Open access • Proceedings Article • DOI:10.2514/6.1984-569

A real-time implementation of an advanced sensor failure detection, isolation, and accommodation algorithm — Source link

J. C. Delaat, W. C. Merrill

Institutions: Glenn Research Center

Published on: 01 Jan 1984

Topics: Redundancy (engineering), Computer programming, Software and Microprocessor

NASA TM-83553

NASA Technical Memorandum 83553

NASA-TM-83553 19840005072

AIAA PAPER 84-0569

A Real-Time Implementation of an Advanced Sensor Failure Detection, Isolation, and Accommodation Algorithm

John C. DeLaat and Walter C. Merrill
Lewis Research Center
Cleveland, Ohio

Prepared for the
Twenty-second Aerospace Sciences Meeting
sponsored by the American Institute of Aeronautics and Astronautics
Reno, Nevada, January 9-12, 1984

NASA

E-1928

A REAL-TIME IMPLEMENTATION OF AN ADVANCED SENSOR FAILURE DETECTION, ISOLATION, AND ACCOMODATION ALGORITHM

John C. DeLaat and Walter C. Merrill

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

Abstract

In recent years, advanced control algorithms for turbofan engines have been implemented using digital electronic control mechanisms. However, digital electronic controls need some component redundancy in order to attain sufficient reliability. A sensor failure detection, isolation, and accommodation algorithm has been developed for NASA Lewis Research Center which incorporates analytic sensor redundancy through software. This algorithm has been implemented in a high level language on a microprocessor based controls computer. Parallel processing and state-of-the-art 16-bit microprocessors are used along with efficient programming practices to achieve real-time operation.

Introduction

In recent years, advanced aircraft turbine engine controls have been implemented using digital electronic control mechanisms rather than the hydromechanical mechanisms of the past.[1,2,3,4] This allows the application of modern (optimal) control theory to the design of control algorithms which can take into account the loop interactions inherent in today's sophistocated turbofan engines. However, studies have shown that current digital electronic controls for turbine engines cannot attain sufficient reliability without some level of control component redundancy.[5,6] There are three major areas in which redundancy can be incorporated to improve the reliability of a digital electronic control system. They are the sensors, the actuators, and the controls computer itself. This effort addresses redundancy in the sensor set.

Two methods are available for incorporating sensor redundancy. The first method, hardware redundancy, involves adding multiple identical sensors to the control system. A technique such as voting can then be used to detect and isolate sensor failures so that a faulty sensor can be eliminated from the system. Redundant multiple sensors, however, do have some drawbacks when incorporated into a control system. Adding redundant sensors to the controls hardware will increase the weight, cost, and complexity of the control system. The second method of incorporating sensor redundancy is software or analytic redundancy. This method requires the controls computer, through software, to determine when a sensor failure has occurred without redundant hardware sensors in the control system. The controls computer can then provide an estimate of the correct value of the failed sensor's output to the control algorithm.

An advanced sensor failure Detection, Isolation, and Accommodation (ADIA) algorithm has been developed for NASA Lewis Research Center under contract.[7,8] This algorithm has been evaluated in nonreal-time using a digital simulation running on a mainframe computer. It is desirable to evaluate the algorithm in a real-time environment to show that it is a practical alternative to hardware sensor redundancy. For this effort, the ADIA algorithm has been implemented on a real-time, microprocessor-based controls computer.[9] This implementation was achieved using parallel processing and a high level programming language. This paper describes the ADIA implementation. The algorithm is described and the hardware and software considerations necessary to achieve the real-time implementation are discussed along with some of the practical experience gained during the process.

ADIA Algorithm Description

The ADIA algorithm detects, isolates, and accommodates sensor failures in turbofan engine control systems. The ADIA algorithm was developed for NASA Lewis under contracts NAS3-22481 and NAS3-23282 by Pratt and Whitney Aircraft with their subcontractor Systems Control Technology. The algorithm incorporates advanced filtering and detection logic and is general enough to be applied to different engine or other types of control systems. A specific version of the ADIA algorithm was designed for use with the F100 engine and the F100 Multivariable Control Synthesis (MVCS) program control.[1,2,3] This combination of engine, control, and ADIA algorithm comprises the F100 Multivariable Control testbed system and is shown in Fig. 1.

Algorithm Operation

The ADIA algorithm consists of four elements: (1) hard failure detection and isolation logic, (2) soft failure detection logic, (3) soft failure isolation logic, and (4) an accommodation filter. These are shown as part of the testbed system in Fig. 1. 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 erors that accumulate relatively slowly with time.

The algorithm inputs are the measured engine inputs, $u_m(t)$, and the measured engine outputs, $z_m(t)$. These variables are defined in Table I. The algorithm outputs are optimal estimates, $\hat{z}(t)$, of the engine outputs, $z(t)$.

The algorithm has two modes of operation, normal and failure. During normal mode operation, i.e., when no sensor failure is present, the normal mode accommodation filter uses all the measured information to determine $\hat{z}(t)$. In failure mode operation, one of the five sensors has failed. Simultaneous multiple sensor failures are rare events and are not considered. A threefold process takes place once the failure has occurred. First the failure is detected. Once a failure is known to have occurred, the specific faulty sensor must be isolated. Finally, when isolation has occurred, the failure is accommodated by reconfiguring the

N84-13140 #

normal mode accommodation filter which generates the estimates, $\hat{z}(t)$. This threefold procedure takes place for both hard and soft failures.

