NASA Contractor Report 179433

Automation Tools for Demonstration of Goal Directed and Self-Repairing Flight Control Systems

A.K. Agarwal

Contract NAS2-12588

August 1988

National Aeronautics and
Space Administration

NASA Contractor Report 179433

Automation Tools for Demonstration of Goal Directed and Self-Repairing Flight Control Systems

A.K. Agarwal
Integrated Systems, Inc., Santa Clara, California

Prepared for
Ames Research Center
Dryden Flight Research Facility
Edwards, California
under Contract NAS2-12588

1988

National Aeronautics and
Space Administration
Ames Research Center
Dryden Flight Research Facility
Edwards, California 93523-5000

Preface

This research was performed under a National Aeronautics and Space Administration Phase I Small Business Innovation Research contract. Support by the NASA SBIR program office is gratefully acknowledged. Technical direction by Mr. E.L. Duke in the artificial intelligence area has been particularly helpful through this phase of the research.

Research at Integrated Systems Inc. was conducted between February 1987 and September 1987 by Mr. Arvind Agarwal and Dr. Naren Gupta.

Contents

List of Figures

1 Introduction

Integrated Systems Inc. (ISI) is pleased to submit the Phase I Small Business Innovation Research (SBIR) Final Report to the National Aeronautics and Space Administration (NASA). This project has involved the feasibility study of using expert system technology for real-time fault detection, isolation and self-correcting control for applications of high performance aircraft flight environments. The goals of the Phase I study have been to analyze several different expert system structures and determine important fundamental aspects required in a real-time environment. The main sections of this report give a research summary of the investigation of each specific area.

1.1 Project Summary

The results of the Phase I effort were as follows:

(i) Determination of key features required to address the issues of real-time on-line monitoring and self-repairing expert systems.

(ii) Design of architecture of embedded expert system required to incorporate and utilize features for real-time use.

(iii) Integration technique required to interface the embedded expert system with an aircraft flight control system.

(iv) Creation of a simple expert system to demonstrate the techniques and the feasibility of the proposed structure to accomplish real-time on-line fault detection, isolation and self-repairing control.

The remainder of this report represents the work supporting the above summary and the feasibility of the design of the proposed expert system.

1.2 Phase I Research Effort

NASA is strongly committed to the strategic goal to develop and apply artificial intelligence technology for aircraft automation including: 1) goal directed guidance, 2) intelligent advisors, and 3) improvements to human error tolerance. Secondly, NASA is committed to provide a national capability for simulation and flight validation of this technology. Three specific areas for which NASA Ames/Dryden is responsible include:

1. Automatic fault tolerance and system reconfiguration,

2. Real-time expert systems to monitor flight test, and

3. The use of expert systems in the air combat context.

These three areas all involve advanced guidance and control technology. Although research in guidance algorithms have advanced technology, it is the incorporation of expert systems, particularly real-time expert systems, which give a much more sophisticated automated system. The effort of the research conducted under this SBIR by Integrated Systems Inc. (ISI) is directed toward the fault tolerant and system reconfiguration area, together with the development of simulation and rapid prototyping tools. It is ISI's feeling that a combination of research of the problem area together with tool development will most quickly advance the state-of-the-art in this area.

The innovation in automation will come from considering both a near term more manageable demonstration problem in the context of flight testing, together with the long term complex and involved problem; namely, automatic fault tolerance and system reconfiguration for high performance aircraft in normal operating conditions.

1.3 Summary

The work performed under Phase I of the SBIR has concentrated on the technology required for the development of an embedded real-time on-line expert system interfaced to an aircraft flight control system. The work has based the foundation and feasibility of the proposed effort to continue and eventually lead into a prototype development of a real-time system by the completion of Phase II.

2 Phase I Research Objectives

Based on the need for goal directed guidance and artificial intelligence-aided fault detection and diagnosis for aircraft automation demonstrations discussed in the previous section, the program here has concentrated on the following research objectives:

(i) Research an expert system architecture for fault detection, isolation, and reconfiguration for aircraft flight control for normal aircraft operation application and demonstration on flight test environment.

(ii) A technology to interface the embedded expert system with data acquisition, computational algorithms, operator displays, and other real-time computation functions.

(iii) Specify key intrinsic features of the expert system necessary for real-time online functioning.

(iv) Develop rules and a knowledge base for a simple aircraft model for the expert system and interface the model algorithms.

(v) Develop scenarios of simulated flight test and faults for demonstration of expert system logic and operation.

The final goal and structure for the expert system and the reconfigurable control system is shown in figure 1. The dividing line between the expert system the conventional numerical algorithms, is to be stressed. This division bypasses the expert system is an easy task and can be done with minimal software effort. The expert system in this structure is meant to enhance the conventional controller rather than replace it. Using this structure will minimize the effort required for implementation, yet offer a convenient package for interfacing peripheral software routines in the future.

2.1 Final SBIR Objectives and Potential Applications

2.1.1 Commercial Applications

Long term commercial applications would be the major aerospace and avionics systems integrators with CAE tools based on this technology to dramatically reduce the development time for advanced fault tolerant and self-repairing flight control systems.

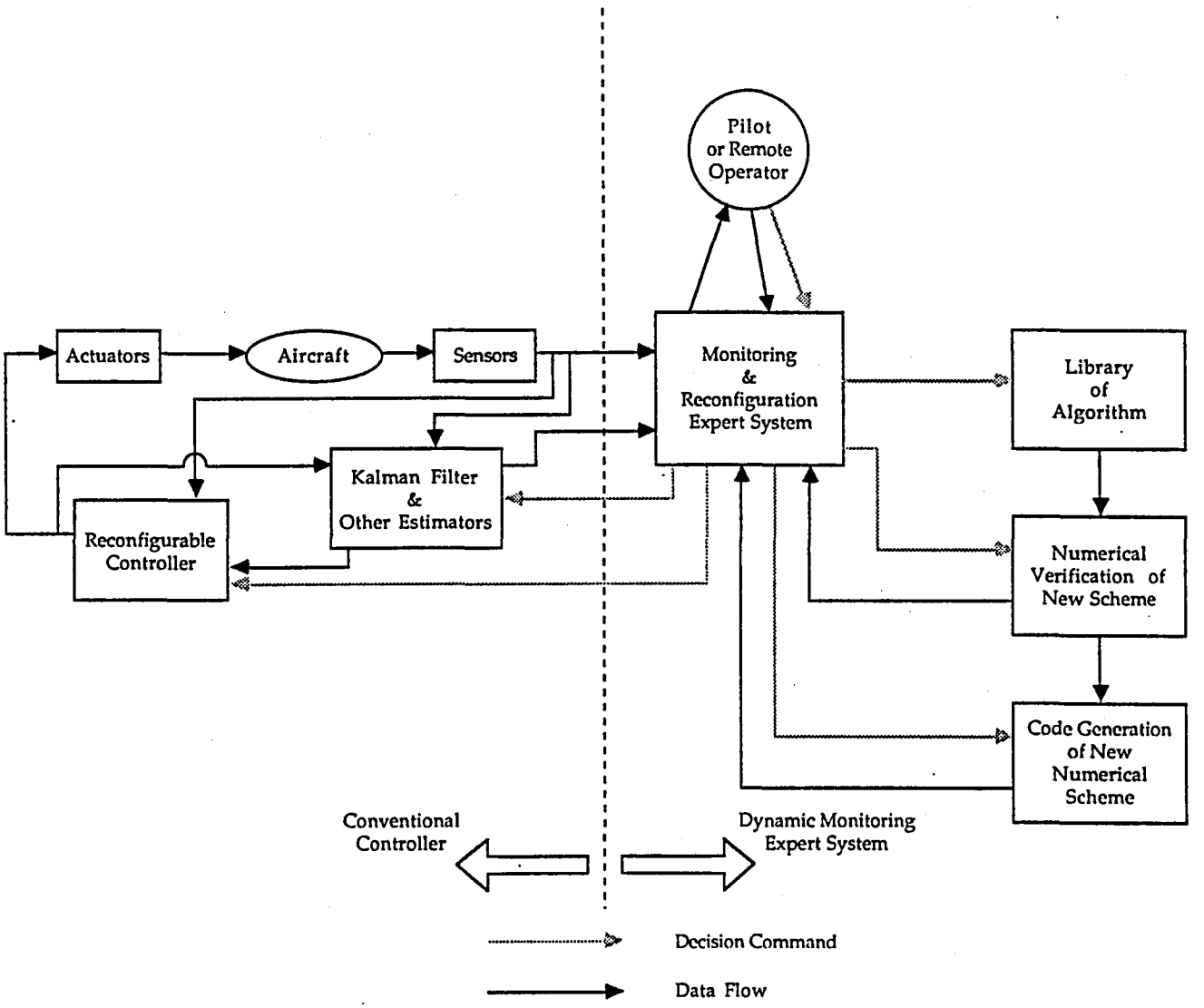


Figure 1: Expert System for Self-Correcting Flight Control

2.1.2 Federal Systems

Advanced aircraft automation is needed by NASA, the Department of Defense, and the Federal Aviation Administration (FAA) for a variety of complex tasks. The technology could lead to significantly improved near term automation demonstrations such as a flight test monitoring, and a rapid introduction of combined guidance/control and expert systems. Autonomous Land Vechicle and Pilot's Associate, DARPA Strategic Computing Programs would benefit directly from this effort also.

3 Technical Background

Fault detection, isolation, and accommodation (FDIA) has been of considerable interest for the last ten years (although suitable algorithms for detection are still a research topic). Whereas the concept of "diagnosis" and "self-repairing" are more recent attempts to structure "intelligent" control systems which include real-time expert systems as well as some standard types of guidance and control logic. There are three ways that real-time expert systems will contribute to this much larger concept for fault tolerant and self-repairing control;

(i) in the diagnosis of flight control systems on-line,

(ii) in the integration of logic based diagnosis and detailed model based analytic redundancy, and

(iii) in mechanizing alternate reconfiguration strategies once the fault is isolated/diagnosed.

We will first summarize an understanding of the most promising diagnostic techniques, then the current state-of-the-art in analytically redundant fault detection, discussing how the expert systems portion will aid in integrating the two types of models, as well as mechanizing a reconfiguration strategy.

3.1 Fault Diagnosis in Aircraft Flight Control

Recent research, both in the area of on-line fault diagnosis [1] and in off-line advisors for numerical algorithms [2], the recognition has been made that underlying models of the situation are in fact the appropriate inference guide, particularily if combined with some general "common sense", or rule-based shallow reasoning. In [1] the deeper inference model came from the mechanical and electrical interconnections. In [2] the model came from a qualitative model of the underlying optimization goal and the monotonic relation between frequency domain analysis criteria and the design goal. For aircraft flight control fault diagnosis we need to consider a spectrum of increasingly detailed models for a sufficiently deep inference foundation.

(i) rule based logic for overall system and some subsystem performance,

(ii) qualitative models on performance changes of flight control subsystems,

(iii) transition models, or a discrete event simulation of the state of the different subsystems in relation to each other, and

(iv) detailed qualitative models for the interconnected dynamics of the aircraft.

In combining recent FDIA analytic redundancy approaches, with real-time expert systems, we may well discover in the research that qualitative or transition models are required to bridge the gap between the technologies. The qualitative model descriptions and evaluations would most appropriately fit into the real-time expert system module, while the run-time version of AutoCode would be modified to incorporate a common data base to be shared with the quantitative model and pass status information to the knowledge base of the real-time expert system. Qualitative models give a predictive element to inference choices which will effect the future without interface to a detailed quantitative model.

3.2 Fault Tolerant Techniques

The most commonly used fault tolerance technique at a hardware/software level include:

(i) Error-detecting and error-correcting codes which correct transmission of data between units of the system.

(ii) Self-checking logic circuits which provide an indication of faults occurring within a unit.

(iii) Machine status and completion signals, programmed exchange of results, checksumming, password, and acknowledgements, which help to achieve fault tolerance in software.

(iv) Duplication and comparison, triple redundancy with majority voting which provide additional fault tolerance in hardware.

The above local sorts of hardware/software based fault checking approaches would continue to be implemented with conventional software techniques except that we would like to interface them to the real-time expert system so that they can be managed in a more comprehensive, but of course, prioritized manner according to the time critical nature of the fault.

3.2.1 Analytic Redundancy

The local approaches have gained favor recently because of the lack of confidence in global models and the simplicity of local FDI. It should be pointed out however,

that for an actuator displacement which is measured, without global FDI, it is not possible to resolve whether

(i) the sensor failed,

(ii) the actuator failed, or

(iii) the surface was partially destroyed and therefore the desired moment effector failed.

