model operating point completely. Previous modeling efforts used altitude, Mach number, and power lever angle (PLA). However, in this study, PLA, inlet pressure (PT2), and inlet temperature (TT2) were used. This second operating point definition is the more appropriate format for ensuring that all significant model dynamics are considered by adequately spanning the entire envelope with model points. Once system matrices are determined at all of the 109 operating points, the individual matrix elements are corrected by the engine inlet conditions E_m and scheduled as nonlinear functions of Z. These functions are given in Ref. 2.

Accommodation Filter

The accommodation filter incorporates the engine model along with a Kalman gain update to generate estimates of the engine outputs $\hat{Z}$ and states $\hat{X}$ as follows:

$$\dot{\hat{X}} = F(\hat{X} - X_b) + G(U_m - U_b) + K\gamma \tag{2a}$$

$$\hat{Z} = H(\hat{X} - X_b) + D(U_m - U_b) + Z_b \tag{2b}$$

$$\gamma = Z_m - \hat{Z} \tag{2c}$$

where K is the Kalman gain matrix and γ the residual vector. As with the system matrices, the elements of K are corrected by E_m and scheduled as nonlinear functions of $\hat{Z}$.

An improvement that was added to the accommodation filter was the incorporation of integral action to improve steady-state accuracy of the FTIT estimate ($\hat{Z}_5$). One impor-

tant engine control mode is the limiting of FTIT at high-power operation. Because the FTIT sensor is relatively slow, control action is based on the dynamically faster FTIT estimate. Because the FTIT limiting control has integral action, a high degree of steady-state accuracy in the FTIT estimate is required to ensure satisfactory control. This accuracy is accomplished by augmenting the filter with the following additional state and output equations:

$$\dot{b} = K_6\gamma \tag{3a}$$

$$\widehat{\text{FTIT}} = \hat{Z}_5 + b \tag{3b}$$

where K_6 is a gain matrix, b the temperature bias, and $\hat{Z}_5$ the unbiased temperature estimate. The addition of these dynamics, although improving steady-state FTIT estimation accuracy, results in a larger minimum detectable FTIT drift failure rate. This filter structure, which includes the FTIT bias state, is the structure used in the accommodation filter and all the hypothesis filters used in the soft detection and isolation logic.

Reconfiguration of the accommodation filter after the detection and isolation of a sensor failure is accomplished by forcing the appropriate residual element to zero. For example, if a compressor speed sensor failure (N2) has been isolated, the effect of reconfiguration is to force $\gamma_2 = 0$.

This is equivalent to setting sensed N2 equal to the estimate of N2 generated by the filter. The residuals generated by the accommodation filter are used in the hard failure detection logic.

Hard Failure Detection and Isolation Logic

The hard failure detection and isolation logic is straightforward. To accomplish hard failure detection and isolation, the absolute value of each component of the residual vector is compared to its own threshold. If the residual absolute value is greater than the threshold, a failure is detected and isolated for the sensor corresponding to the residual element. Thresholds are initially sized from the standard deviation of the noise on the sensors. These standard deviation magnitudes are then increased to account for modeling errors in the accommodation filter. The hard detection threshold values are twice the magnitude of these adjusted standard deviations. These magnitudes are summarized in Table 1. Failure accommodation is accomplished by reconfiguring the accommodation filter and all of the hypothesis filters in the soft failure detection logic.

Soft Failure Detection and Isolation Logic

The soft failure detection and isolation logic consists of multiple hypothesis-based testing. Each hypothesis is implemented using a Kalman filter. The soft detection and isolation logic structure is shown in Fig. 2. A total of six hypothesis filters are shown, one for normal mode operation and five for the failure modes (one for each engine output sensor). The structure for each hypothesis filter is identical to the accommodation filter. However, each hypothesis filter uses a different set of measurements. For example, the first hypothesis filter (H1) uses all of the sensed engine outputs except the first, N1. The second uses all of the sensed outputs except the second, N2, and so on. Thus, each hypothesis filter generates a unique residual vector γ_i. From this residual, each hypothesis filter generates a statistic or likelihood based on a weighted sum of squared residuals (WSSR). If we assume Gaussian sensor noise, each sample of γ_i has a certain likelihood or probability.

$$L_i = p_i(\gamma_i) = ke^{-\text{WSSR}_i} \tag{4}$$

where k is a constant and

$$\text{WSSR}_i = \gamma_i^T \Sigma^{-1}\gamma_i \tag{5}$$

with

$$\Sigma = \text{diag}(\sigma_j^2) \tag{6}$$

Table 1 Hard detection threshold magnitudes

Sensor	j	Adjusted standard deviation, σ_j	Detection threshold, λ_H
N1	1	300 rpm	600 rpm
N2	2	400 rpm	800 rpm
PT4	3	30 psi	60 psi
PT6	4	5 psi	10 psi
FTIT	5	250 R	500 R