The normal mode accommodation filter logic, shown in Fig. 2, generates the estimates of the engine outputs, $\hat{z}(t)$. In the Kalman filter equations, the matrices F, G, H, and D are typical state space system matrices where $x(t)$ is the 4x1 vector of estimates of the engine's state variables and and $\gamma(t)$ is the 5x1 vector of residuals. The matrix K is the Kalman gain matrix. All the system matrices as well as the Kalman gain matrix are scheduled as a function of operating point to model variations in engine dynamics. Almost all of the matrices' elements are nonzero, thus, almost all the matrix elements must be multiplied through the filter equations.

The hard failure detection and isolation logic, shown in Fig. 3, performs a straightforward threshold check on each sensor residual, γ_i. Threshold values are determined from sensor and process noise values as well as sensor range considerations. If a residual value is greater that the threshold, λ_H, hard failure detection and isolation follow immediately.

If a hard failure has not occurred, then a soft failure check is performed. The soft failure detection logic, shown in Fig. 4, first calculates an average weighted sum of squared residuals (WSSR). A soft failure is detected when the weighted sum is greater than a prespecified soft failure detection threshold, λ_S. The number, N, of past residuals summed to obtain the average, the weighting factor, W, and the detection threshold, λ_S, are design parameters that are chosen to provide an acceptable tradeoff between false alarms and missed detections.

Once a soft failure has been detected, the soft failure isolation logic, shown in Fig. 5, is used to isolate the failed sensor. Six different isolation filters generate six different sets of residuals, one for each possible failed sensor, and one based on no failed sensors. A log likelihood ratio, H_i is generated for each set of residuals. A test is then performed which determines the most probable set of residuals by finding the maximum H_i. When this maximum H_i is above an isolation threshold, λ_I, the faulty sensor is isolated.

Once a hard or soft failure is detected and isolated, the accommodation filter is reconfigured by an appropriate change of its Kalman gain matrix K to remove the failed sensor from consideration as shown in Fig. 6. For a soft failure of sensor i, the accomodation filter is also reinitialized to the current value of $z_i(t)$ and $x_i(t)$ from the appropriate isolation filter. Reinitialization is necessary since a significant amount of time will have elapsed between failure and isolation.

Algorithm Interface

The ADIA algorithm interface to the F100 MVCS control is shown in Fig. 1 and in detail in Fig. 7. The MVCS control includes four major logic sections: (1) transition control, (2) integral control, (3) proportional control (Linear Quadratic Regulator or LQR), and (4) engine protection logic. In the normal, no failure mode the output of the accommodation filter, $\hat{z}(t)$ is fed to the LQR portion of the control while sensor outputs are fed directly to the integral portion of the control. When a failure of sensor i is accommodated, the output of the reconfigured accommodation filter, $\hat{z}(t)$, is fed to the LQR. Now, however, the ith element of $\hat{z}(t)$ replaces the ith sensor output, which is faulty, in the integral control.

Using $\hat{z}(t)$ in the LQR in both modes minimizes the transient impact of sensor failures on the engine. However, the accommodation filter will inevitably bias the estimates of the engine outputs, $\hat{z}(t)$, due to modelling error. So, in the integral control during no failure mode operation, the measured values of engine outputs, $z_m(t)$ are used so that the engine outputs rather that the biased estimates of the engine outputs will be controlled to the desired setpoint. In the failure mode, only the measurement corresponding to the failed sensor is replaced by an estimate to minimize the effect of the biased terms. Transient effects upon engine operation due to this signal switching are effectively removed by the filtering action of the integrators. A complete, detailed description of the ADIA algorithm can be found in Refs. 7 and 8.

Implementation Description

A real-time Intel 8086 microprocessor based control system has been designed and fabricated at NASA Lewis for controls research.[9] The controls microcomputer in this system uses an iSBC 86/30 singe board computer as its central processing unit (CPU) and runs under a commercially available disk-based operating system. In addition to the CPU board, the controls microcomputer contains two floppy disk drives and their associated controller, analog input/output boards, and digital input/output boards. A block diagram of the system is shown in solid lines in Fig. 8. This system, was used in a previous effort to evaluate an implementation of the F100 MVCS control algorithm in real time.[10] The controls computer with a 5 MHz version of the 8086 was able to execute a fixed point assembly language implementation of the MVCS algorithm in 19 msec. This implementation occupied 13 kilobytes of microprocessor memory.

During the nonreal-time evalutation of the ADIA algorithm, it was estimated that the ADIA would take at least as long to execute as the MVCS algorithm. Thus it was estimated that the combined MVCS-ADIA would take over 40 msec to execute. Previous evalutation of the MVCS algorithm suggested that this execution time was too long to achieve an update interval commensurate with the system dynamics. So it was decided to use parallel processing to add computational power to the system and thus speed up the execution time of the combined MVCS-ADIA to acceptable levels. To achieve parallel processing, a second 86/30 CPU board, shown in dashed lines in Fig. 8, was added to the controls microcomputer. Since the ADIA and the MVCS algorithms have portions that can run in parallel, the second CPU board is used to run the ADIA algorithm while the first CPU board runs the MVCS algorithm as it did in the previous effort. The bus structure designed into the controls microcomputer for the MVCS evaluation allowed room for expansion so that adding another CPU board to the system was fairly straightforward. The interrupt structure on this bus is used for communication and synchronization between the two CPU boards.