Fault detection and isolation, (FDI) or failure detection and identification (FDI) involves two stages, filtering and decision making [3], or residual generation and decision making [4]. Detection is easier than isolation at least from the prospective of implementation, requiring one or several detection filters versus a bank of filters or residual generators in th isolation case with at least one residual generator for each fault isolation required. Once measurements of the system have been filtered, or parameters in a matched filter have been estimated, decisions regarding the nominal state could be based on one or more of the following criteria:

(i) Innovations tests (mean, covariance, spectrum),

(ii) Likelihood-ratios (LR), generalized-likelihood-ratios (GLR),

(iii) Sequential probability-ratio tests,

(iv) Parameter values, parameter bounds, and

(v) Parameter jumps.

While many different decision making strategies can be made to work, and FDI techniques are beginning to be applied, little practical work has been done on the "robust" synthesis of the filters. The work by Chow and Willsky [4], and Kosut, et al. [5] both give a good theoretical framework for the filter design or residual generator design, neither give much detail on the importance of the model structure used to generate the filter and its relation to the determination of the model error necessary for robust design. Gupta and Walker [3] start with a more intuitive approach toward the model error and extend frequency shaped LQG synthesis concepts to the FDI problem. FDI is part of a larger objective to develop reliable, reconfigurable controls, initiated several years ago which has been discussed in the literature [6].

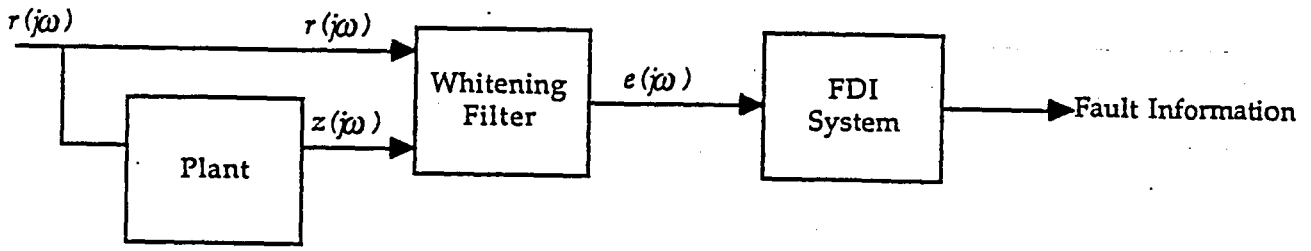


Figure 2: Whitening Filter Realization of FDI

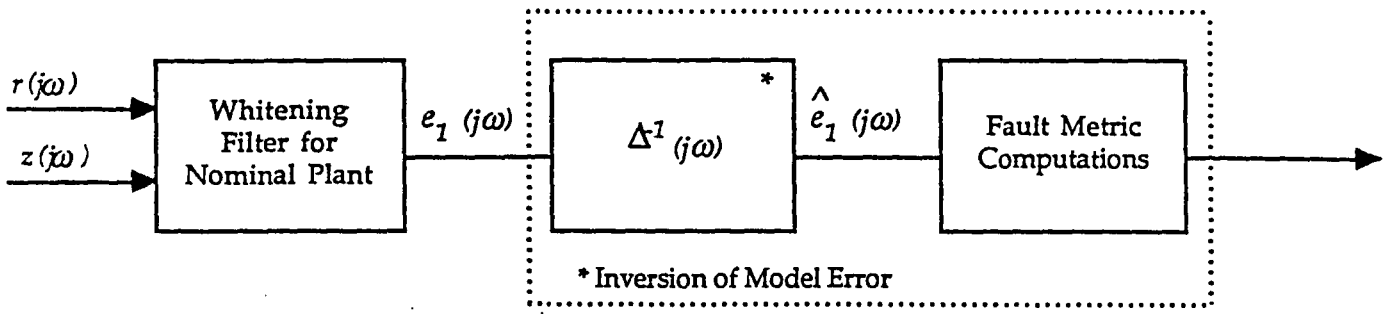
3.3 Analytic Redundancy Approach

One possible scheme for analytic redundancy is shown in figure 2 where a whitening filter is designed to map the plant reference and output signals into the error signal $e(j\omega)$. This signal is then used for FDI. An alternative approach is illustrated in figure 3. Here the whitening filter is designed to generate intermediate error $e(j\omega)$ from a model of the nominal plant. This nominal error is then filtered by a state space representation of the inverse model error $\Delta^{-1}(j\omega)$ for the respective detection or isolation filter. With any practical implementation of either filter, the error from the augmented model must be filtered by a suitable frequency shaping filter $\delta^{-1}(s)$, as shown in figure 4, to ensure robust operation where $\delta^{-1}(s)$ satisfies the following relationship with respect to the true model error $\Delta_0(s)$, and $\Delta(s)$ the assumed model error:

$$|\Delta(s) - \Delta_0(s)| \leq \delta(s) \ .$$

The shaping function filters low and high frequency uncertainties in the error model response, so that the sensitivity of the FDI filter system to uncertainties is greatly reduced. Therefore undesirable false alarms, missed alarms, as well as failure identification are minimized. The design of each filter type must include as much information about the particular fault as possible in order to ensure robust performance.

As described above a bridge between this analytic, detailed quantitative approach and more abstract models which incorporate broader issues and constraints will be a major focus of the research. Work on monitoring flight tests will give a practical logical starting point to build up a framework suitable for addressing the complexity of fault tolerant flight control overall.



The above system is equivalent to:

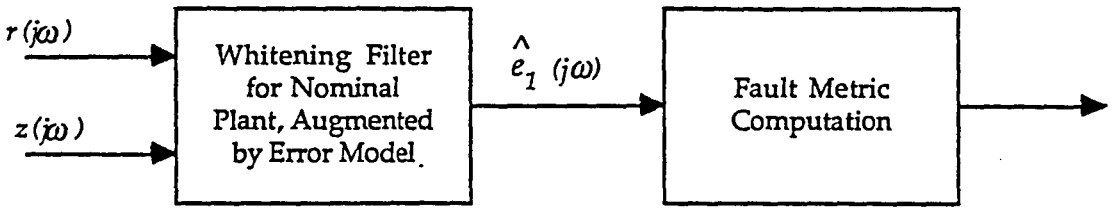
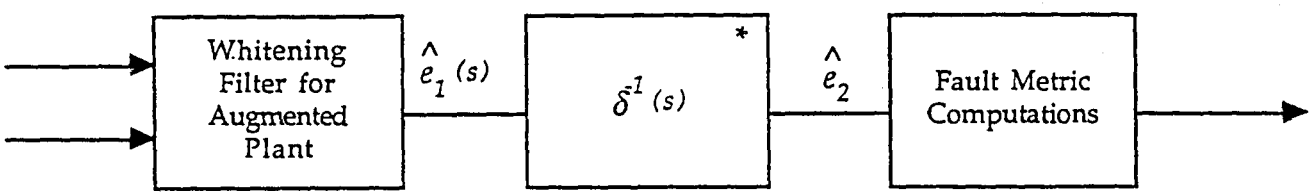


Figure 3: Whitening Filter with Inverse of Model Error



* Frequency Shaping Function used in Robust Estimator Design

Figure 4: FDI System with Frequency Shaping

Self-repairing control is an entirely analogous area of research. Once the diagnosis portion is clearly laid out, this task focuses on incorporating advanced approaches such as on-line uncertainty estimation and implementation rules for reduced order adaptive control. These algorithms will be controlled by high-level rules and simple procedural models by a real-time expert system. The architecture specified will take into account the requirements necessary to program such functions.

4 Real-Time Requirements for Expert System Development

The proposed integration of artificial intelligence and trajectory guidance research will focus on an advanced generic real-time development environment for self-repairing flight control system. While the software will be designed to facilitate the implementation of advanced analytical feedback control laws, the emphasis in this project will be on the development of real-time expert system technology lacking in existing expert system shells. The ability to interface analytical databases with a logical knowledge base will be a key factor in the development of the embedded expert system.

4.1 Embedded Expert System Development

While considerable work has been done on expert system technology and development shells, little attention has been given to expert systems for real-time on-line processes. There are several other packages which have attempted to address this significant discipline. These packages have addressed certain issues required for real-time use, such as shared memory and time tagging of data, but they have neglected the issue of scheduling needed for time critical tasks required for on-line monitoring.

The key features that will be developed for the proposed expert system under this program will incorporate the following functions.

4.1.1 Scheduling

Real-time expert systems must produce the best conclusion within a fixed amount of time. In many cases, the first conclusion will not be the optimal solution due to a possible time critical situation. This requires that some sub-optimal solution be made quickly and refinement to the solution be done as time goes on.

Since the expert system may be required to handle many tasks, it should be possible to assign priorities to various parts of the heuristic knowledge and have the search procedure be interruptible by other tasks which have a higher assigned priority. The scheduler also needs to have the ability to dynamically reassign priorities depending on the incoming data and the stated objectives.

4.1.2 Multirate Execution

Various aspects of the system may require a higher level of evaluation due to the dynamics or reliability of the subsystem. It should be possible to specify the frequency with which various conditions are determined or faults are evaluated. This evaluation will work in coordination with the scheduler in sorting the priorities of the tasks. This will also allow significant savings in computation time where the expert system is using precious time wisely where it is most needed.

4.1.3 Hierarchical Searching

The user should be allowed to specify a hierarchical priority or any other desirable order of search. This makes it possible for the user to assign varying levels of importance to different faults or other conditions. For example, in a particular operating environment, certain fault characteristics might have a higher level of significance in the functioning of the system. This allows the user to cater the requirements of the expert system depending on the performance objectives of the operation.

In many cases, the priority of search or detection might change dynamically depending on certain events that have occurred at an earlier time. This aspect is tied closely with the scheduler and the structuring of the knowledge base. This function will be considered when determining the overall structure of the system so that some form of dynamic reallocation of hierarchical priorities is possible.

4.1.4 Time-Tagging of Data

Due to the dynamic nature of the data acquisition being performed, it is necessary both to insure that correct data is used for decision making and to have the historical data available for referencing and comparison. By time-tagging data, it is possible to determine rate of change of states which, in certain conditions, can be analyzed with less complex relationships.

4.1.5 Dynamic Truth Maintenance

Many short term decisions or sub-optimal decisions are used in long term analysis. The short term solutions have been made on the bases of old data. The validity of the short term solution made in the past needs to be checked before being applied to current long term solutions.

4.1.6 On-line Data Interface

The system must be capable of sampling sensors or inputs at specific intervals and storing the data in a database where the expert system will have access to the value and the time frame that the data represents. This will constitute the shared memory between the embedded expert system and the data acquisition system.

4.1.7 Interface to Computational Algorithms

A certain amount of coupled knowledge of the expert system and the computational algorithms will be required in the rules developed. This interface makes it possible to take extended action (for example, reconfiguring an estimator) based upon certain decisions. Without knowledge of the computational algorithms and the effects of it on the system, commitment to a decision by the expert system will be near impossible.

4.1.8 Specified Levels of Autonomy

With a pilot or a remote operator in the loop, the system can be tailorable to the level of commitment required. This allows the pilot to have the ultimate control of the expert system. Possible levels of autonomy are:

(i) do it,

(ii) do it and tell me,

(iii) tell me and then do it,

(iv) tell me, but don't do it.

The operator will specify the level of commitment and decision making automation allowed of the expert system.

4.2 Expert System Software Structure

The embedded expert system will have the structure shown in figure 5 to satisfy the need for real-time execution. Note that the scheduler effects the data acquisition, database and the inference engine giving it the level of management required to have an interrupt-driven system capable of having multi-rate execution.