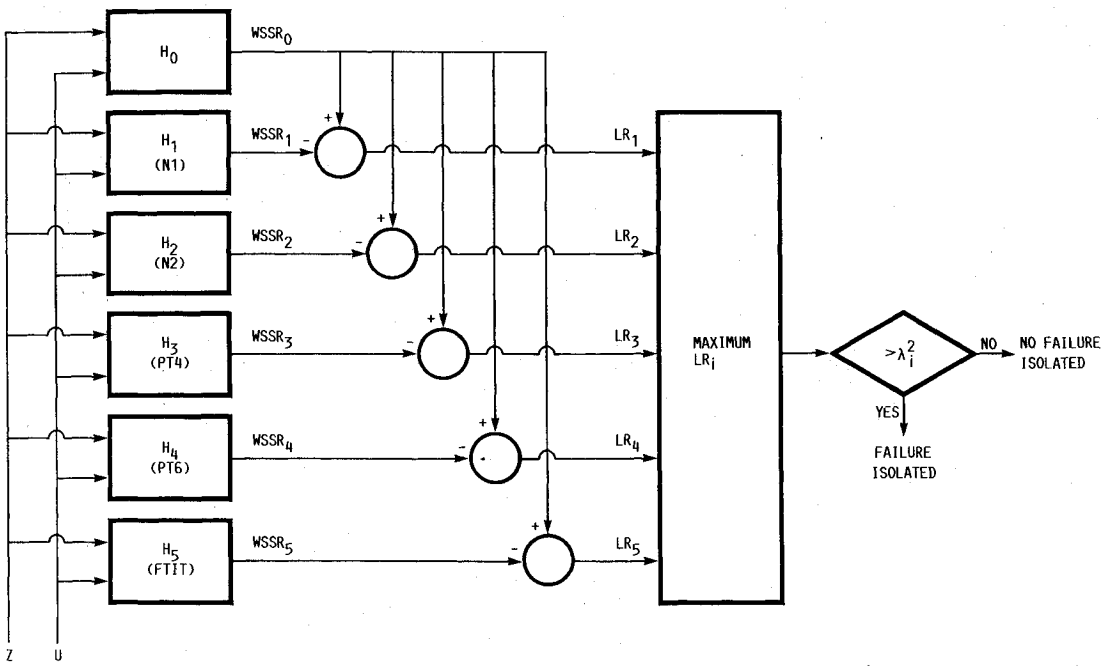


Fig. 2 Soft failure detection and isolation structure.

The σ_j are the adjusted standard deviations defined in Table 1. These standard deviation values scale the residuals to unitless quantities that can be summed to form WSSR. The WSSR statistic is smoothed to remove gross noise effects by a first-order lag with a time constant of 0.1. Mathematically, when the log of the ratio of likelihood is taken, then

$$LR_i = \log(L_i/L_0) = \text{WSSR}_o - \text{WSSR}_i \quad (7)$$

If the maximum log likelihood ratio exceeds the threshold, then a failure is detected and isolated and accommodation occurs. If a sensor failure has occurred in N1, for example, all of the hypothesis filters except H1 will be corrupted by the faulty information. Thus, each of the corresponding likelihoods will be small, except for H1. Therefore the H1 likelihood ratio will be the maximum and it will be compared to the threshold to detect the failure.

For accommodation, two steps are taken. First, all seven of the filters (accommodation and six hypothesis) are reconfigured to account for the detected failure mode. Second, the states and estimates of all seven filters are updated to the correct values of the hypothesis filter that corresponds to the failed sensor.

Adaptive Threshold

Since the WSSR statistic is the sum of Gaussian variables squared, the WSSR statistic has a χ^2 distribution. Initially, the soft failure detection/isolation threshold is determined by standard statistical analysis of this distribution to set the confidence level of false alarms and missed detections. Next, the threshold is modified to account for modeling error. It was soon apparent from initial evaluation studies that transient modeling error was dominant in determining the fixed threshold level. It was also clear that this threshold was too large for desirable steady-state operation.

Thus, an adaptive threshold was incorporated to make the algorithm more robust to modeling error. The adaptive threshold as defined in Eqs. (8)

$$\lambda_i = \lambda_{iSS}(\lambda_{EXP} + 1) \quad (8a)$$

$$\tau\dot{\lambda}_{EXP} + \lambda_{EXP} = M_{tran} \quad (8b)$$

was heuristically determined and consists of two parts. One part, λ_{iSS}, the steady-state detection/isolation threshold, accounts for the steady-state or low-frequency modeling error. The second part, λ_{EXP}, accounts for the transient or high-frequency modeling error. The adaptive threshold is triggered by an internal control system variable, M_{tran}, that is indicative of transient operation. The values of λ_{iSS}, τ, and M_{tran} were determined by experimentation to minimize false alarms due to modeling during transients. When the engine experiences a transient, M_{tran} is set to 4.5; otherwise, it is 0. The time constant $\tau = 2$ s. The adaptive threshold expansion logic enabled λ_{iSS} to be reduced to 40% of its original value, which results in an 80% reduction in the detection/isolation threshold λ_i^2.