The resulting relative system timing, including the synchronization interrupts and the parallel portions of the MVCS-ADIA algorithm is shown in Fig. 9. The blocks represent the various sections of the two algorithms and the time it takes to execute each section. Upon receiving the update interval interrupt, CPU board 1 gets the algorithm inputs from the sensors through analog-to-digital converters. It then scales these inputs to fixed point integers for the MVCS and to floating point numbers for the ADIA. Board 1 then sends an interrupt to board 2 to tell it the inputs are available and to start processing. Board 1 then calculates the transition control. In parallel, board 2 does the engine model and accommodation filter calculation. When board 2 has finished these calculations, it sends an interrupt to board 1 telling it that the estimates of the sensors are available. Board 2 then performs the hard and soft failure detect logic and starts the soft failure isolation logic if necessary. Any failures isolated will be accommodated and flagged to board 1 during the next update interval. While board 2 performs the detect and isolate logic, board 1 finishes the MVCS calculation by computing the integral control, the LQR, and the engine protect logic. The algorithm outputs are then passed to the engine actuators through digital-to-analog converters. The next update interrupt will then trigger the process again starting with the input and scaling of the sensors.

Until this effort, digital control algorithms at NASA Lewis had been programmed exclusively in assembly language. It was generally believed that assembly code generated by hand would run faster than code generated by a high level language compiler and was therefore necessary to achieve real-time execution of the control algorithms. As mentioned earlier, the MVCS algorithm was implemented using fixed point assembly language. Assembly language also has the advantage that programs written in it can interface directly to input/output hardware such as digital-to-analog converters. However, assembly language has the disadvantage that it is not an application oriented language. It is therefore very time consuming to program a sophisticated application in assembly language.

To overcome this disadvantage, a decision was made up front to implement the ADIA algorithm in a high level language. In addition to being easier to program, a high level language implementation also has the advantage of being simpler to debug, maintain, and make changes to due to its increased readability over assembly language. Two factors allow the use of a high level language for real-time control applications. First, manufacturers of high level language compilers for microprocessors are aware that fast execution is very important to a vast number of end users of their products. Thus current compilers have been designed to generate efficient, optimized code. Second, advances in microprocessor hardware speed allow the code generated by these compilers to be executed much faster than on previously attainable hardware. One example of this advanced hardware is the floating point coprocessor, such as an Intel 8087. Running with an 8086 it can execute floating point instructions in hardware many times faster than the software equivalent of the same instructions would execute on the 8086 alone. Higher maximum clock rate microprocessor chips are another example of advanced microprocessor hardware. These allow each instruction to execute proportionately faster.

The ADIA algorithm was delivered to NASA Lewis in FORTRAN. Since a good FORTRAN compiler is available for the 8086-8087 processors, FORTRAN was chosen as the high level language in which to implement the ADIA algorithm. FORTRAN, however, has the disadvantage that it cannot easily interface to input/output hardware such as that contained in the controls microcomputer. For instance, there is no instruction in FORTRAN that will output directly to a digital-to-analog converter.

Therefore, a software implementation was arrived at which takes advantage of the best features of both assembly language and FORTRAN. Assembly language is used for the executive which calls the algorithm subroutines, for all hardware input/output routines, and for the interrupt handlers which allow communication between the two CPU boards. In addition, since the MVCS algorithm had been implemented and evaluated in assembly language, the assembly language software is used for this effort. The ADIA algorithm is left in FORTRAN and therefore benefits from the advantages of a high level language implementation.

Practical Experience Gained

A number of practical experiences were gained during the process of building up the MVCS-ADIA hardware and software. Many of these experiences were driven by the fact that the ADIA algorithm as received took over 600 msec to execute on the controls microcomputer rather than the desired 20-30 msec. More than an order of magnitude reduction in execution time was necessary before real-time evaluation could start. Most of the reduction was achieved in two ways. First, the FORTRAN code for the ADIA algoritm was scrutinized to identify those parts of the algorithm which could be made to run faster through the use of efficient FORTRAN programming. In line code was generated to take the place of most of the DO loops and subroutine and function calls. The FORTRAN compiler was used to generate assembly language listings to aid in identifying those constructs which generate inefficient code. Eliminating indexed addressing, such as indexed array element references, and combining statements often helps the compiler optimize the code it generates. The second method used to reduce execution times was to substitute assembly language routines for the FORTRAN routines which could not be reduced sufficiently. Only one section of the ADIA algorithm had to be converted to assembly language. This section involves many function look-ups and runs much faster in assembly language.

As of the writing of this paper, the ADIA algorithm is very close to being able to execute in the desired time. Higher speed versions of the 8086-8087 microprocessors are on order which will allow a direct increase in the speed of code execution and will thereby eliminate any further worries about execution time. The version of the algorithm incorporating all the changes made to reduce execution time occupies about 40 kilobytes of microprocessor memory.

Conclusions

A real time implementation of the F100 MVCS-ADIA algorithm has been achieved and is ready for evaluation. In order to take advantage of high level language programming, concentration of effort was necessary in two major areas. First, state-of-the-art hardware including parallel processing, high clock rate 16-bit microprocessors, and floating point coprocessors must be used. Second, a thorough understanding of the FORTRAN program, including what type of code the compiler will generate, is necessary in order to generate execution-time efficient code.

In turn, several benefits are derived from implementing the algorithm in a high level language. First, it is easier to understand what the program is doing. This makes validation of the program easier and allows engineers without a microprocessor/assembly language background to work with the program. Second, high level language programs are portable, that is, they can run on more than just the processor they were written for. Thus the programs can be evaluated and tested on any number of computers. Therefore, it has become a viable alternative and a beneficial one to use high level language programming for complex, real-time control algorithms.