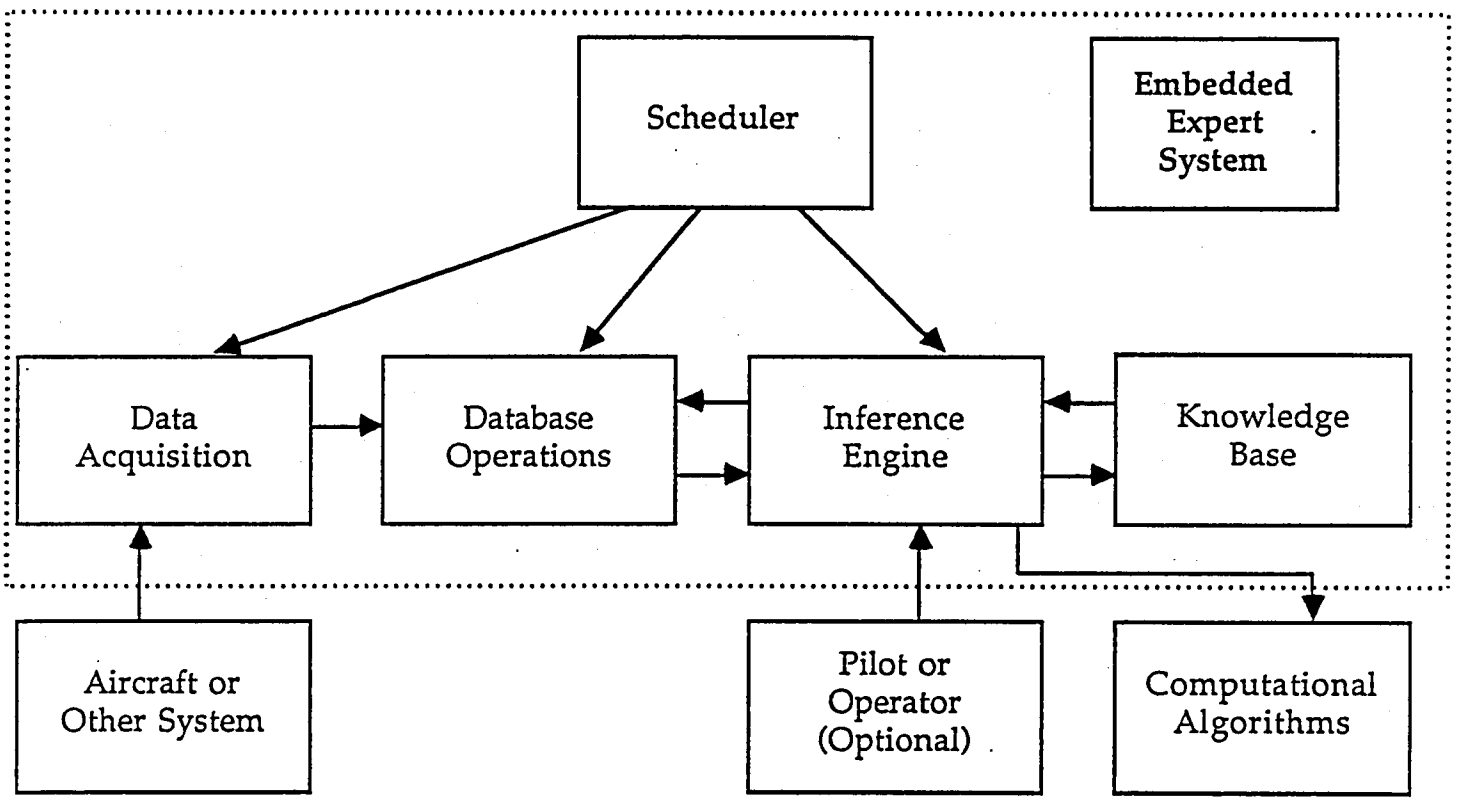


Figure 5: Structure of Embedded Expert System

The characteristics of the key embedded expert system components are described in the following subsections.

4.2.1 Knowledge-Base Development Kit

This is an off-line module used to create the knowledge base. This would be similar to any expert system shell with the exception that it will make it easier to deal with numerical data. In particular, the following features will be incorporated.

- Initialization and re-initialization
- General logical expressions on measured data and logical flags (IF-THEN-ELSE, unconditional, or general logical equations)
- General arithmetic assignment statement
- Statistical or other operations on time histories (i.e., rules that deal with time histories or time history statistical trends rather then individual numbers)

The other key feature of this off-line system, will be an environment needed to create partitioned knowledge and rules bases. This will allow the user to enter knowledge relevant to a specific aspect of the system in the portion of the knowledge which contains information specific to that sub-area. By partitioning the knowledge and rule base, search can be performed more efficently which will be needed in real-time. This idea of partitioning is presented with a system break down for the demonstration of a F-18 aircraft in section 4 of this chapter.

4.2.2 Scheduler

The scheduler will be similar to that for any interrupt-driven real-time process. This module must allow concurrent operation of database and inference functions. To optimize real-time performance, the scheduler may have to be problem specific. Then, it will be necessary to have a code generator for the problem specific scheduler.

4.2.3 Data Acquisition

This module should include standard input/output drivers to accept data from any of a variety of sources. Multiple sources may be used in parallel. Appropriate interfaces to the database must also be provided.

4.2.4 Database

The database should be suitable for storing time histories and time-tagging of all data. It should be structured for efficient access both in the storage and the retrieval mode.

4.2.5 Inference Engine

The inference engine is the most sophisticated part of the entire software and must be small and efficient for real-time use. It must be able to operate with interrupts from the scheduler. Other characteristics desirable are:

(i) Provide the best inference in a fixed computation time,

(ii) Provide its best inference at any time (e.g. following an interrupt),

(iii) Provide preliminary inference quickly followed by a more complete evaluation of the situation at a later time, and

(iv) Allow iterative operation such that inferences are refined after each sample point.

The embedded inference engine should in general be able to deal with time histories rather than individual value of each variable. Often inferences are based on statistical information about time histories and changes in trends in various time histories. Many concepts from state transition diagrams (STD) could be useful in this development.

The inference engine will be developed around the C Language Inference Processing System (CLIPS) developed by NASA Langley which will be delivered to ISI on the start of the Phase II program.

5 Method of Integration

In developing an expert system for flight monitoring and self-correcting control, the main issue in concern is the dynamic nature of the decision making that must occur. The key issues in this project are:

1. The dynamic nature of the system and information in decision-making or operation of expert systems,
2. Defining the dividing line between the monitoring expert system and the conventional numerical algorithms, and
3. The expert system used for enhancement of conventional control rather than replacement.

The final objective of the Phase II SBIR will be to develop a prototype expert system for self-correcting flight control. A diagram of the structure is shown in figure 1. The stages of the development are outlined in the following sections.

5.1 Step 1: Sensor Failure Recognition

Shown in figure 8, is the diagram for this step. This step will require creating rules for sensor characteristics to be used. This includes limitations, maximum variations and noise characteristics. The basis for this has been developed in the Phase I of this program, the results are discussed in the following section.

5.2 Step 2: Inter-Dependency of State Recognition

This step requires assuming a simple model for the relation of states. Creating simple trend type rules for increasing, decreasing and constant values to draw conclusions about expected characteristics of other states. Being able to determine a noisy environment and apply filtering (Kalman Filtering) as needed on the output. This part of the work is shown in figure 9.

5.3 Step 3: Dynamic Failure Recognition

This step requires closing the control loop with the expert system monitoring the input and output of the system. Rules and knowledge will be added about the

effects of control on the state of the system. The expert system is not yet making any type of decision, just monitoring the control and trying to determine mishap or unexpected behavior of the closed-loop system. The work in this stage is shown in figure 10.

5.4 Step 4: Dynamic Failure Recognition and System Identification

The numerical model of the system will be interfaced to the expert system. It will be able to simulate the effect of the control on the model to determine the validity of the outputs of the system. The system should be able to look at trends of the output to determine a state run away or system failure. This step is shown in figure 11.

5.5 Step 5: Failure Recognition and Model Reconfiguration

Knowledge of system modeling and control will be added to the expert system. The expert system will be able to detect some form of failure and determine what corrections need to be made and numerically verify its decision. This would be a iterative process where the expert system will try to find the most optimal solution in the appropriate amount of time. This also means it must understand the time critical nature of certain failures so that some form of decision is made within some critical time frame. This is shown in figure 12.

5.6 Step 6: Failure Recognition and Dynamic Control Recognition

The reconfiguring scheme is shown in figure 13. The expert system will now be able to update the control on the system with the decision making ability of Step 5. The expert system will also need to understand how, for different situations, the transition needs to occur. The expert system at that point must realize the failure that occurred and contiuously analyze the failure with all its inferences. If the system realized that the failure was diagnosed incorrectly, it must come to a new conclusion and reconfigure again.

5.7 Step 7: Expert System for Dynamic Environment

The diagram for this is the final objective of the program, shown in figure 1. The decision paths will be put in for updated kalman filtering as needed, the user decision flow is added to over ride any decision that the expert system comes too. The system will generate replacement code on-line for the controller update. Previously the expert system was using existing code in the library of algorithms and inserting in the controller. This way, the expert system will replace/modify the control with the most appropriate control parameter and structure.

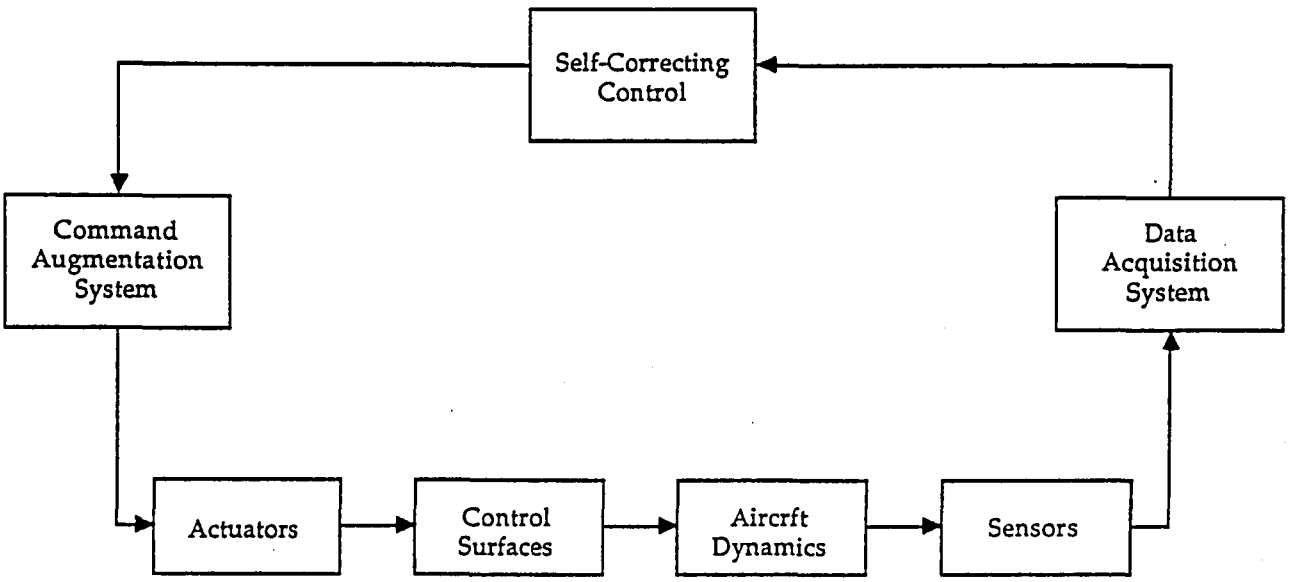


Figure 6: Modular Breakdown of Closed-Loop Aircraft Control System

6 Demonstration of Self-Repairing Flight Control System

Figure 1 shows the final integration objective of the embedded expert system with an aircraft flight control system. The emphasis of the overall system is the capability of self-repairing control based on a control objective. This section will outline the modules of the aircraft and specify the types of faults to be determined. The modular breakdown of the system can be seen in figure 6.

Work on self-correcting control has been done in the past by Alphatech. During the Phase II contract effort, this work will be evaluated for applicability to this demonstration and incorporation of the concepts into the final objectives of Phase II.

The proposed scheme breaks the components of the aircraft into four main modules. The diagnosis of the system in the event of a failure is done on the bases of these modules. This allows for a efficient search strategy where computation time is not wasted trying to isolate particular failures in a region where the failure is

known not to occur. The scheme for diagnosis, isolation, and correction is shown in figure 7. The fuzzy region represents failures which can not be diagnosed in any particular module but instead restricted to a certain region between two consecutive modules.

This task will use a complete 6 degree-of-freedom (6-DOF) nonlinear model of an F-18 aircraft. Thrust vectoring and nose strake effects will be included. This model will provide a good test for the expert system technology, because multiple control needs are available and the flight envelope of the aircraft covers diverse operating regions.

A nonlinear digital simulation of this aircraft has been developed by NASA. ISI will need to obtain this simulation for use in this part of the program effort.

The following sections summarize each of the aircraft component sub-system models.