ADIA Algorithm Implementation

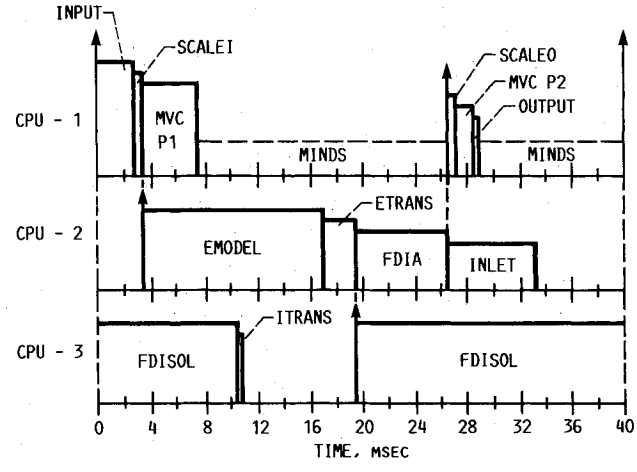
The implementation of the ADIA algorithm required the integration of the ADIA algorithm with an existing microcomputer implementation of the F100 multivariable control (MVC) to give a full microcomputer implementation of the control algorithm with sensor analytical redundancy. The resulting controls microcomputer system is based on the Intel 80186 microprocessor architecture. To satisfy the control update interval requirement of 40 ms necessary for stable engine operation three processors (CPU's) operating in parallel are used. Data are transferred between CPU's through dual-ported memory. Synchronization between CPU's was achieved through interrupts. Relative CPU timing and algorithm partitioning are shown in Fig. 3.

The existing MVC implementation had been programmed in fixed-point assembly language and was used without change on CPU 1. Most of the ADIA algorithm running on CPU 2 and 3 was programmed using floating-point arithmetic and the application-oriented language FORTRAN. FORchosen because the ADIA as developed was coded in FORTRAN. The use of floating-point arithmetic was possible because of the availability of efficient hardware-based floating-point coprocessing.

To achieve real-time operation, the FORTRAN source code has been optimized to make it run more efficiently.[10] In addition, floating-point table lookup routines,[11] which are executed frequently in the algorithm, and the hardware interface routines, which have no FORTRAN equivalent, are implemented in assembly language. As shown in Fig. 3, the entire MVC-ADIA algorithm executes in less than the required 40 ms.

A software-based data-acquisition package [the microcontroller interactive data system (MINDS)[12]] executes in the spare time on CPU no 1 (Fig. 3). The package has both steady-state and transient data capabilities and can access any variable in the MVC or ADIA algorithm for display or plotting.

The memory requirements for each of the three CPU's are shown in Fig. 4. Each of the three CPU's has, in addition to its share of the MVC-ADIA algorithm, an executive routine that maintains correct real-time operation of the total algorithm. In addition, the memory requirement of MINDS is



CPU-1

INPUT	A/D CONVERSION AND SCALING OF ENGINE MEASUREMENTS
SCALEI	CONVERSION OF MEASUREMENTS TO FLOATING POINT AND TRANSFER TO CPU-2
MVCP1	MVC PART 1: REFERENCE POINT SCHEDULE, TRANSITION CONTROL, AND GAIN SCHEDULE CALCULATIONS
MINDS	INTERACTIVE DATA SYSTEM (SPARE TIME CALCULATION)
SCALED	TRANSFER OF ESTIMATES AND RECONFIGURATION INFORMATION FROM CPU-2 AND CONVERSION TO FIXED POINT
MVCP2	MVC PART 2: PROPORTIONAL CONTROL, INTEGRAL CONTROL, AND ENGINE PROTECTION LOGIC
OUTPUT	UNSCALING AND D/A CONVERSION OF ENGINE INPUTS

CPU-2

EMODEL	ENGINE MODEL MATRIX AND BASE POINT CALCULATION
ETRANS	TRANSFER OF EMODEL INFORMATION TO CPU-3
FDIA	ACCOMMODATION FILTER AND HARD DETECTION LOGIC CALCULATIONS
INLET	MACH NUMBER AND ALTITUDE CALCULATION

CPU-3

FDISOL	HYPOTHESIS FILTERS AND SOFT ISOLATION LOGIC CALCULATIONS
ITRANS	TRANSFER OF SOFT ISOLATION INFORMATION TO CPU-3

Fig. 3 ADIA algorithm partitioning and CPU timing.

shown for CPU 1. In all cases, the code and constants were about 65% and the data or variables about 35% of the total memory required. Last, Fig. 4 shows the total memory requirements for all executives, the total algorithm, and MINDS for all three CPU's combined.

Real-Time Evaluation

This section describes the evaluation of the ADIA algorithm using a hybrid computer-based, real-time simulation of an F100 engine. The objectives, procedure, and results of the evaluation are discussed.

Objectives

The first objective of the evaluation was to validate the operation and performance of the ADIA algorithm and its implementation. It was especially important to conduct this validation in a real-time environment to establish the feasibility and practicality of the implementation. The second objective of the evaluation was to document the performance of the algorithm over the envelope of the engine. Finally, the third objective was to establish a data base for comparison with results obtained during the demonstration phase of the program.

Procedure

The procedure for evaluating the algorithm is defined by a test matrix. The test matrix, shown in Fig. 5, consists of the different operating points used for the evaluation across the top of the matrix and the different tests conducted at these points along the side. An engine-operating point is defined by the pilot's power request (power lever angle, PLA), and the altitude (ALT) and Mach number (MN) at which the engine is operating.

The rationale used to select the test matrix operating points was to duplicate as many of the points used in the F100 Multivariable Control Program[6] as possible, to avoid high fan inlet pressures, and to reasonably span the envelope. This rationale is a compromise between taking advantage of previous results for comparison, limited risk engine operation, and full envelope validation.