References

1. Szuch, J. R., Soeder, J. F., Seldner, K., and Cwynar, D. S., "F100 Multivariable Control Synthesis Program: Evaluation of a Multivariable Control Using a Real-Time Engine Simulation," NASA TP-1056, 1977.
2. Lehtinen, B., Dehoff, R. L., and Hackney, R. D., "Multivariable Control Altitude Demonstration on the F100 Turbofan Engine," AIAA Paper 79-1204, June, 1979.
3. Soeder, J. F., "F100 Multivariable Control Synthesis PRogram - Computer Implementation of the F100 Multivariable Control Algorithm," NASA TP-2231, 1983.
4. Arpasi, D. J., Zeller, J. R., and Batterton, P. G., "A General Purpose Digital System for On-line Control of Air Breathing Propulsion Systems," NASA TM X-2168, 1971.
5. McGlone, M. E., Davies, W. J., Miller, R. J., Smith, T. B., LaLa, J. H., and Peck, W. C., "Full-Authority Fault-Tolerant Electronic Engine Control Systems for Variable Cycle Engines," Pratt and Whitney Aircraft, AFWAL-TR-81-2121, Dec. 1981.
6. Baker, L., Brainard, W. E., Curry, C. E., Disparte, C. P., Dolny, L. J., Fleming, R. E., and Warner, D. E., "Full-Authority Fault-Tolerant Electronic Engine Control Systems for Variable Cycle Engines," Detroit Diesel Allison, AFWAL-TR-82-2037, Apr. 1982.
7. 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 and Whitney, East Hartford, Conn., PWA-5736-17, Aug. 1981 (NASA CR-165515).
8. Beattie, E. C., Laprad, R. F., Akhter, M. M., and Rock, S. M., "Sensor Failure Detection for Jet Engines," Pratt and Whitney Aircraft, East Hartford, Conn., PWA-5891-18, May 1983 (NASA CR-168190).
9. DeLaat, J. C., Soeder, J. F., "Design of a Microprocessor - Based Control, Interface, and Monitoring (CIM) Unit for Turbine Engine Controls Research, NASA TM-83433, 1983.
10. DeLaat, J. C., Soeder, J. F., "Evaluation of a Microprocessor Implementation of the F100 Multivariable Control. Proposed NASA Technical Memorandum.

TABLE I. - ENGINE INPUTS AND OUTPUTS

Engine inputs:
- Fuel flow
- Nozzle area
- Compressor inlet guide vane angle
- Rear compressor variable vane angle
- Bleed flow

Engine outputs:
- Fan speed
- Compressor speed
- Burner pressure
- Augmentor pressure
- Fan turbine inlet temperature

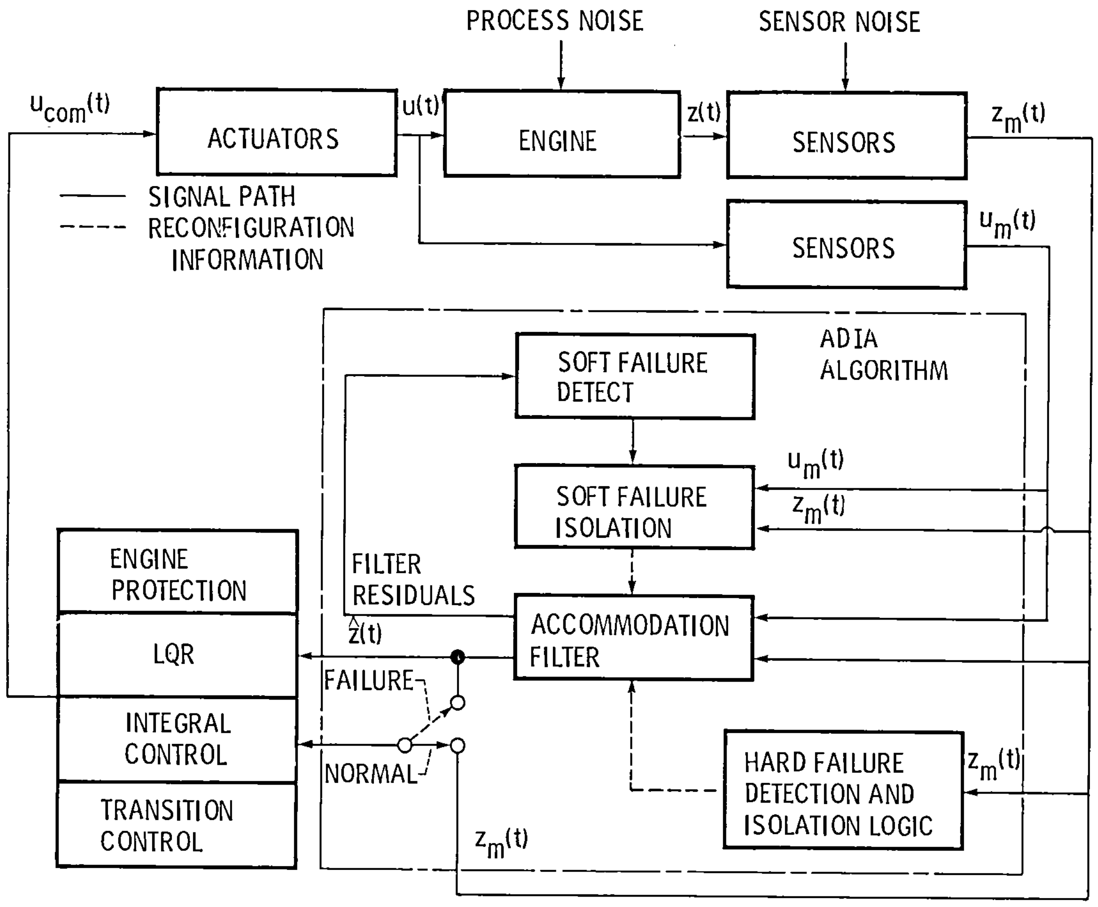


Figure 1. - F100 testbed system with ADIA algorithm.