6.1 Actuator and Control Surfaces

In terms of diagnosis, it is near impossible to detect a failure between these two modules. The only information available is the input to the actuators. Given a control input to the actuators, there is no way of getting data about the output of the actuator of the input to the control surfaces. Due to this, in the case of the control surface breaking off, that would be covered in the module of dynamics. These modules deal strictly with the nature of uncontrollability of the aircraft. The diagnosis would lead to a isolation of the failure and if possible a model reconstruction if the mission objective can be met without the control of the isolated control surface. Otherwise this would result in a situation where the operator is informed of an abort alert.

6.2 Aircraft Dynamics

This module covers all aspects of the F-18 aircraft which are included in the mathematical model of the system. When outputs of the model differ from the actual outputs of the aircraft in some controlled manner, the diagnosis is the inconsistency in the numerical model which means either the controller has to be tuned or the system model has to be modified. This module will be the largest part of the embedded expert system self-repairing flight controller. Again, there are fuzzy areas on both sides of this modules. These are cases where the control surfaces are not functioning correctly or the sensors are giving inconsistent readings. These areas

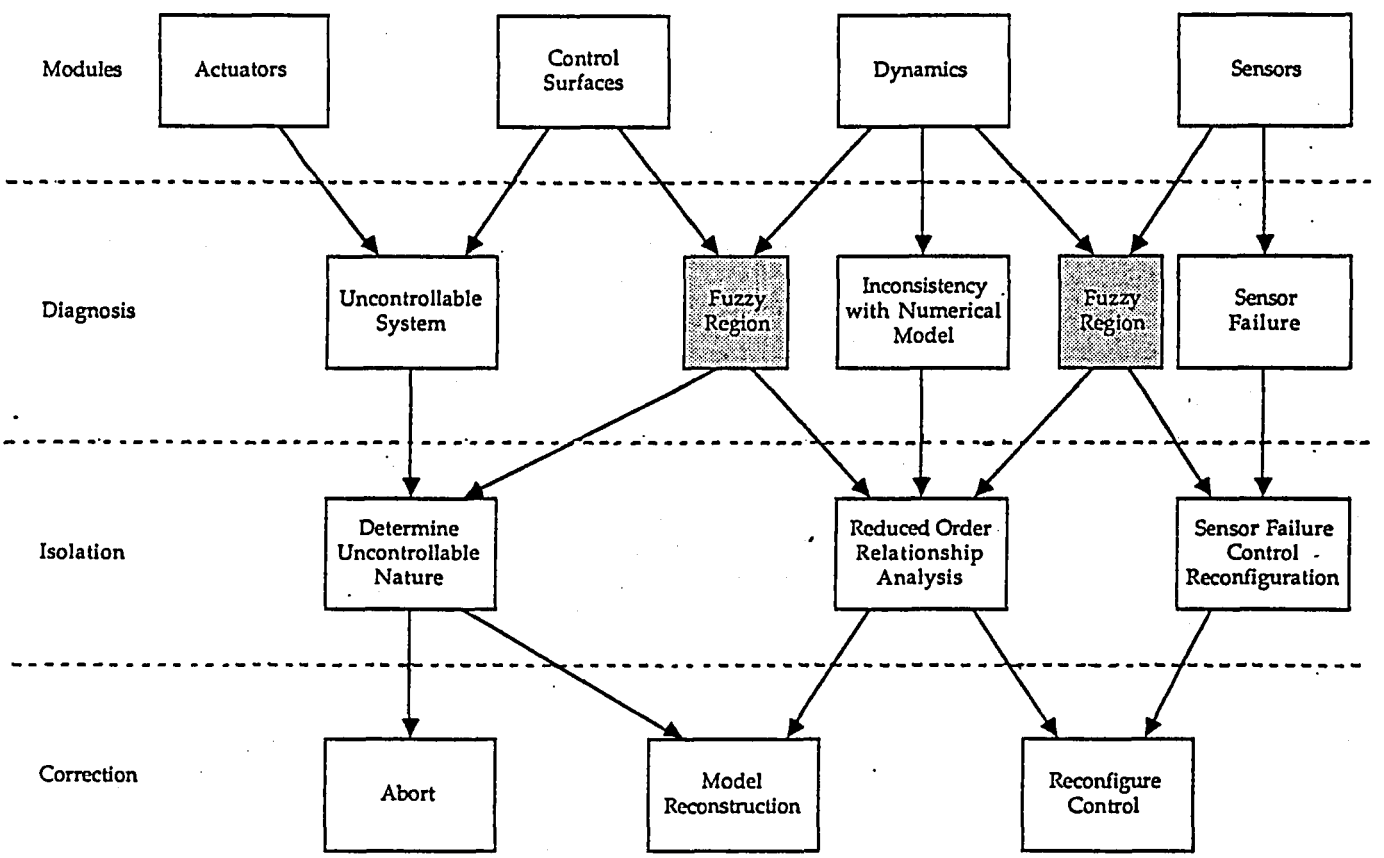


Figure 7: Fault Diagnosis, Isolation and Correction Scheme

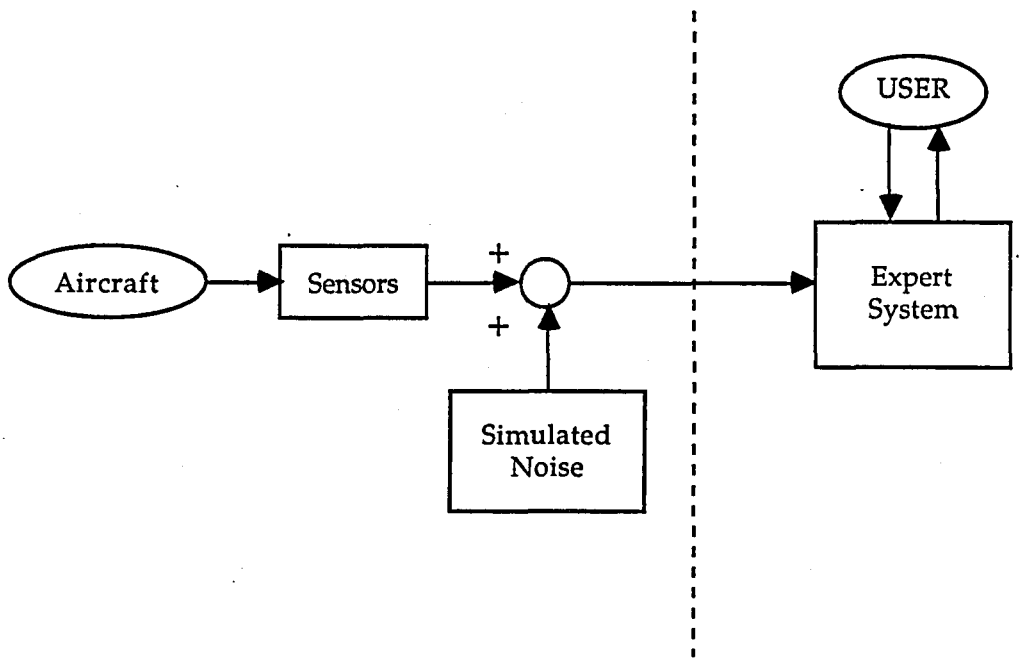


Figure 8: Sensor Failure Recognition

could be diagnosed as an inconsistency with the numerical model. In this case, the isolation task will take much longer due to the uncertainty of the diagnosis.

6.3 Sensors

Sensor failures will be detected similarly to the method used in the EXSYS demonstration. A frame will be created for each sensor with its noise characteristics, variation allowance, limitations, etc. Similar rules will be used to determine validity of the sensor readings. Once a particular sensor is diagnosed as a failure or as a malfunction, the isolation will consist of using an observer for the state or an appropriate filtering technique. The action for this case will result in the reconfiguration of the control depending on the mode of the failure.

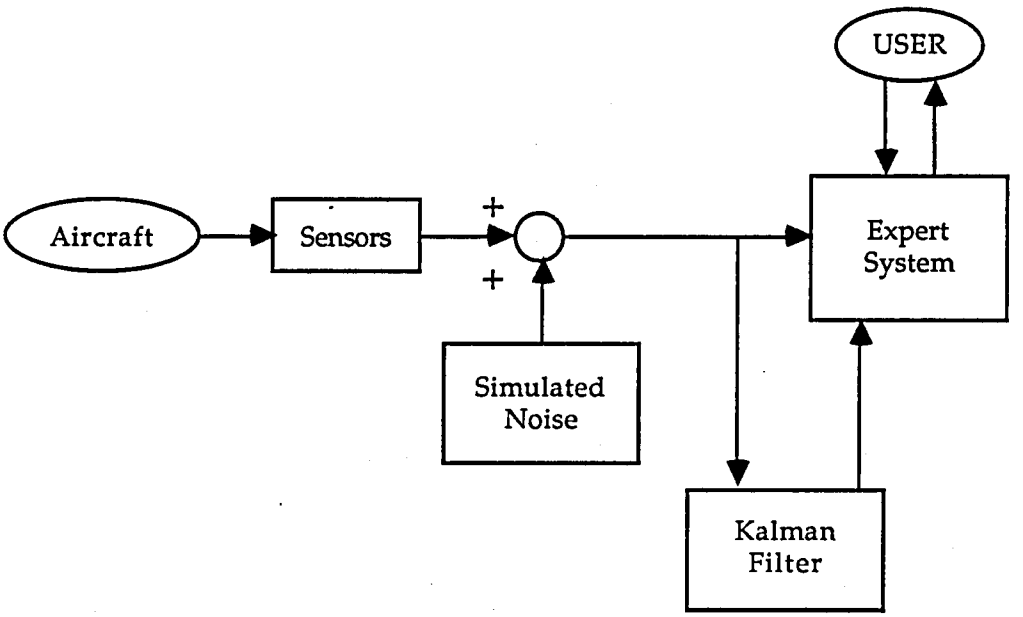


Figure 9: Inter-Dependency of State Recognition

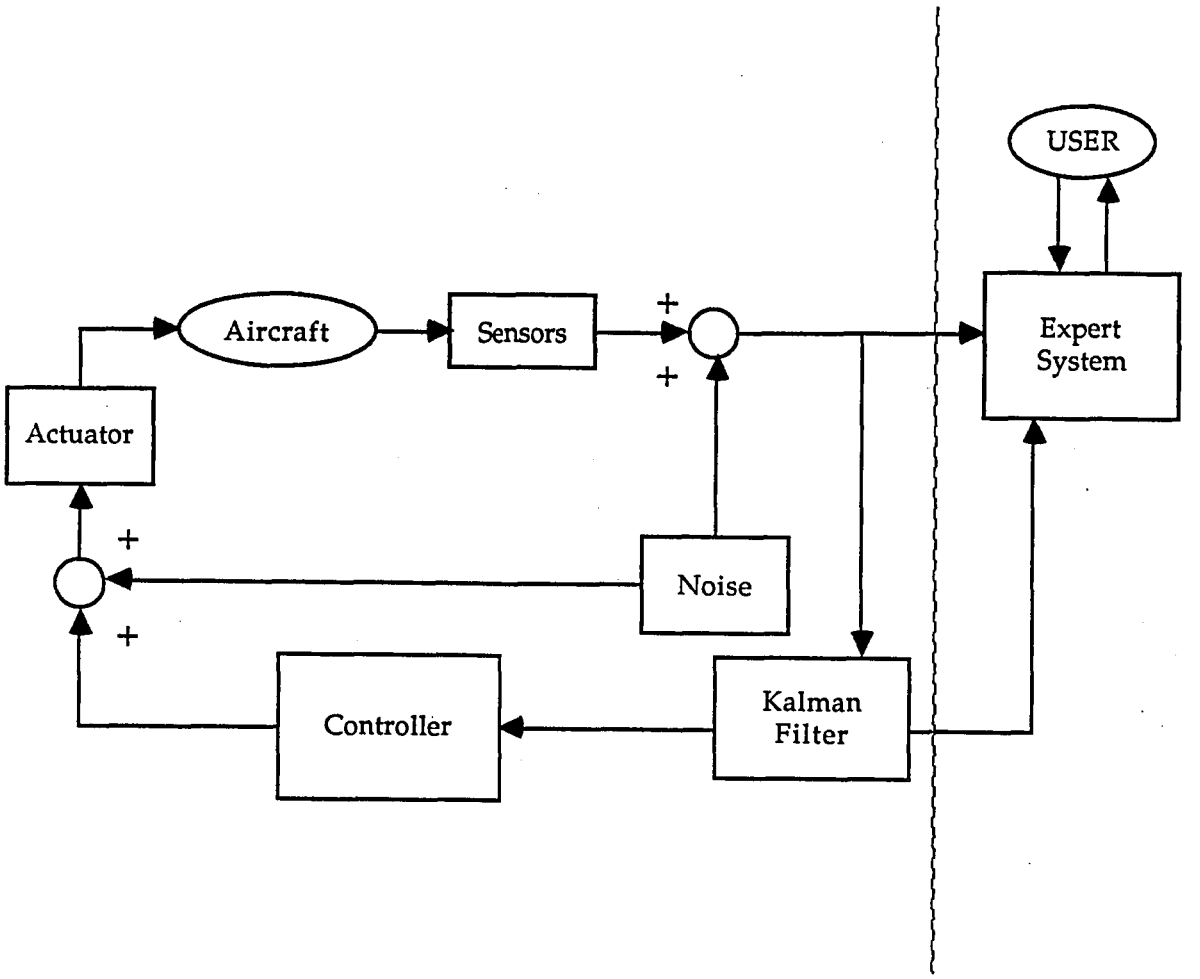


Figure 10: Dynamic Failure Recognition

7 Simplified Example Using Exsys

A simplified expert system for the purpose of demonstrating fault detection was created. This demonstration system incorporated the first three steps outlined in the previous section for interfacing of real-time expert system with an aircraft flight control system. This example incorporates the basic ideas presented earlier in a working demonstration.