The tests used in the evaluation were selected to completely define detection performance for five common failure modes. Also, tests were conducted to determine engine control performance with and without the ADIA algorithm and with and without engine sensor failures. The tests are summarized in Table 2.

Results

Three types of real-time evaluation results are presented. The first shows the performance of the microprocessor-based MVC control. The second shows the accuracy of the Kalman-filter-based estimator. Finally, the performance of the ADIA algorithm itself is given.

MVC Performance

The performance of the MVC control was evaluated without the ADIA logic. This evaluation demonstrated that the microprocessor-based implementation of the MVC control algorithm could accurately and safely control the test bed engine. Steady-state and transient results were obtained at the test matrix operating points. In every case, both the steady-state and transient performance were good.

Estimator Accuracy

The single most important element in determining ADIA algorithm performance is the accuracy of the engine output estimates used in the algorithm. These estimates are determined by the accommodation filter that incorporates a simplified engine model. The accuracy of the output estimates for both steady-state and transient operation was evaluated at various engine-operating points. The accuracy of the estimates is presented in two parts, steady-state and transient.

Steady-state accuracy was obtained in a straightforward

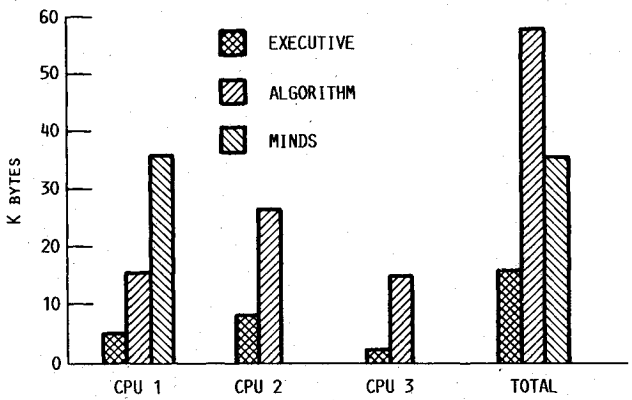


Fig. 4 MVC-ADIA memory requirements.

Table 2 Test definitions

Test name	Description
	Sensor failures
Hard	Large-magnitude bias failure
Soft	Small-magnitude bias failure
Drift	Small-magnitude drift failure
Noise	Random noise failure
Scale	Scale factor bias change
Sequence	Sequence of successive sensor failures
	PLA transients
Pulse	Minimum to military to minimum transient power excursions. The military power level is maintained for 10 s.
Open	Same as the pulse test, except that the military power level is decreased slightly and the engine is controlled without using any sensed engine output information in the control.
Single	Same as the pulse test with a single sensor failure accommodated before initiating the transient.

	ALTITUDE/MACH NUMBER										
	10K/0.6		30K/0.9		10K/0.9		45K/0.9	10K/1.2	50K/1.8	35K/1.9	55K/2.2
PLA	50	80	50	80	50	80	70	80	80	80	80
HARD	X	X									
SOFT	X	X	X	X	X	X	X	X	X	X	X
DRIFT	X	X	X	X	X	X	X	X	X	X	X
SSF	X	X	X	X							
PULSE	X		X		X		X	X			
SINGLE	X		X		X		X	X			
OPEN	X		X		X		X	X			

Fig. 5 Evaluation test matrix.

manner. The simulation was "flown" to the desired operating point and allowed to reach steady state. MINDS was then used to sample and store a set of steady-state data. Comparisons of measured and estimated variables for seven operating points are given in Table 3 by showing the difference (the residual) between sensed and estimated engine outputs as a percent of nominal vlaue. From these comparisons, it is clear that the estimates exhibit excellent steady-state accuracy. Maximum error values occur for PT4 at the 55K point and PT6 at the 35K point. These error magnitudes can easily be reduced by straightforward adjustment of the base-point schedules used in the algorithm.

Transient accuracy data were obtained in the following manner. Again, the simulation was flown to the desired operating point and allowed to reach steady state. An idle to intermediate (military) power PLA pulse transient was then simulated at five different operating conditions. MINDS was used to sample and store data throughout the transient. An example plot of sensed and estimated fan speed and its residual, as well as the likelihood ratio for N1, are presented in Fig. 6.

These trajectories give the reader a "feel" for the summarized results of Tables 4–7. In Table 4, the maximum value of the residuals obtained in response to the reference transient is given for each output at each of the five operating points. In Table 5, the average absolute values of the residuals are given.

Since detection performance is determined by the likelihood ratios, estimate accuracy interpreted in terms of these statistics is critical to understanding algorithm performance. The maximum ratio values and the average ratio values are given in Tables 6 and 7, respectively, to summarize transient accuracy for the reference trajectories.

Plots of the likelihood ratios became the standard tool used for evaluation and performance prediction. Overall, transient accuracy was considered to be quite good although not as good as steady-state accuracy. It was fairly evident, then, that detection performance could be greatly improved if different thresholds for steady-state and transient detection were allowed. This observation led immediately to the implementation of the adaptive threshold logic just described.

Estimator accuracy results were also obtained when a single sensor failure had been detected and accommodated before the pulse transient. These results are given in Table 8 for maximum residual values and in Table 9 for average residual values. These tests show very little degradation in estimator accuracy performance when operating with only four sensors. Thus, ADIA performance will not degrade significantly after a single sensor failure.