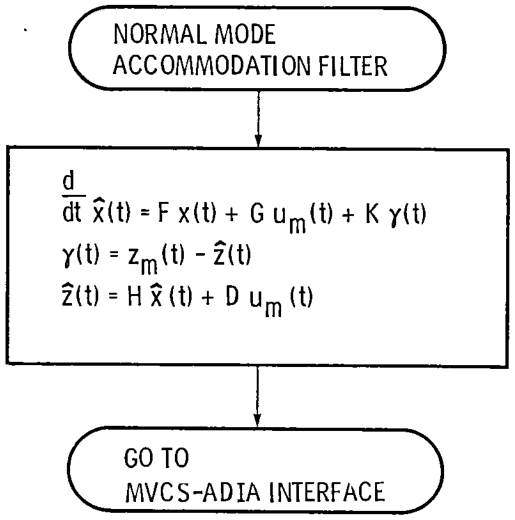

NORMAL MODE ACCOMMODATION FILTER
$\frac{d}{dt}\hat{x}(t) = F\,x(t) + G\,u_m(t) + K\,\gamma(t)$
$\gamma(t) = z_m(t) - \hat{z}(t)$
$\hat{z}(t) = H\,\hat{x}(t) + D\,u_m(t)$
GO TO MVCS-ADIA INTERFACE

Figure 2. - Normal mode accommodation filter logic.

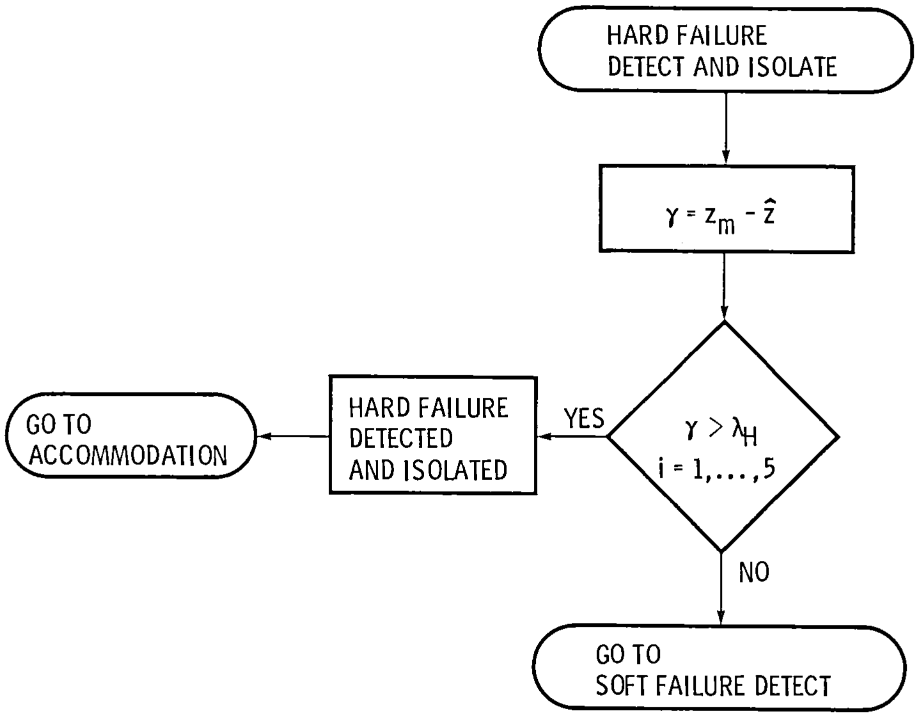

HARD FAILURE DETECT AND ISOLATE
$\gamma = z_m - \hat{z}$
$\gamma > \lambda_H$
$i = 1,\ldots,5$
YES
HARD FAILURE DETECTED AND ISOLATED
GO TO ACCOMMODATION
NO
GO TO SOFT FAILURE DETECT

Figure 3. - Hard failure detection and isolation logic.