7.1 Sensor Failure Detection

The rules that where created dealt with sensor failures and simple state interactions. Simple characteristics of sensor readings were detected such as,

1. Sensor reading outside of limit,
2. Sensor reading in high noise environment,
3. Sensor varying faster than desired, and

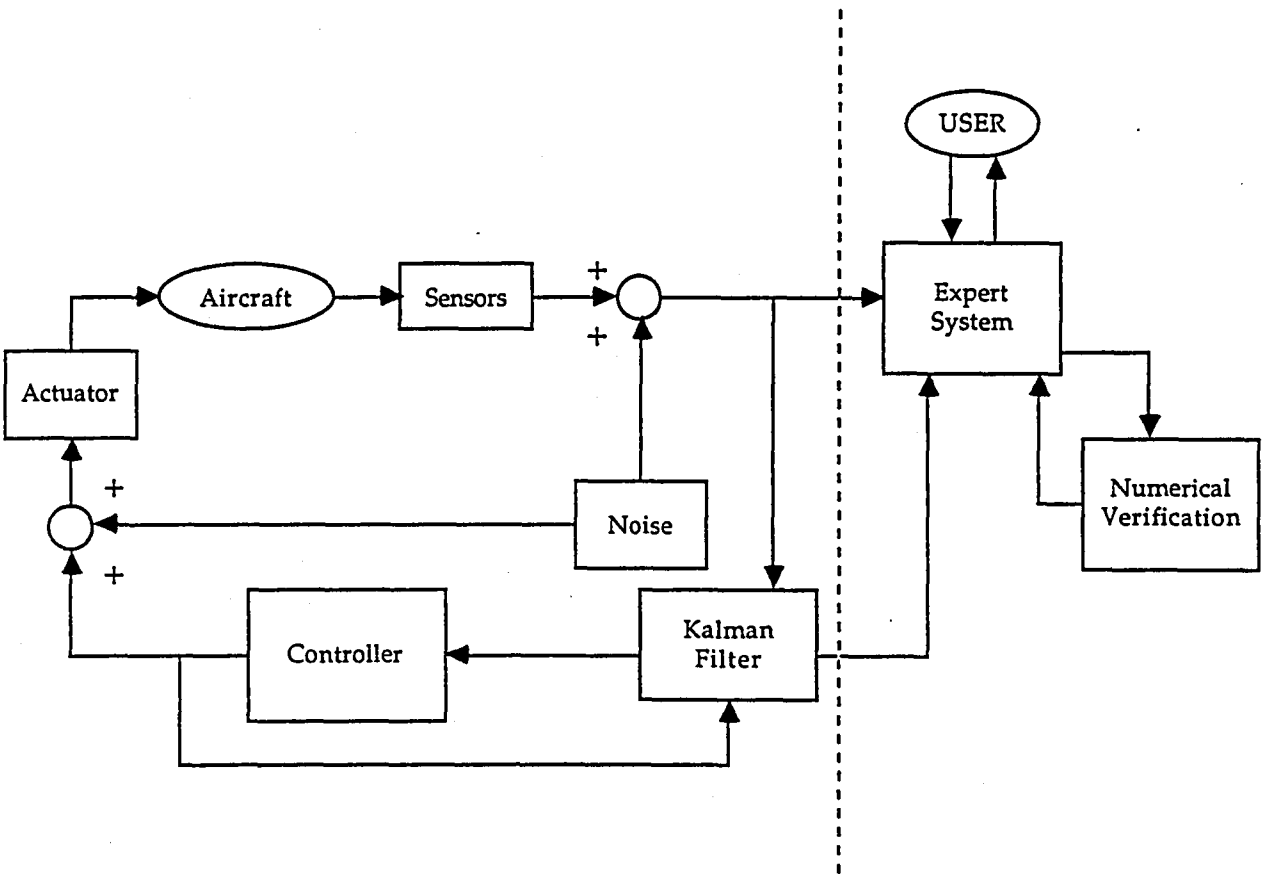


Figure 11: Dynamic Failure Recognition and System Identification

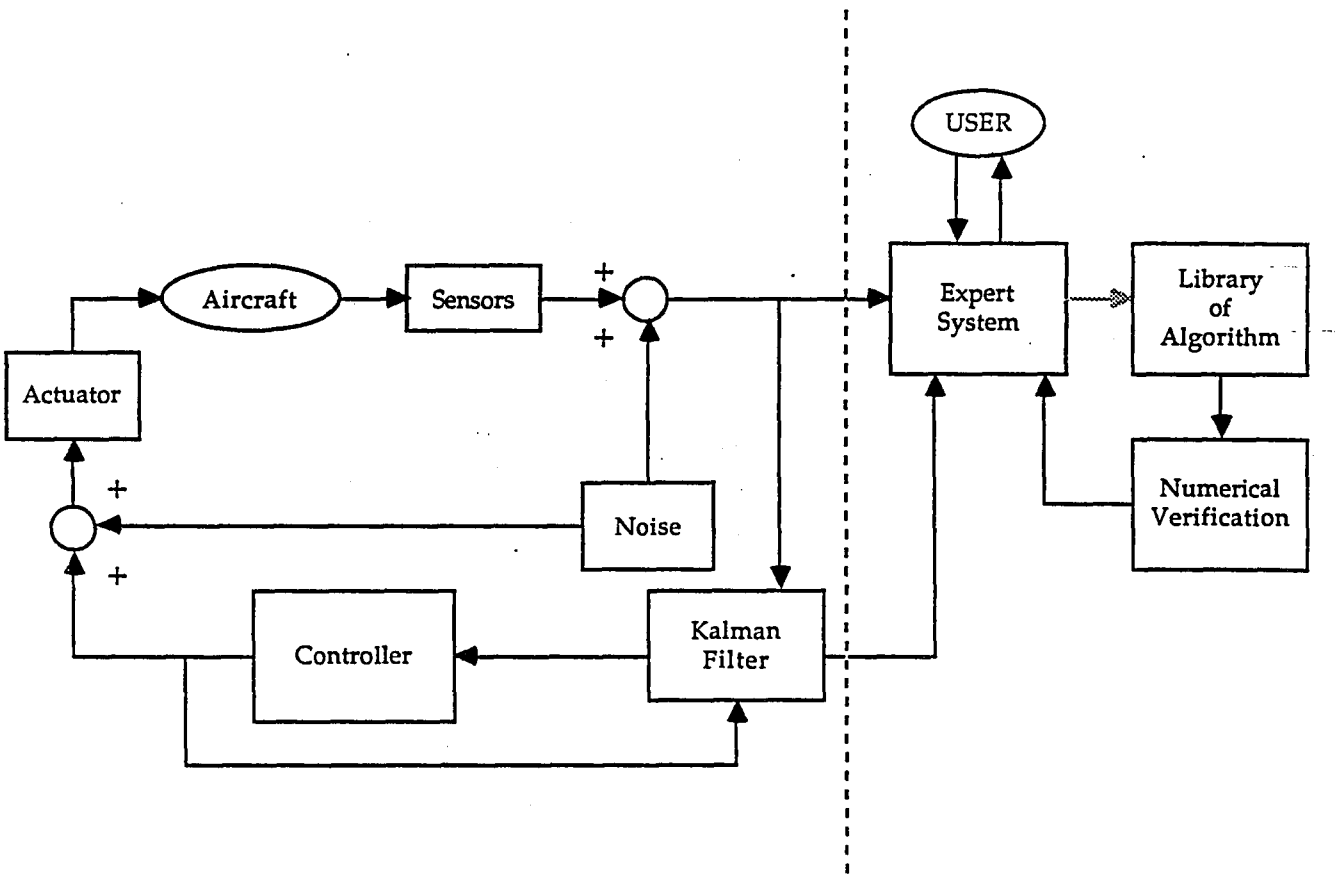


Figure 12: Failure Recognition and Model Reconfiguration

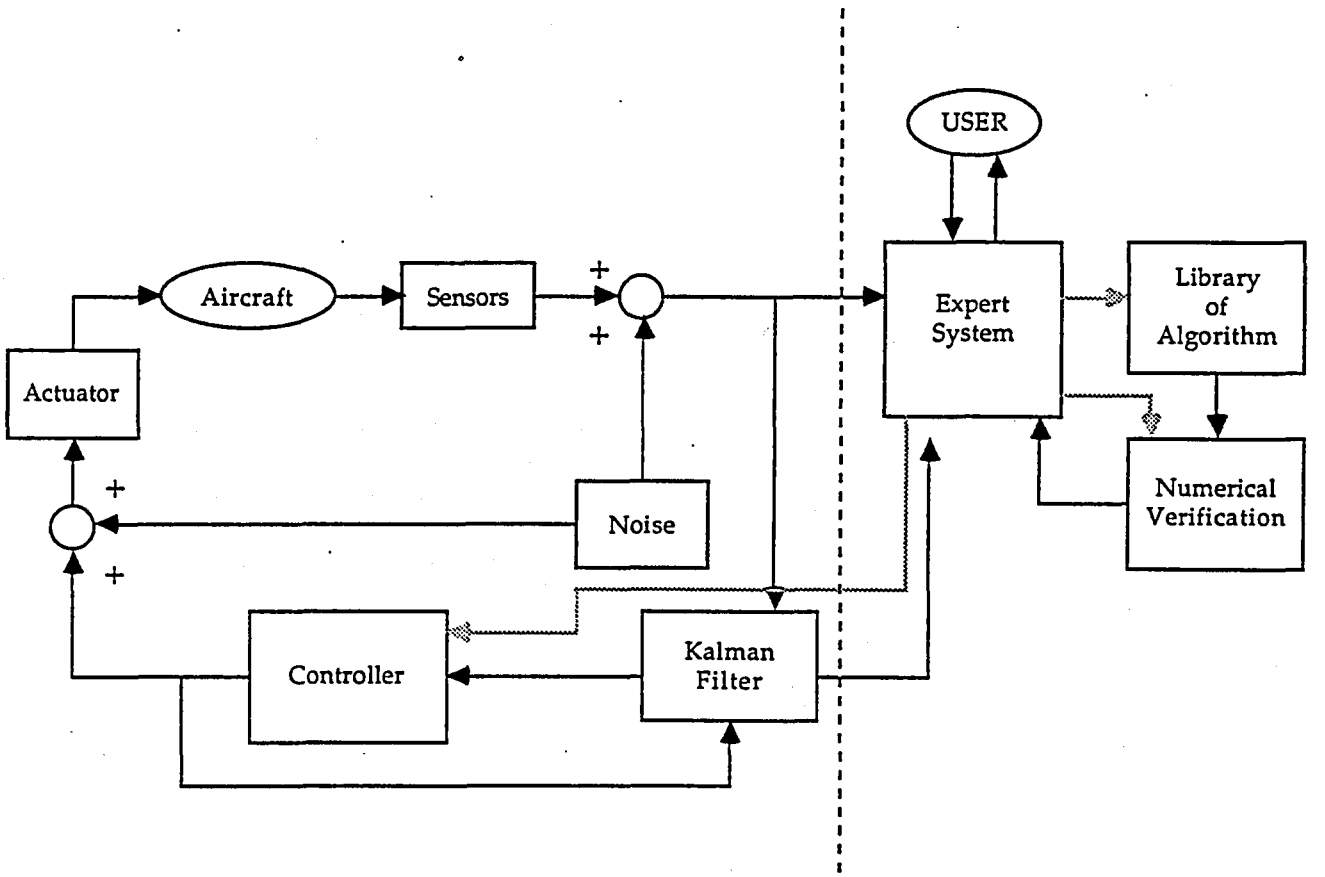


Figure 13: Failure Recognition and Dynamic Control Recognition

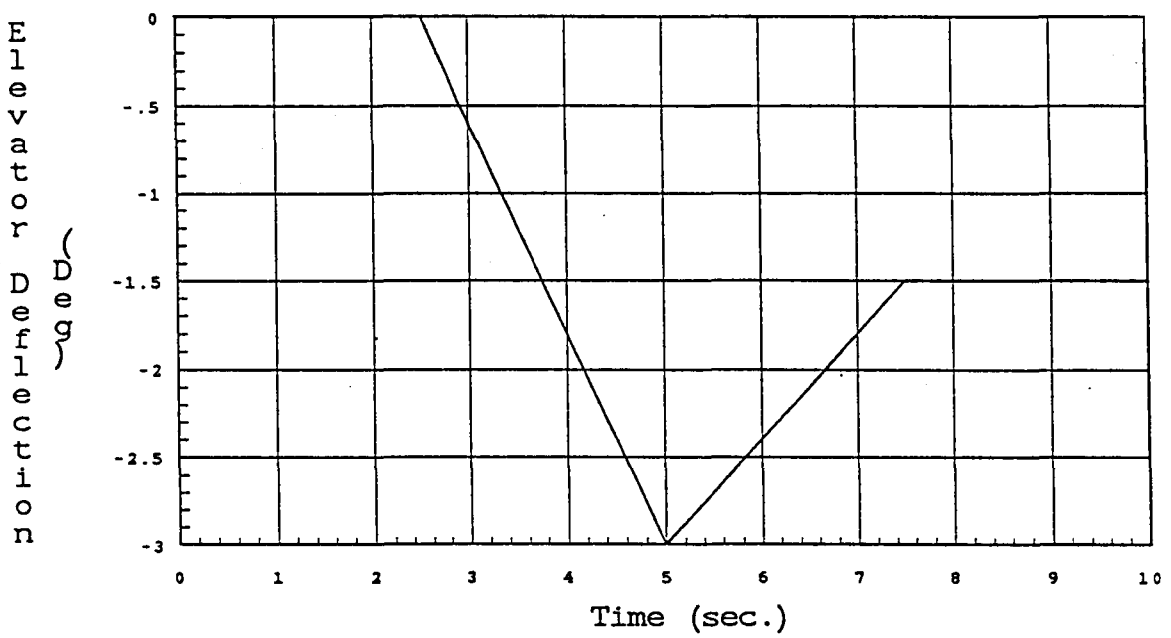


Figure 14: Elevator Deflection

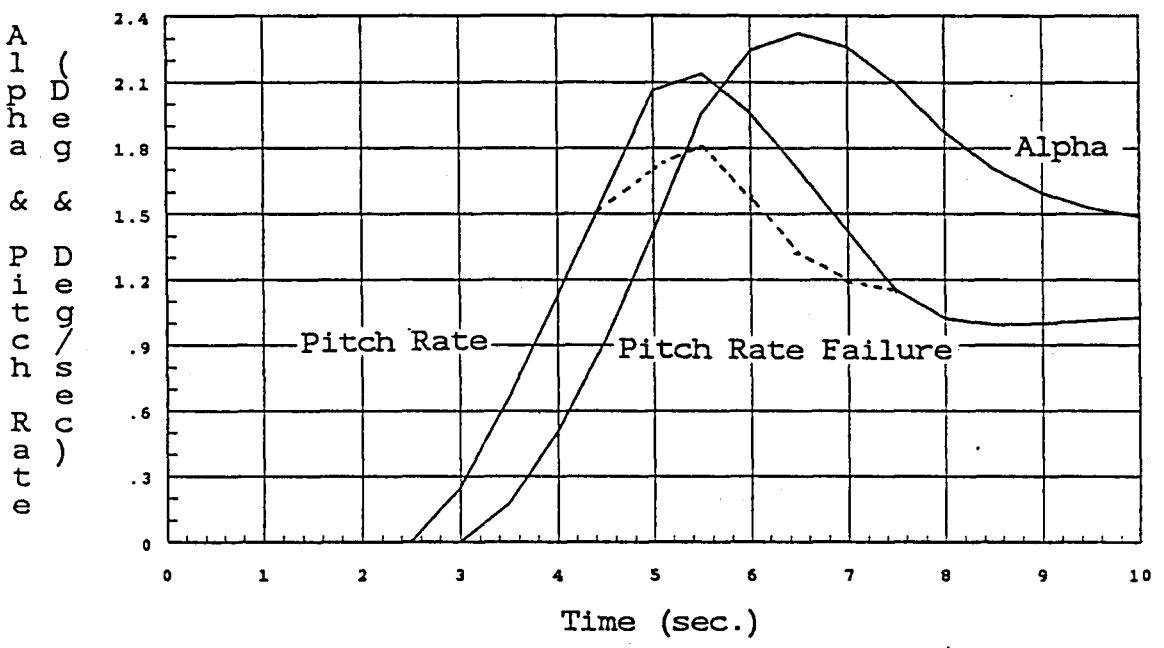


Figure 15: Error in Pitch Rate Sensor

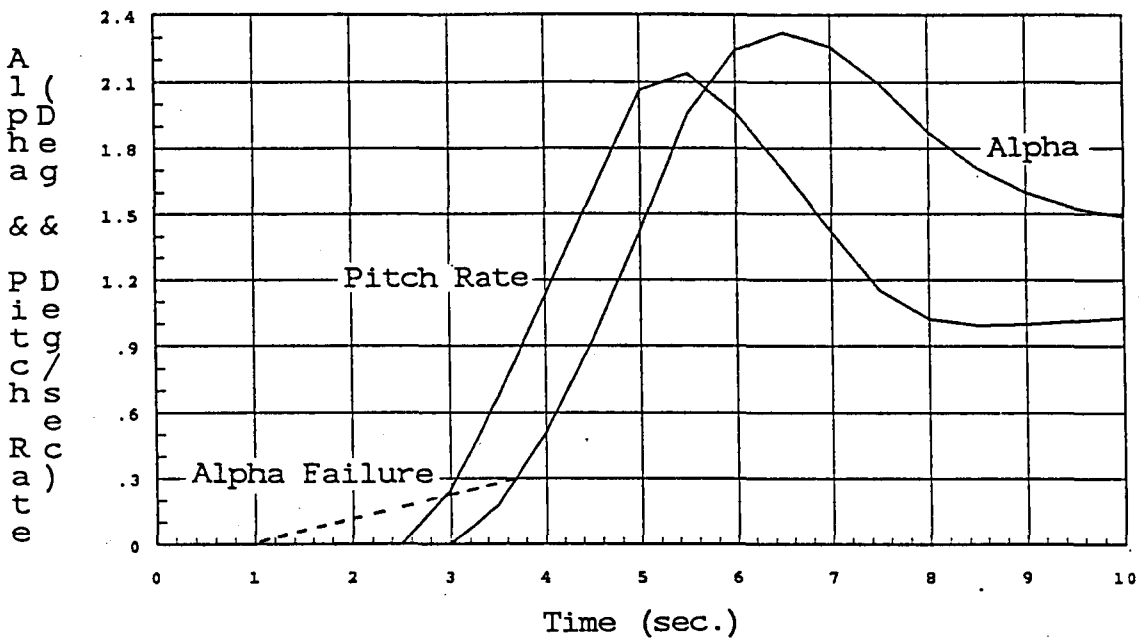


Figure 16: Error in Angle-of-Attack Sensor

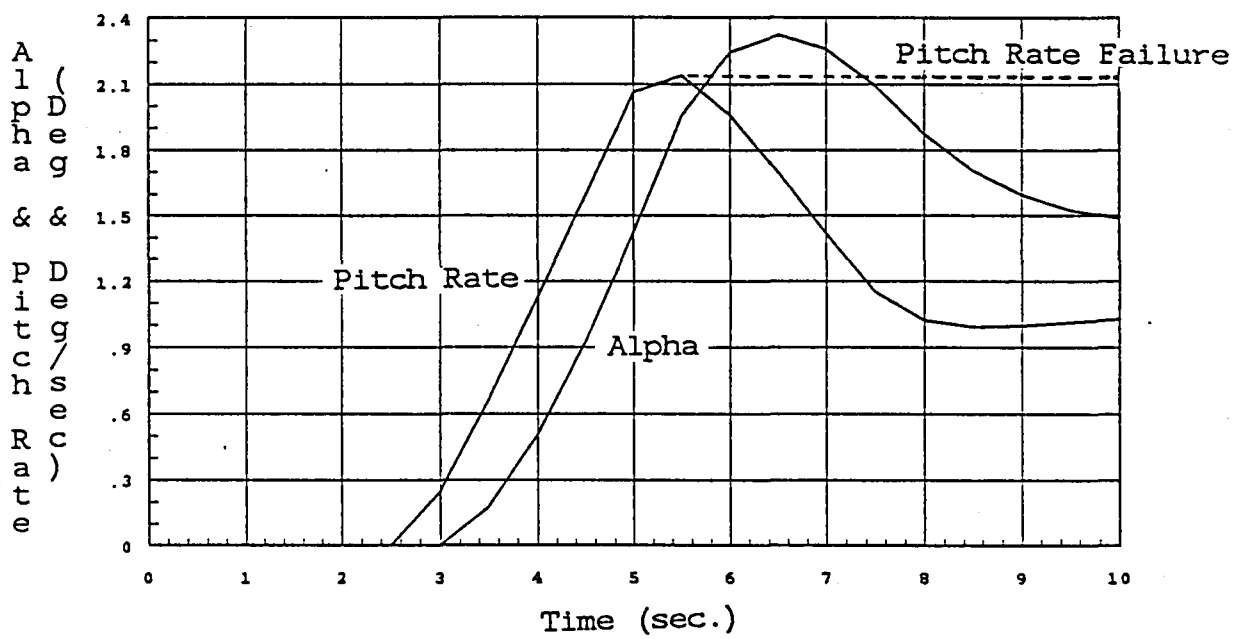


Figure 17: Pitch Rate Sensor Failure

4. normal operation.

For the expert system to come to these conclusions, some characteristic knowledge of the sensor and the state, which it represents, was required. The limits allowed on the sensor reading corresponded to the physical limitation applied to the sytsem. The maximum rate of change was also required along with the sampling time of the readings. With this information, conclusions could be made about the noise environment and the validity of variations in the sensor readings.

7.2 Analysis Technique

Rules were created on the basis of a three point analysis. At time $T = t_3$, conclusion were drawn based on information from t_1, t_2, and t_3 just as at time $T = t_4$, conclusions were made based on t_2, t_3, and t_4. By doing this, individual readings are not considered due to inherent noise characteristics expected from real-time on-line data. Trends of the readings are analyzed and conclusions are drawn on the basis of variations of the readings and rate of change of the sensors.