Detection/Accommodation Performance

Two types of sensor failures were considered, hard and soft. Hard failures, because of their size, are easily detected. Thus, hard failure detection performance, although important to system reliability, was examined at only two operating points. The ADIA algorithm exhibited excellent hard detection

Table 3 Steady-state estimation accuracy results (percent of nominal), no sensor failures

	Engine outputs				
Alt./Mach/PLA	N1	N2	PT4	PT6	FTIT
10K/0.6/83	0.43	0.11	3.16	0.53	0.11
10K/0.9/50	0.06	0.16	0.21	1.53	0.11
30K/0.9/83	0.42	0.28	1.36	0.69	0.04
45K/0.9/60	0.12	0.21	1.87	1.45	0.04
10K/1.2/83	0.17	0.11	1.36	0.33	0.11
55K/2.2/83	0.33	0.54	5.64	2.48	0.05
35K/1.9/83	0.03	0.32	0.92	5.12	0.01
50K/1.8/83	0.39	0.49	3.12	2.48	0.04
Average	0.24	0.28	2.21	1.83	0.06
Maximum	0.43	0.54	5.64	5.12	0.11

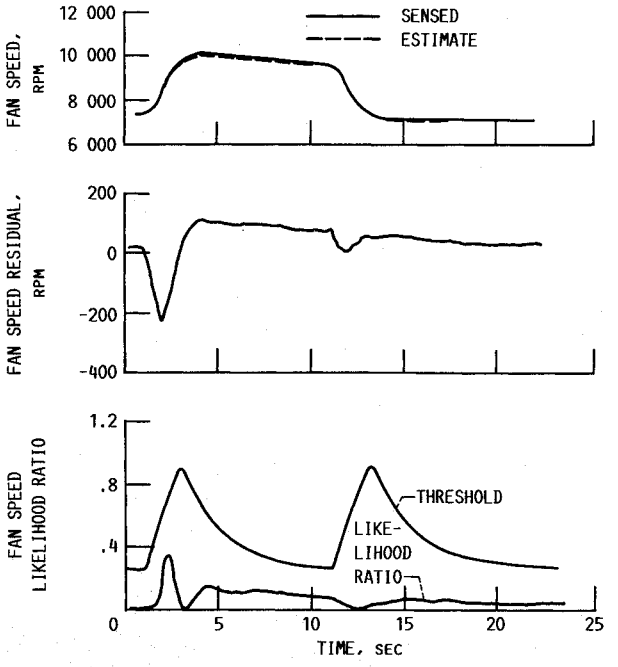


Fig. 6 Fan speed pulse transient response at the 10,000 ft, 0.6 Mach number condition.

Table 4 Maximum residual value (percent of nominal) in response to the PLA pulse input (normal mode)

	Engine outputs				
Alt./Mach	N1	N2	PT4	PT6	FTIT
10K/0.6	3.57	0.81	6.50	12.55	5.78
30K/0.9	1.47	0.74	4.48	13.08	5.49
10K/0.9	4.30	1.13	5.22	14.98	5.68
45K/0.9	2.89	1.86	7.81	19.30	4.44
10K/1.2	1.54	1.24	5.02	9.21	3.50
Average	2.74	1.16	5.81	13.82	4.98
Maximum	4.30	1.86	7.81	19.30	5.78

Table 5 Average residual absolute value (percent of nominal) in response to a PLA pulse input (normal mode)

	Engine outputs				
Alt./Mach	N1	N2	PT4	PT6	FTIT
10K/0.6	0.77	0.25	1.67	2.33	1.44
30K/0.9	0.63	0.28	0.92	2.94	1.64
10K/0.9	0.60	0.42	0.78	2.98	1.39
45K/0.9	0.49	0.36	2.37	5.78	1.21
10K/1.2	0.21	0.23	1.21	0.75	0.85
Average	0.54	0.31	1.39	2.96	1.31
Maximum	0.77	0.42	2.37	5.78	1.64

Table 6 Maximum likelihood ratio value in response to a PLA pulse input (normal mode)

	Engine outputs				
Alt./Mach	N1	N2	PT4	PT6	FTIT
10K/0.6	0.876	0.342	0.286	0.598	0.080
10K/0.9	0.241	0.146	0.076	0.305	0.042
10K/0.9	1.025	0.236	0.162	1.340	0.059
45K/0.9	0.400	0.238	0.009	0.059	0.042
10K/1.2	0.126	0.253	0.355	0.401	0.041
Average	0.533	0.243	0.177	0.540	0.053
Maximum	1.025	0.342	0.355	1.340	0.080

Table 7 Average likelihood ratio value in response to a PLA pulse input (normal mode)

	Engine outputs				
Alt./Mach	N1	N2	PT4	PT6	FTIT
10K/0.6	0.086	0.057	0.044	0.059	0.010
30K/0.9	0.060	0.037	0.006	0.045	0.007
10K/0.9	0.088	0.047	0.038	0.123	0.010
45K/0.9	0.044	0.034	0.003	0.007	0.007
10K/1.2	0.015	0.107	0.024	0.020	0.006
Average	0.059	0.056	0.023	0.051	0.008
Maximum	0.088	0.107	0.044	0.123	0.010

Table 8 Maximum residual errors for pulse transients at 10K/0.6 with a single sensor failure