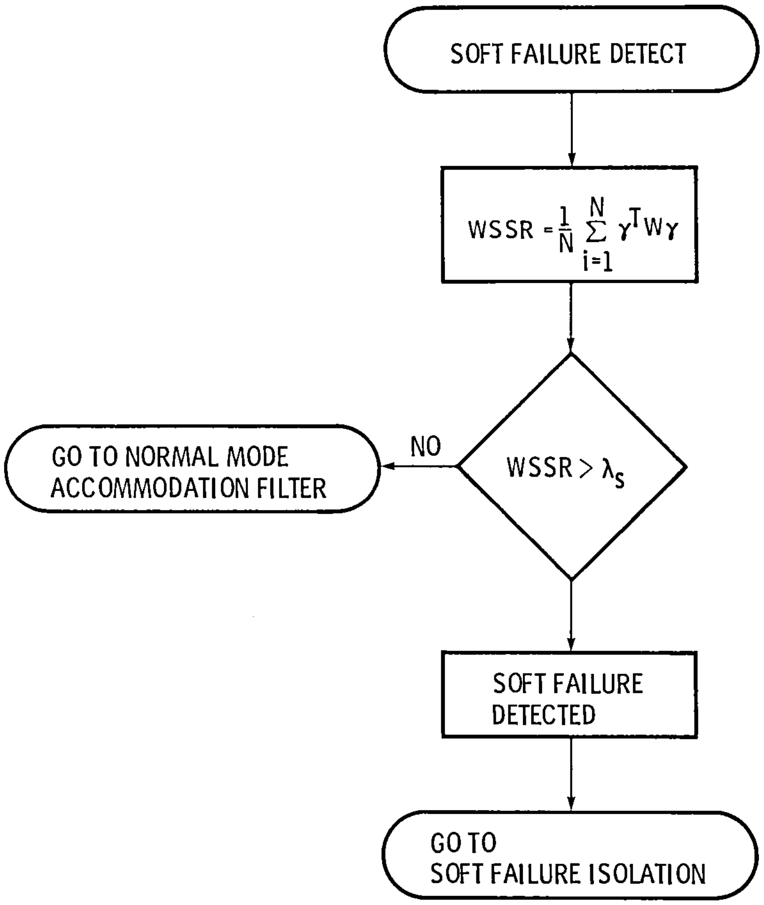

SOFT FAILURE DETECT
$WSSR = \frac{1}{N}\sum_{i=1}^{N} \gamma^T W \gamma$
$WSSR > \lambda_S$
NO
GO TO NORMAL MODE ACCOMMODATION FILTER
SOFT FAILURE DETECTED
GO TO SOFT FAILURE ISOLATION

Figure 4. - Soft failure detection logic.

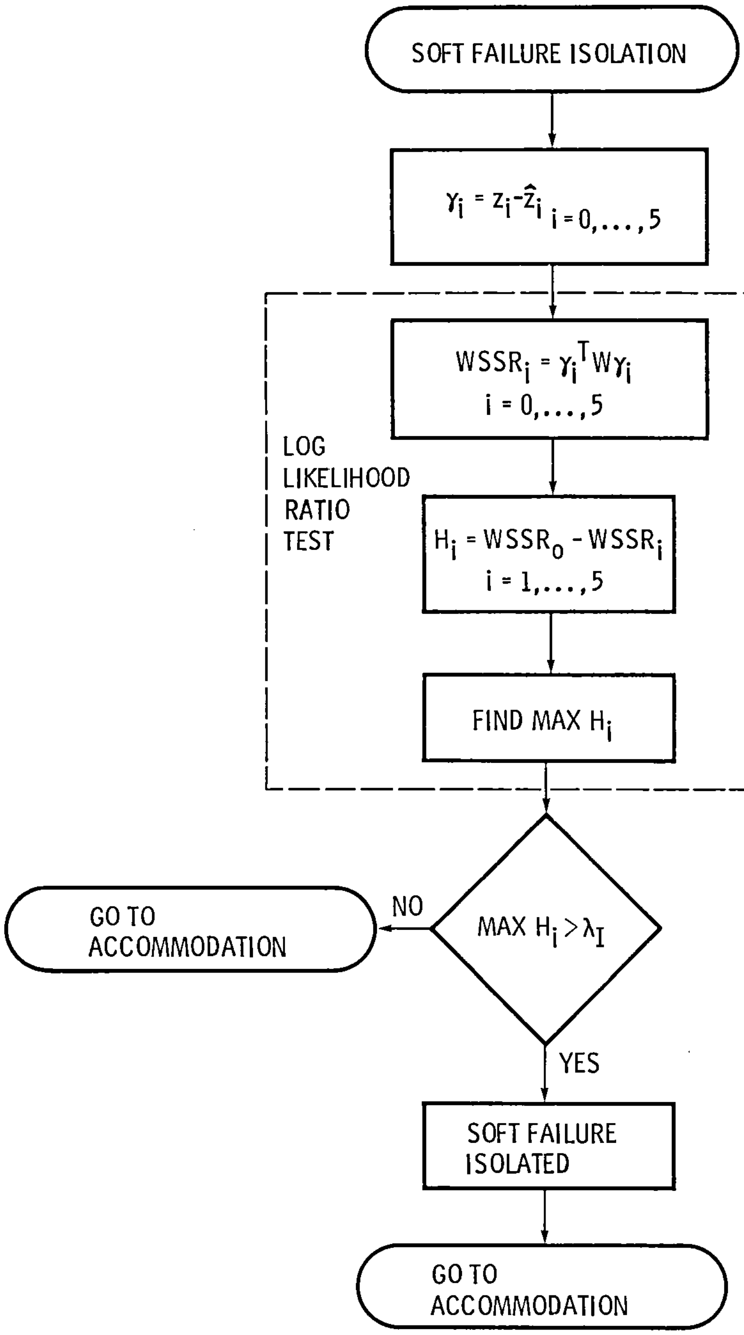
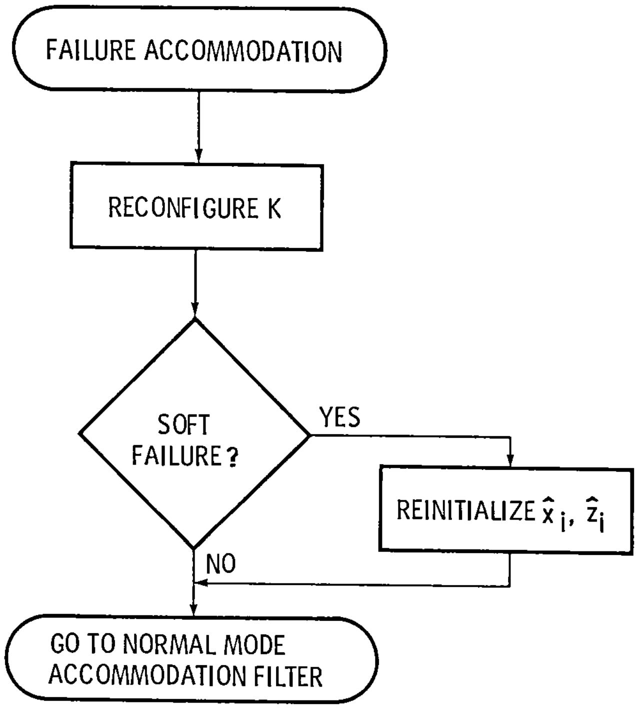


Figure 5. - Soft failure isolation logic.