To test the developed expert system for functionality and to create rules dependent on state interrelationships, the short period longitudinal mode of a C-8 Buffalo aircraft was considered. The state variables in the system model are the pitch rate q and the angle-of-attack α which obey the differential equations (units deg-sec and deg)

$$\begin{bmatrix} \dot{q} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} -1.588 & -.562 \\ 1.000 & -.737 \end{bmatrix} \begin{bmatrix} q \\ \alpha \end{bmatrix} + \begin{bmatrix} -1.660 \\ -.005 \end{bmatrix} \delta_e \ ,$$

where δ_e is the elevator deflection input. The transfer functions from the input to each state can be found. For the pitch rate the transfer function is

$$\frac{\delta_e}{q} = G_q(s) = \frac{-1.66s - 1.2206}{s^2 + 2.325s + 1.732}$$

and the transfer function of the angle-of-attack is

$$\frac{\delta_e}{\alpha} = G_\alpha(s) = \frac{-.005s - 1.668}{s^2 + 2.325s + 1.732} \ .$$

As can be seen, both of these transfer functions are second order and it would be very difficult to create straight forward rules from them. By combining the two transfer functions and eliminating δ_e the transfer function between pitch rate and the angle-of-attack becomes

$$\frac{\alpha}{q} = G_{\alpha q}(s) = \frac{.005s + 1.668}{1.66s + 1.2206} \ ,$$

which is an easier transfer function to approximate. Since the expert system is determining rate of change of the sensors, consistency checks can be done with

$$\alpha \approx \frac{.005\dot{q} + 1.668q - 1.66\dot{\alpha}}{1.2206} .$$

This check can be simplified even farther if the expert system determines that one of the states is not varying, since at that time the time derivative of that state in zero.