	Failure mode					
Output	None	N1	N2	PT4	PT6	FTIT
N1	3.570	0.000	2.931	2.592	2.481	2.877
N2	0.810	0.979	0	0.823	0.784	0.791
PT4	6.500	7.513	5.211	0	6.454	6.550
PT6	12.550	10.740	12.220	11.480	0	11.880
FTIT	5.780	6.383	5.704	5.650	5.731	0

Table 9 Average residual errors for pulse transients at 10K/0.6 with a single sensor failure

	Failure mode					
Output	None	N1	N2	PT4	PT6	FTIT
N1	0.770	0	0.755	0.472	0.571	0.483
N2	0.240	0.249	0	0.250	2.65	0.270
PT4	1.670	1.479	0.693	0	1.067	1.260
PT6	2.330	2.211	2.486	2.432	0	2.431
FTIT	1.440	1.305	1.417	1.406	1.455	0

performance at these points. There were no false alarms or missed detections of any hard failures. Hard detection and accommodation of the failure were accomplished in each case. In addition, no false alarms in the hard detection logic were encountered during the subsequent soft failure evaluation.

Soft sensor failures, although small in magnitude, if undetected may result in degraded or unsafe engine operation. Soft failures are more difficult to detect. Therefore, the evaluation concentrated on soft failure detection and isolation performance. Four soft failure modes were considered: bias, drift, noise, and scale factor. Algorithm performance for the bias and drift failure modes was studied extensively. Performance for the noise and scale factor modes was studied at a limited number of conditions. Performance criteria studied were minimum detectable bias values and drift rates, detection time, steady-state performance degradation, and transient response to failure accommodation.

The procedure followed to obtain performance data was identical to that used to obtain transient accuracy data. Additionally, sensor failures of the appropriate size were simulated at the desired time. The results obtained are summarized for the minimum detectable level of bias in Table 10.

In Table 10, the minimum detectable biases at 11 different operating points for each engine output are given. Detection times for these biases were essentially instantaneous. The results are presented as a percent of full scale and of nominal. Full-scale values are constant and nominal values can vary throughout the operating range. Notice that the size of failures detected as a percent of full scale is essentially constant over the operating range. Notice also that the maximum minimum (greater lower bound) detectable bias (29.96%) as a percent of nominal occurs for PT6 at 45K/0.9/70 due to the low nominal value of PT6 at this condition. However, the failure magnitude as a percent of scale (7.74%) is about the average. Overall, the minimum detectable bias magnitudes are small and represent excellent performance.

Table 10 Minimum bias failure magnitudes (percent)

Operating point	NL, rpm	NH, rpm	PT4, psi	PT4, psi	FTIT, °F
			Bias/nominal		
10K/0.6/50	3.47	2.60	6.39	12.02	12.00
10K/0.6/83	3.41	2.67	3.85	7.74	8.75
30K/0.9/50	3.43	3.10	10.92	18.10	12.36
30K/0.9/83	3.25	2.73	6.99	13.24	9.41
10K/0.9/50	3.54	2.60	6.17	9.74	12.17
10K/0.9/83	2.91	2.66	2.86	6.40	8.72
45K/0.9/70	2.12	3.34	21.33	29.96	18.18
10K/1.2/70	2.11	3.21	5.71	7.78	10.00
50K/1.8/83	2.94	2.70	8.33	17.44	9.06
35K/1.9/83	2.95	2.31	5.94	7.69	8.94
55K/2.2/83	2.57	3.76	16.34	15.54	8.80
			Bias/full-scale		
10K/0.6/50	2.92	2.29	3.85	7.74	8.75
10K/0.6/83	3.41	2.67	3.85	7.74	8.75
30K/0.9/50	2.92	2.67	4.15	7.10	8.75
30K/0.9/83	3.17	2.67	4.15	7.74	8.75
10K/0.9/50	2.92	2.29	4.15	7.10	8.75
10K/0.9/83	2.92	2.67	3.38	7.74	8.75
45K/0.9/70	1.95	3.05	5.54	7.74	14.59
10K/1.2/70	1.95	3.05	6.15	9.04	8.75
50K/1.8/83	2.92	2.67	3.85	7.74	8.75
35K/1.9/83	2.92	2.29	5.85	7.74	8.75
55K/2.2/83	2.44	3.81	7.69	7.74	8.75

Table 11 Minimum drift failure magnitudes (percent)

Operating point	NL, rpm/s	NH, rpm/s	PT4, psi/s	PT6, psi/s	FTIT, °F/s
			Drift/nominal		
10K/0.6/50	1.16	0.87	1.28	3.20	5.60
10K/0.6/83	1.22	0.95	0.38	1.55	4.08
30K/0.9/50	1.49	0.89	2.02	3.95	5.77
30K/0.9/83	1.50	0.97	1.55	2.65	4.39
10K/0.9/50	1.18	0.87	1.14	2.13	5.68
10K/0.9/83	1.21	0.95	0.52	1.28	4.07
45K/0.9/70	1.06	1.25	3.55	7.99	5.45
10K/1.2/70	0.26	1.20	1.00	2.67	4.67
50K/1.8/83	0	0.01	0.01	0.03	0.05
35K/1.9/83	1.23	0.38	0.94	0.51	4.17
55K/2.2/83	0.77	1.61	2.94	3.11	4.40
			Drift/full-scale		
10K/0.6/50	0.97	0.76	0.77	2.07	4.08
10K/0.6/83	1.22	0.95	0.38	1.55	4.08
30K/0.9/50	1.27	0.76	0.77	1.55	4.08
30K/0.9/83	1.46	0.95	0.92	1.55	4.08
10K/0.9/50	0.97	0.76	0.77	1.55	4.08
10K/0.9/83	1.22	0.95	0.62	1.55	4.08
45K/0.9/70	0.97	1.14	0.92	2.07	4.38
10K/1.2/70	0.24	1.14	1.08	3.10	4.08
50K/1.8/83	0	0.01	0.01	0.03	0.04
35K/1.9/83	1.22	0.38	0.92	0.52	4.08
55K/2.2/83	0.73	1.64	1.38	1.55	4.38