FAILURE ACCOMMODATION
RECONFIGURE K
SOFT FAILURE?
YES
NO
REINITIALIZE $\hat{x}_i$, $\hat{z}_i$
GO TO NORMAL MODE ACCOMMODATION FILTER

Figure 6. - Failure accommodation logic.

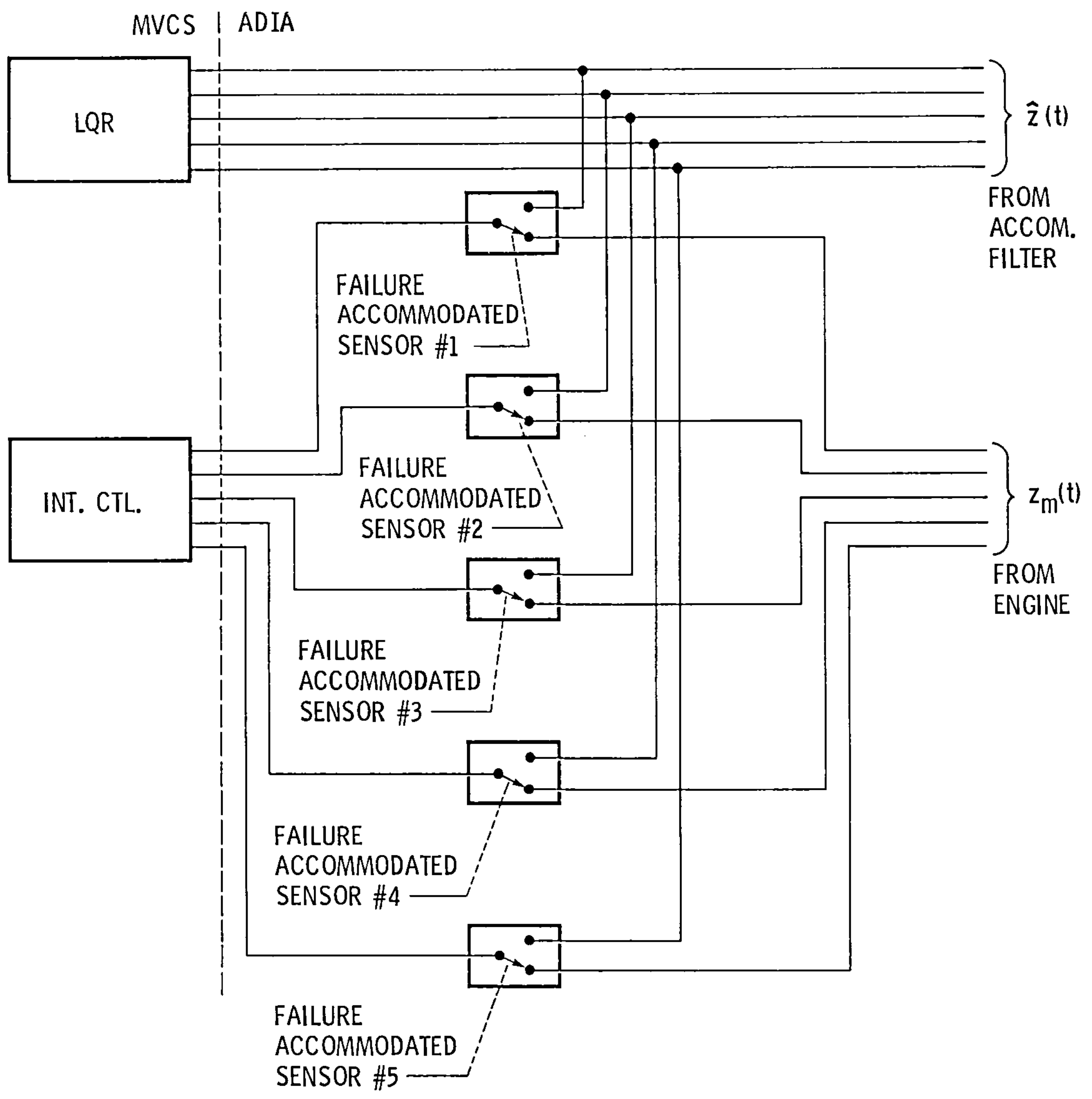

MVCS
ADIA
LQR
INT. CTL.
$\hat{z}(t)$
FROM ACCOM. FILTER
$z_m(t)$
FROM ENGINE
FAILURE ACCOMMODATED SENSOR #1
FAILURE ACCOMMODATED SENSOR #2
FAILURE ACCOMMODATED SENSOR #3
FAILURE ACCOMMODATED SENSOR #4
FAILURE ACCOMMODATED SENSOR #5

Figure 7. - MVCS-ADIA interface.