Doing this type of analysis and model reduction, mathematical rules can be created based on expected results of the states. This serves as a consistency check on the interelationships of the states with each other and with the inputs.

This expert system is structured similarly to the proposed architecture in the previous section. There is a FORTRAN program which performs data acquisition and numerical computations. Specific values are passed into the expert sysetm which then go through the created rules to find all the possible solutions. The results and conclusions that the expert system derives are inserted into a database. This database can then be manipulated to act on the conclusions of the expert system. The system does not have interrupt or scheduling capabilities which will be important elements of the final design.

The capabilities of this system are limited due to the restricted nature of the way knowledge can be represented in the Exsys development environment. Exsys only allows the user to represent knowledge in the form of IF-THEN rules which is very restrictive when trying to detect a failure in a component of a system. Component characteristics can not be structured and represented in an concise rule due to the inherant dependancies on other facts. This is why frames and semantic nets will be used to structure system knowledge in the proposed expert system. Characteristics of a system which are static during a process can be represented well using semantic nets. This will include interdependancies of system components (i.e., sensors, actuators, etc.).

7.3 Results of Fault Scenarios

The expert system was run for the following four scenarios:

1. Error in pitch rate sensor,
2. Error in angle-of-attack sensor,
3. Pitch rate sensor failure, and

4. Normal operation.

The same elevator input was given to all the cases shown in figure 14.

The rules and the system output from the program are seen in Appendix A. This expert system could be considered a sub-module of a larger system where analysis can be done on an isolated area to minimize the search through the rule base. The output of this system would be sent to a level higher in the expert system so that conclusions from the module can be compared with conlusions from other modules. This allows for an efficient and organized search and decision making process.

Case 1: Error in Pitch Rate Sensor

The expected and actual values of the pitch rate are shown in figure 15. The pitch rate data was changed corresponding to a drop in gain at larger magnitudes. The expert system was able to detect an inconsistency in the pitch rate and the angle-of-attack. Since there was not a rule about the relation of pitch rate and the elevator deflection with the given alpha rate, the expert system was not able to state an inconsistency between them.

Case 2: Error in Angle-of-Attack Sensor

The angle-of-attack sensor failure data is shown in figure 16. The expert system was able to detect an inconsistency in the pitch rate and angle-of-attack. Again, since this was a small model, the system does not have a relationship between the angle-of-attack and the elevator deflection.

Case 3: Pitch Rate Sensor Failure

This scenario of a failure is shown in figure 17. The pitch rate sensor dies at a certain point, and remains at the failure value. The expert system is able to detect an inconsistency between the pitch rate and angle-of-attack along with an inconsistency between the pitch rate and the elevator deflection. With these two conclusions, it is possible for the expert system to deduce that the pitch rate sensor is most likely the problem since both diagnostic conclusions involve the pitch rate.

Case 4: Normal Operation

This case was run and included to show that the expert system is capable of realizing when the operation is running functionally.

One of the desired outcomes of the final expert system design, is the capability of multi-rate analysis on different parameters. In this demonstration, analysis

on the validity of the sensor reading can be done for different sample rates, but the interrelationships of the states assumes sampled data value at the same time interval.

7.4 Demonstration Summary

The demonstration expert system created on Exsys has served as a verifying feasibility example. The basic elements of the proposed interface structure for the expert system was incorporated in this demonstration. The issues involved for a real-time expert system have not been addressed in the demonstration due to the simple nature of the software package used.

8 Summary and Conclusions

The results of the Phase I effort were as follows:

1. Determination of key features required to address the issues of real-time on-line monitoring and self-repairing expert systems.
2. Design of architecture of the embedded expert system required to incorporate and utilize features for real-time use.
3. Integration technique required to interface the embedded expert system with an aircraft flight control system.
4. Creation of a simple expert system to demonstrate the techniques and the feasibility of the proposed structure to accomplish real-time on-line fault detection, isolation and self-repairing control.

This research has analyzed the fundamental understanding of the real-time requirements to interface expert system technology with algorithm control computation. In particular, bringing intelligent fault detection and isolation techniques into a high performance aircraft flight control system environment.

The goal for the end of Phase II is a real-time expert system environment prototype which can be used for developing and integrating fault detection techniques with reconfigurable control systems. The Phase I effort has determined the key issues that will be addressed and developed in Phase II of this program.

Although the prototype that will be developed by the end of Phase II will be specifically tailored for NASA, the technology developed through the course of the research will apply to many different areas requiring real-time expert system tools.

A Rules and Expert System Output

Subject:
Detecting faulty sensor readings in a dynamically changing environment.

Author:
Arvind K. Agarwal
Integrated Systems, Incorporated (C)

Starting text:
This expert system will detect the faulty readings in a sensor based on the sensor inputs, the maximum and minimum value allowed on the sensor, the sampling time used to get the values, and the maximum allowed deviation of the sensors between two consecutive readings. This expert system will accept three sensor readings. Sensor one will be the pitch rate of the aircraft, sensor two is the angle of attack, and sensor three is the elevator deflection.

Ending text:
The results are as follows:

Uses all applicable rules in data derivations.