In Table 11, the minimum detectable drift rates are given for the 11 operating points of the test matrix. These rates were determined by adjusting the drift magnitude such that a failure was detected approximately 5 s after failure inception. Again, failure magnitude results are presented as a percent of full scale and of nominal. As in the bias case, the maximum minimum detectable drift rate as a percent of nominal occurs at the 45K/0.9/70 operating condition. However, as before, this value is well below the maximum value as a percent of full scale. The percentages of full-scale values are all similar in magnitude. In general, these results are excellent.

To place these results in perspective, a comparison of the

Table 12 Steady-state results of slow drift failure transients for the original ADIA algorithm

Flight operating point, deg	Failure parameter	ADIA algorithm: Parameter bias before DIA	Change in thrust before DIA, %	Time for DIA, s	Comments	Performance[a]
0/0/24	PT6	7.4 psi (42%)	−4.5	0.490		A
0/0/40	N1	1333 rpm (12.1%)	−44.5	1.994		U
0/0/83	PT4	46.5 psi (12.7%)	−0.1	3.080	Filter noisy during DIA	A
0/1.2/83	FTIT	90°F (5.2%)	−2.2	2.53		A
10K/0.75/50	PT4	40.5 psi (19.6%)	−0.2	2.664		A
10K/0.75/83	PT6	9 psi (21.8%)	−1.5	0.572		A
20K/0.3/40	N2	Undetected	–	Undetected	Unstable diverging	U
20K/0.3/83	N2(−)	Undetected	–	Undetected	Unstable	U
20K/0.3/83	N2	1415 rpm (11.4%)	−4.5	3.518		A
25K/1.0/83	PT4	46.5 psi (18.1%)	−0.2	3.066		A
25K/2.2/83	PT4	False alarm	–	–	PT4 and PT6 false alarms prior to failure	U
40K/0.6/40	N2	Miss	−48	–	2000 rpm drift miss	U
40K/0.6/83	PT6	6.75 psi (63.7%)	−0.5	0.448	System oscillatory after failure induced	A
45K/2.2/83	PT4	−56.4 psi (−24.5%)	−0.16	3.750		A
60K/1.2/83	N2	2000 rpm (15.8%)	−19.4	5.016	Drift caused system to go unstable	U
65K/2.5/83	PT6	−3.75 psi (−27.4%)	+24.7	0.400	FTIT false alarm	U

[a]A = acceptable; U = unacceptable.

Table 13 Drift failure results

Flight operating points	Failure	Parameter bias	Thrust change, %	DIA time
0/0/24	PT6	3.7 psi	0	0.5
0/0/40	N1	263.0 rpm	1	0.3
0.0/83	PT4	15.2 psi	0	0.9
0/1.2/83	FTIT	124.0 F	−4	2.5
10K/0.75/50	PT4	16.4 psi	0	0.9
10K/0.74/83	PT6	2.7 psi	−8	3.7
20K/0.3/40	N2	248.0 rpm	−1	0.3
20K/0.3/83	N2(−)	−234.0 rpm	1	0.5
20K/0.3/83	N2	265.0 rpm	−1	0.3
25K/1.0/83	PT4	16.8 psi	0	1.0
25K/2.0/83	PT4	17.8 psi	−1	1.2
40K/0.6/40	N2	216.0 rpm	−5	0.5
40K/0.6/83	PT6	2.6 psi	−11	1.6
45K/2.1/83	PT4	15.7 psi	0	0.5
60K/1.2/83	N2	274.0 rpm	−15	0.8
65K/2.3/83	PT6	3.7 psi	−5	2.3
45K/2.1/83	PT4(−)	−15.0 psi	0	1.5

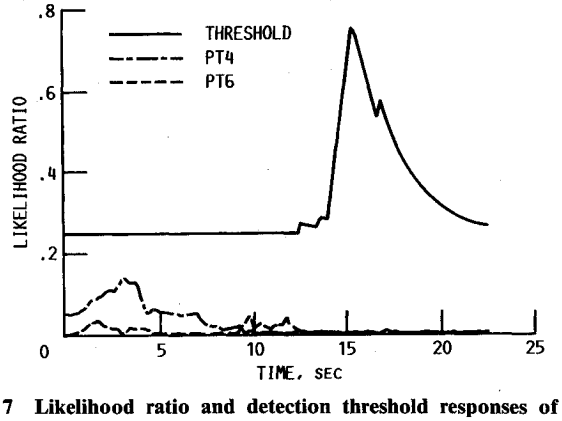


Fig. 7 Likelihood ratio and detection threshold responses of the burner pressure (PT4) and exhaust nozzle pressure (PT6) outputs during a 10K/0.6–45K/0.9 excursion.