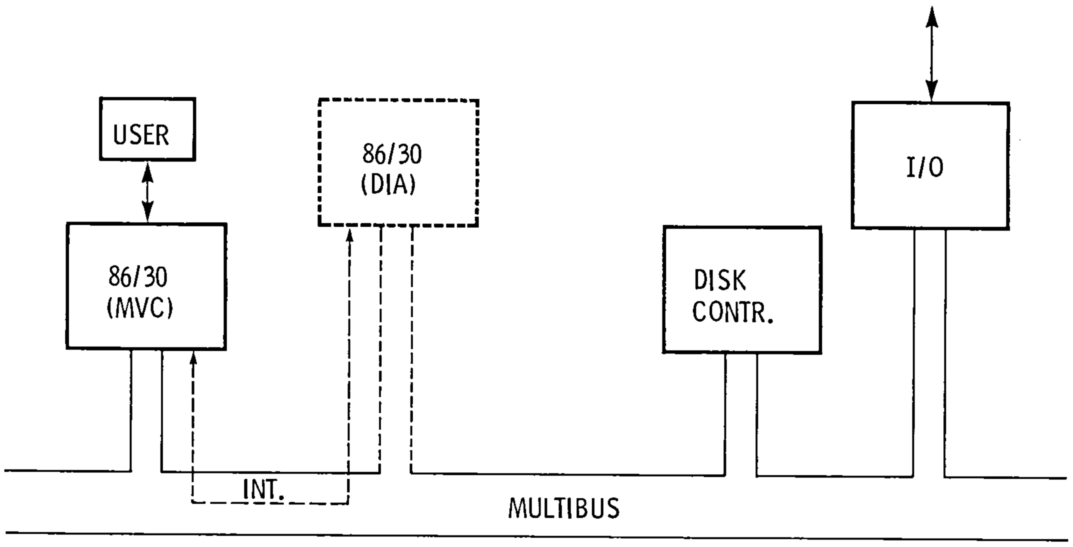


Figure 8. - MVCS-ADIA hardware block diagram.

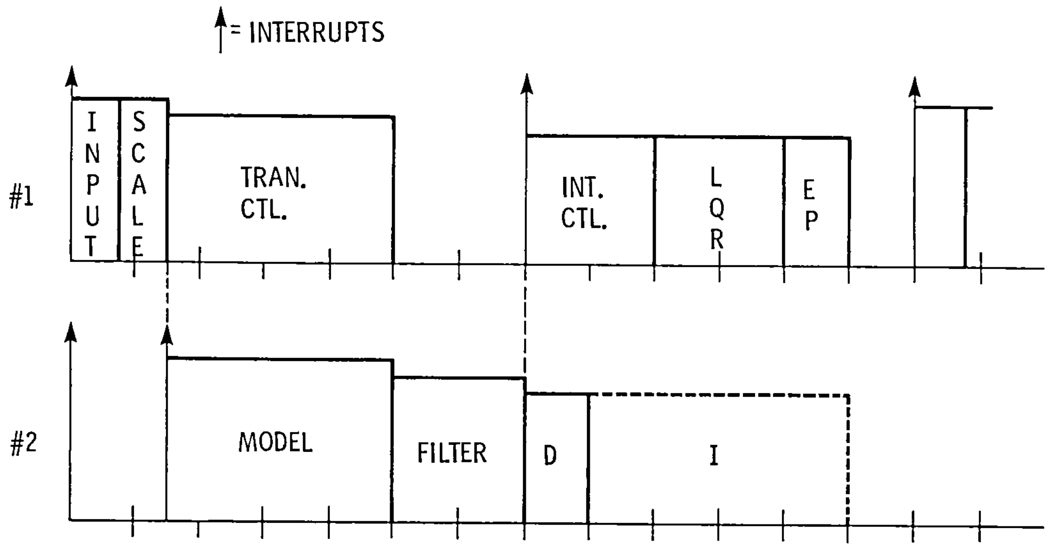


Figure 9. - MVCS-ADIA timing.

1. Report No. NASA TM-83553	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle A Real-Time Implementation of an Advanced Sensor Failure Detection, Isolation, and Accommodation Algorithm		5. Report Date
		6. Performing Organization Code 505-32-6B
7. Author(s) John C. DeLaat and Walter C. Merrill		8. Performing Organization Report No. E-1928
		10. Work Unit No.
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		14. Sponsoring Agency Code

15. Supplementary Notes
Prepared for the Twenty-second Aerospace Sciences Meeting sponsored by the American Institute of Aeronautics and Astronautics, Reno, Nevada, January 9-12, 1984.

16. Abstract
In recent years, advanced control algorithms for turbofan engines have been implemented using digital electronic control mechanisms. However, digital electronic controls need some component redundancy in order to attain sufficient reliability. A sensor failure detection, isolation, and accommodation algorithm has been developed for NASA Lewis Research Center which incorporates analytic sensor redundancy through software. This algorithm has been implemented in a high level language on a microprocessor based controls computer. Parallel processing and state-of-the-art 16-bit microprocessors are used along with efficient programming practices to achieve real-time operation.

17. Key Words (Suggested by Author(s)) Sensor failure; Detection; Isolation; Accommodation; Microprocessor; Real-time		18. Distribution Statement Unclassified - unlimited STAR Category 01	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages	22. Price*

*For sale by the National Technical Information Service, Springfield, Virginia 22161

National Aeronautics and
Space Administration

Washington, D.C.
20546

Official Business
Penalty for Private Use, $300

SPECIAL FOURTH CLASS MAIL
BOOK

U.S.MAIL

Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

NASA

If Undeliverable (Section 158
Postal Manual) Do Not Return

DO NOT REMOVE SLIP FROM MATERIAL

Delete your name from this slip when returning material to the library.

NAME	DATE	MS
K. [illegible]	9-99	130

NASA Langley (Rev. Dec. 1991) RIAD N-75