soft detection performance of this improved version of the algorithm is made with the original algorithm.[1] In this comparison, drift failures were injected at 17 "edge-of-the-envelope" conditions. Performance data for the original algorithm, presented in Table 12, were obtained from a nonlinear, digital simulation of the engine. The performance of the improved algorithm is presented for comparison in Table 13. The improved algorithm incorporates a more accurate model for detecting smaller failures, an adaptive detection threshold for more robust performance, and a different functional architecture of the isolation portion of the algorithm that allows faster detection times. These improvements result in the demonstrated improved performance evident in the comparison of Tables 12 and 13.

Some of the operating points of Table 12 are only approximated in Table 13. This is because the hybrid computer could not successfully attain these conditions due to scaling limitations. In every case but one, the parameter bias before detection of the improved version of the algorithm is better than in the original version. In all failure cases, the improved algorithm allows continued engine operation, whereas in some cases, the original algorithm would have required an engine shutdown. The drift failure rates used in this comparison are given in Ref. 1.

Estimation accuracy and false alarm performance were also evaluated for an altitude and Mach number excursion from 10K/0.6–45K/0.9. Likelihood ratio and detection threshold responses are given in Fig. 7 for the PT4 and PT6 engine outputs. The likelihood ratios for N1, N2, and FTIT were all less in magnitude than for PT4 and PT6. Here, the accuracy is quite good and there are no false alarms.

Conclusions

As a result of the real-time evaluation study, several conclusions have been reached. First, it can be concluded that the failure detection algorithm works and works quite well. Sensor failure detection and accommodation were demonstrated over a broad range of operating points and power conditions. The minimum detectable failure magnitudes represent excellent algorithm performance. Second, the algorithm can be implemented in a realistic computer environment with an update interval consistent with real-time operation. Off-the-shelf microprocessor-based hardware and straightforward programming procedures, including FORTRAN and floating-point arithmetic, were used. Parallel processing was also used and shown to be an effective multiplier of computer computational resources. Finally, it is concluded that the failure detection algorithm and multivariable control microprocessor-based implementations are ready for the demonstration phase of the program. The demonstration of the failure detection algorithm will take place on a full-scale F100 engine in the NASA Lewis' altitude test facility.

References

[1]Merrill, W.C., "Sensor Failure Detection for Jet Engines Using Analytical Redundancy," *Journal of Guidance, Control, and Dynamics*, Vol. 8, Nov.–Dec. 1985, pp. 673–682.

[2]Beattie, E.C., Laprad, R.F., McGlone, M.E., Rock, S.M., and Akhter, M.M., "Sensor Failure Detection System for the F100 Turbofan Engine," Pratt & Whitney Aircraft Group, East Hartford, CT, PWA-5736-17, 1981; also, NASA CR-165515.

[3]Beattie, E.C., Laprad, R.F., Akhter, M.M., and Rock, S.M., "Sensor Failure Detection for Jet Engines," Pratt & Whitney Aircraft Group, East Hartford, CT, PWA-5891-18, 1983; also, NASA CR-168190.

[4]DeLaat, J.C. and Merrill, W.C., "A Real-Time Implementation of an Advanced Sensor Failure Detection, Isolation, and Accommodation Algorithm," AIAA Paper 84-0564, Jan. 1984.

[5]Merrill, W.C. and DeLaat, J.C., "A Real-Time Simulation Evaluation of an Advanced Detection, Isolation and Accommodation Algorithm for Sensor Failures in Turbine Engines," *1986 American Control Conference*, IEEE, New York, 1986, pp. 162–169.

[6]Lehtinen, B., Costakis, W.G., Soeder, J.F., and Seldner, K., "F100 Multivariable Control Synthesis Program-Results of Engine Altitude Tests," NASA TM S-83367, 1983.

[7]Szuch, J.R., Seldner, K., and Cwyner, D.S., "Development and Verification of a Real-Time, Hybrid Computer Simulation of the F100-PW-100(3) Turbofan Engine," NASA TP-1034, 1977.

[8]DeLaat, J.C. and Soeder, J.F., "Design of a Microprocessor-Based Control, Interface and Monitoring (CIM) Unit for Turbine Engine Controls Research," NASA TM-83433, 1983.

[9]Melcher, K.J., DeLaat, J.C., Merrill, W.C., Oberle, L.G., Schaefer, J.H., and Sadler, G.G., "A Personal Computer Based Sensor Failure Simulator for Control System Reliability Studies," NASA TM-87271, 1986.

[10]DeLaat, J.C., "A Real-Time FORTRAN Implementation of a Sensor Failure Detection, Isolation and Accommodation Algorithm," *1984 American Control Conference*, IEEE, New York, 1984, pp. 572–573.

[11]Mackin, M.A. and Soeder, J.F., "Floating-Point Function Generation Routines for 16-Bit Microcomputers," NASA TM-83783, 1984.

[12]Soeder, J.F., "MINDS: A Microcomputer Interactive Data System for 8086-Based Controllers," NASA TP-2378, 1985.