Calls the external program st

RULES:

```
----------------------------------------
C
C   This rule checks the maximum limit on the third reading
C   of sensor three. In Exsys a wild card can not be used to
C   lock a variable name in a rule, so each reading along with
C   a high and low check must be performed.
C

RULE NUMBER: 1

IF:
       [SENSOR33] > [MAXVAL3]


THEN:
       Sensor three is out of limit - Probability=1


----------------------------------------
C
C   This rule checks the third reading of sensor two to
C   check if it is within the lower bounds. If it is, a
C   remark will be made to the output.
C

RULE NUMBER: 2

IF:
       [SENSOR32] < [MINVAL3]


THEN:
       Sensor three is out of limit - Probability=1


----------------------------------------
C
C   This rule checks to see if the second reading of the
C   of sensor three is within the maximum range.
C

RULE NUMBER: 3

IF:
       [SENSOR32] > [MAXVAL3]


THEN:
       Sensor three is out of limit - Probability=1


----------------------------------------
```

```
C
C   This rule checks if the first reading of sensor three
C   is within the minimum range.
C

RULE NUMBER: 4

IF:
       [SENSOR31] < [MINVAL3]


THEN:
       Sensor three is out of limit - Probability=1


----------------------------------------
C
C   This rule checks to see if the first reading on sensor three
C   is within the maximum limit.
C

RULE NUMBER: 5

IF:
       [SENSOR31] > [MAXVAL3]


THEN:
       Sensor three is out of limit - Probability=1


----------------------------------------
C
C   This rule checks if the first reading of sensor one
C   is within the maximum limits.
C

RULE NUMBER: 6

IF:
       [SENSOR11] > [MAXVAL1]


THEN:
       Sensor one out of limit - Probability=1


----------------------------------------
C
C   This rule checks if the first reading of sensor one
C   is within the minimum value.
C

RULE NUMBER: 7

IF:
       [SENSOR11] < [MINVAL1]
```

```
THEN:
       Sensor one out of limit - Probability=1


----------------------------------------
C
C   This rule checks if the second reading of sensor one
C   is within the maximum limit.
C

RULE NUMBER: 8

IF:
       [SENSOR12] > [MAXVAL1]


THEN:
       Sensor one out of limit - Probability=1


----------------------------------------
C
C   This rule check if the second reading of sensor one
C   is within the minimum limits.
C

RULE NUMBER: 9

IF:
       [SENSOR12] < [MINVAL1]


THEN:
       Sensor one out of limit - Probability=1


----------------------------------------
C
C   This rule checks if the third reading on  sensor one
C   is within the maximum limits.
C

RULE NUMBER: 10

IF:
       [SENSOR13] > [MAXVAL1]


THEN:
       Sensor one out of limit - Probability=1


----------------------------------------
C
C   This rule checks if the third reading of sensor one
C   is within the minimum range limit.
C
```

```
RULE NUMBER: 11

IF:
        [SENSOR13] < [MINVAL1]

THEN:
        Sensor one out of limit - Probability=1

----------------------------------------
C
C    This rule check to see if sensor three has varied
C    in the two stages. If not, then it notifies the system
C    of possibly a bad or stuck sensor.
C

RULE NUMBER: 12

IF:
        [SENSOR33] - [SENSOR31] = 0.
  and   Sensor three in stage one is stuck

THEN:
        Sensor three is not changing - Probability=1

ELSE:
        Sensor three in stage two is not stuck

----------------------------------------
C
C    This rule checks to see if the combined change in
C    sensor three in the two stages has been larger then
C    allowed. If the change in stage two was satisfactory
C    but not in stage one, then there is possibly high noise
C    in sensor three.

RULE NUMBER: 13

IF:
        ABS(([SENSOR33] - [SENSOR31])/2*[DELT3]) < [MAXVAR3]
  and   Sensor three in stage one is changing too quickly

THEN:
        Possible high noise environment in sensor three - Probability=1

ELSE:
        Sensor three in stage two is not noise

----------------------------------------
C
```

```
C   This rule checks the other case of rule 13. If the change
C   in both stages is to high, then possibly the sensor or the
C   state the sensor represents is running away.
C

RULE NUMBER: 14

IF:
       ABS(([SENSOR33] - [SENSOR31])/2*[DELT3]) > [MAXVAR3]
  and  Sensor three in stage one is changing too quickly


THEN:
       Sensor three changing faster then allowed - Probability=1


ELSE:
       Sensor three in stage two is not changing

----------------------------------------
C
C   This rule checks the variations in sensor two. If no change
C   is seen in both stages, then possibly the sensor is dead.
C

RULE NUMBER: 15

IF:
       [SENSOR23] - [SENSOR21] = 0
  and  Sensor two in stage one is stuck


THEN:
       Sensor two is not changing - Probability=1


ELSE:
       Sensor two in stage two is not stuck


----------------------------------------
C
C   This rule checks the rate of change of sensor two. If
C   one of the stages has large variations but the other
C   does not, then there is possibly noise in the sensor.
C

RULE NUMBER: 16

IF:
       ABS(([SENSOR23] - [SENSOR21])/(2*[DELT2])) < [MAXVAR2]
  and  Sensor two in stage one is changing too quickly


THEN:
       Possibly high noise environment in sensor two. - Probability=1
```

```
ELSE:
       Sensor two in stage two is not noise


----------------------------------------
C
C   This rule is the case of rule 16 where if both stages
C   have large variations, then possibly the sensor or the
C   state the sensor represents is running away.
C

RULE NUMBER: 17

IF:
       ABS(([SENSOR23] - [SENSOR21])/(2*[DELT2])) > [MAXVAR2]
  and  Sensor two in stage one is changing too quickly


THEN:
       Sensor two changing faster then allowed - Probability=1


ELSE:
       Sensor two in stage two is not changing


----------------------------------------
C
C   This rule checks to see if sensor one has changed in
C   the two stages. If it has not, then the sensor could
C   be stuck or dead.
C

RULE NUMBER: 18

IF:
       [SENSOR13] - [SENSOR11] =  0.
  and  Sensor one in stage one is stuck


THEN:
       Sensor one is not changing - Probability=1


ELSE:
       sensor one in stage two is not stuck


----------------------------------------
C
C   This rule checks the possibility of high noise environment
C   in sensor one.
C

RULE NUMBER: 19

IF:
```

```
        ABS(([SENSOR13] - [SENSOR11])/2*[DELT1]) < [MAXVAR1]
  and   Sensor one in stage one is changing too quickly

THEN:
        Possibly high noise environment in sensor one. - Probability=1

ELSE:
        sensor one in stage two is not noise

----------------------------------------
C
C    This rule checks to see about the possibility of a
C    run away on sensor one.
C

RULE NUMBER: 20

IF:
        ABS(([SENSOR13] - [SENSOR11])/2*[DELT1]) > [MAXVAR1]
  and   Sensor one in stage one is changing too quickly

THEN:
        Sensor one changing faster then allowed. - Probability=1

ELSE:
        sensor one in stage two is not changing

----------------------------------------
C
C    This rule checks to see if all the conditions have
C    been met to say that sensor one is operating
C    functionally.
C

RULE NUMBER: 21

IF:
        sensor one in stage two is not stuck
  and   sensor one in stage two is not noise
  and   sensor one in stage two is not changing

THEN:
        Sensor one functional operation. - Probability=1

----------------------------------------
C
C    The rule check to see if the third reading of sensor
C    two is within the minimum value limit.
C
```

```
RULE NUMBER: 22

IF:
       [SENSOR23] < [MINVAL2]


THEN:
       Sensor two out of limits. - Probability=1


----------------------------------------
C
C   This rule checks to see if the third reading of sensor
C   two is within the maximum value limit.
C

RULE NUMBER: 23

IF:
       [SENSOR23] > [MAXVAL2]


THEN:
       Sensor two out of limits. - Probability=1


----------------------------------------
C
C   This rule checks to see if the second reading on sensor
C   two is within the minimum value limit.
C

RULE NUMBER: 24

IF:
       [SENSOR22] < [MINVAL2]


THEN:
       Sensor two out of limits. - Probability=1


----------------------------------------
C
C   This rule checks to see if the second reading on sensor
C   two is within the maximum limit allowed.
C

RULE NUMBER: 25

IF:
       [SENSOR22] > [MAXVAL2]


THEN:
       Sensor two out of limits. - Probability=1
```

```
----------------------------------------
C
C   This rule checks to see if the first reading of sensor
C   two is within the minimum limit range.
C

RULE NUMBER: 26

IF:
       [SENSOR21] < [MINVAL2]


THEN:
       Sensor two out of limits. - Probability=1


----------------------------------------
C
C   This rule checks if the first reading on sensor two
C   is within the maximum limit range.
C

RULE NUMBER: 27

IF:
       [SENSOR21] > [MAXVAL2]


THEN:
       Sensor two out of limits. - Probability=1


----------------------------------------
C
C   This rule check to see if all the conditions are met
C   to say that sensor two is functionally operational.
C

RULE NUMBER: 28

IF:
       Sensor two in stage two is not stuck
  and  Sensor two in stage two is not noise
  and  Sensor two in stage two is not changing


THEN:
       Sensor two functional operation. - Probability=1


----------------------------------------
C
C   This rule checks to see if all the conditions are met
C   to say that sensor three is functionally  operational.
C

RULE NUMBER: 29
```

```
IF:
        Sensor three in stage two is not changing
  and   Sensor three in stage two is not noise
  and   Sensor three in stage two is not stuck


THEN:
        Sensor three functional operation. - Probability=1


----------------------------------------
C
C    This rule check the consistence between the dynamical
C    relationship between the angle-of-attack and the elevator
C    deflection when the rate of change of the pitch rate is zero.
C

RULE NUMBER: 30

IF:
        Sensor two is not changing = 1
  and   ABS([SENSOR22] - .0029*[VAR3] + .963*[SENSOR32]) > .03


THEN:
        Inconsistency in angle of attack and elevator deflection. -
        Probability=1


----------------------------------------
C
C    This rule checks the dynamical relationship between the
C    angle-of-attack and elevator deflection when the rate of
C    change of the pitch rate and the angle-of-attack is zero.
C

RULE NUMBER: 31

IF:
        Sensor one is not changing = 1
  and   Sensor two is not changing = 1
  and   ABS([SENSOR22] - [SENSOR32]) > .03


THEN:
        Inconsistency in angle of attack and elevator deflection. -
        Probability=1


----------------------------------------
C
C    This rule check the dynamical relationship between the
C    pitch rate and the angle-of-attack when the rate of
C    change of the pitch rate and the angle-of-attack is zero
C

RULE NUMBER: 32
```

```
IF:
        Sensor one is not changing = 1
  and   Sensor two is not changing = 1
  and   ABS([SENSOR12] - .75*[SENSOR22]) > .03


THEN:
        Inconsistency in pitch rate and angle of attack. - Probability=1

----------------------------------------
C
C   This rule checks the dynamical relation between the pitch
C   rate and the elevator deflection when the rate of change
C   of the pitch rate is zero.
C

RULE NUMBER: 33

IF:
        Sensor one is not changing = 1
  and   ABS([SENSOR12] + .958*[VAR3] + .7046*[SENSOR32]) > .03


THEN:
        Inconsistency in pitch rate and elevator deflection. - Probabilit

----------------------------------------
C
C   This rule checks the rate of variation in sensor one in
C   stage one.
C

RULE NUMBER: 34

IF:
        ABS(([SENSOR12] - [SENSOR11])/[DELT1]) > [MAXVAR1]


THEN:
        Sensor one in stage one is changing too quickly


ELSE:
        Sensor one in stage one is varying


----------------------------------------
C
C   This rule checks to see if sensor one in stage one
C   could possibly be stuck.
C

RULE NUMBER: 35

IF:
```

```
        ABS(([SENSOR12] - [SENSOR11])) = 0.

THEN:
        Sensor one in stage one is stuck

----------------------------------------
C
C   This rule checks the variation of sensor two in stage one
C   to ensure that it has changed less then allowed.
C

RULE NUMBER: 36

IF:
        ABS(([SENSOR22] - [SENSOR21])/[DELT2]) > [MAXVAR2]

THEN:
        Sensor two in stage one is changing too quickly

ELSE:
        Sensor two in stage one is varying

----------------------------------------
C
C   Thir rule checks to see if possibly sensor two could
C   be stuck in stage one.
C

RULE NUMBER: 37

IF:
        ABS(([SENSOR22] - [SENSOR21])) = 0

THEN:
        Sensor two in stage one is stuck

----------------------------------------
C
C   This rule checks the variation on sensor three in
C   stage one to see if it is within the allowed range.
C

RULE NUMBER: 38

IF:
        ABS(([SENSOR32] - [SENSOR31])/[DELT3]) > [MAXVAR3]

THEN:
        Sensor three in stage one is changing too quickly
```

```
ELSE:
       Sensor three in stage one is varying


----------------------------------------
C
C   This rule checks to see if sensor three has changed
C   in stage one.
C

RULE NUMBER: 39

IF:
       ABS(([SENSOR32] - [SENSOR31])) = 0.


THEN:
       Sensor three in stage one is stuck


----------------------------------------
C
C   This rule checks the number of iterations to see
C   when to stop.
C

RULE NUMBER: 40

IF:
       [POSITION] = [CHECK]


THEN:
       report(specify)
  and  stop


----------------------------------------
C
C   16 rules follow allowing the system to report to a
C   different system or different part of the same system
C   about the condition of the operation. Error handling
C   programs can be installed in this program to handle the
C   errors in different. manners.
C   This rule reports that the pitch rate sensor and the
C   angle-of-attack sensor are changing faster then allowed.
C

RULE NUMBER: 41

IF:
       Sensor one changing faster then allowed. = 1
  and  Sensor two changing faster then allowed = 1
  and  [POSITION] < [CHECK]


THEN:
```

```
        report(specify)

-----------------------------------------
C
C   This rule reports that both the pitch rate and the
C   angle-of-attack sensor might have high noise present.
C

RULE NUMBER: 42

IF:
        Possibly high noise environment in sensor one. = 1
  and   Possibly high noise environment in sensor two. = 1
  and   [POSITION] < [CHECK]


THEN:
        report(specify)


-----------------------------------------
C
C   This rule reports that the pitch rate sensor and the
C   angle-of-attack sensor are not changing, meaning they
C   might be dead of stuck.
C

RULE NUMBER: 43

IF:
        Sensor one is not changing = 1
  and   Sensor two is not changing = 1
  and   [POSITION] < [CHECK]


THEN:
        report(specify)


-----------------------------------------
C
C   This rule reports that both the pitch rate and the
C   angle-of-attack sensor are operating functionally.
C

RULE NUMBER: 44

IF:
        Sensor one functional operation. = 1
  and   Sensor two functional operation. = 1
  and   [POSITION] < [CHECK]


THEN:
        report(specify)
```

```
----------------------------------------
C
C   This rule reports that the pitch rate sensor is changing
C   faster then allowed and the angle-of-attack sensor could
C   be in a high noise environment.
C

RULE NUMBER: 45

IF:
       Sensor one changing faster then allowed. = 1
  and  Possibly high noise environment in sensor two. = 1
  and  [POSITION] < [CHECK]


THEN:
       report(specify)


----------------------------------------
C
C   This rule reports that the pitch rate sensor is changing
C   faster then allowed and the angle-of-attack sensor could
C   be stuck or dead.
C

RULE NUMBER: 46

IF:
       Sensor one changing faster then allowed. = 1
  and  Sensor two is not changing = 1
  and  [POSITION] < [CHECK]


THEN:
       report(specify)


----------------------------------------
C
C   This rule reports that the pitch rate sensor is changing
C   faster then allowed and the angle-of-attack sensor is operating
C   functionally.
C

RULE NUMBER: 47

IF:
       Sensor one changing faster then allowed. = 1
  and  Sensor two functional operation. = 1
  and  [POSITION] < [CHECK]


THEN:
       report(specify)
```

```
----------------------------------------
C
C   This rule reports that the pitch rate sensor has high noise
C   and that the angle-of-attack sensor is changing faster then
C   allowed.
C

RULE NUMBER: 48

IF:
        Possibly high noise environment in sensor one. = 1
  and   Sensor two changing faster then allowed = 1
  and   [POSITION] < [CHECK]


THEN:
        report(specify)


----------------------------------------
C
C   This rule reports that the pitch rate sensor has high
C   noise and that the angle-of-attack sensor is not changing
C

RULE NUMBER: 49

IF:
        Possibly high noise environment in sensor one. = 1
  and   Sensor two is not changing = 1
  and   [POSITION] < [CHECK]


THEN:
        report(specify)


----------------------------------------
C
C   This rule reports that the pitch rate sensor has high noise
C   and that the angle-of-attack sensor is operating functionally.
C

RULE NUMBER: 50

IF:
        Possibly high noise environment in sensor one. = 1
  and   Sensor two functional operation. = 1
  and   [POSITION] < [CHECK]


THEN:
        report(specify)


----------------------------------------
C
```

```
C   This rule reports that the pitch rate sensor is not changing
C   and that the variation on the angle-of-attack sensor is larger
C   then allowed.
C

RULE NUMBER: 51

IF:
       Sensor one is not changing = 1
  and  Sensor two changing faster then allowed = 1
  and  [POSITION] < [CHECK]


THEN:
       report(specify)


----------------------------------------
C
C   This rule reports that the pitch rate sensor is not changing
C   and that the angle-of-attack sensor has high noise.
C

RULE NUMBER: 52

IF:
       Sensor one is not changing = 1
  and  Possibly high noise environment in sensor two. = 1
  and  [POSITION] < [CHECK]


THEN:
       report(specify)


----------------------------------------
C
C   This rule reports that the pitch rate sensor is not changing
C   and that the angle-of-attack sensor is operating functionally.
C

RULE NUMBER: 53

IF:
       Sensor one is not changing = 1
  and  Sensor two functional operation. = 1
  and  [POSITION] < [CHECK]


THEN:
       report(specify)


----------------------------------------
C
C   This rule reports that the pitch rate sensor is operating
C   functionally and the angle-of-attack sensor is changing
```

```
C    faster then allowed.
C

RULE NUMBER: 54

IF:
        Sensor one functional operation. = 1
  and   Sensor two changing faster then allowed = 1
  and   [POSITION] < [CHECK]


THEN:
        report(specify)


----------------------------------------
C
C    This rule reports that the pitch rate sensor is operating
C    functionally and the angle-of-attack sensor is in a high
C    noise environment.
C

RULE NUMBER: 55

IF:
        Sensor one functional operation. = 1
  and   Possibly high noise environment in sensor two. = 1
  and   [POSITION] < [CHECK]


THEN:
        report(specify)


----------------------------------------
C
C    This rule reports that the pitch rate sensor is operating
C    functionally and that the angle-of-attack sensor is not
C    changing and could possibly be dead.
C

RULE NUMBER: 56

IF:
        Sensor one functional operation. = 1
  and   Sensor two is not changing = 1
  and   [POSITION] < [CHECK]


THEN:
        report(specify)
```

OUTPUT FOR CORRECT SENSOR VALUES

Time Tick	Diagnosis
	Sensor one is not changing: Probability=1
3	Sensor two is not changing: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one is not changing: Probability=1
4	Sensor two is not changing: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one is not changing: Probability=1
5	Sensor two is not changing: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one functional operation.: Probability=1
6	Sensor two is not changing: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
7	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
8	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
9	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
10	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
11	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
12	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
13	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
14	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
15	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1

```
                Sensor one functional operation.: Probability=1
16              Sensor two functional operation.: Probability=1
                Sensor three functional operation.: Probability=1

                Sensor one functional operation.: Probability=1
17              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1

                Sensor one functional operation.: Probability=1
18              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1

                Sensor one functional operation.: Probability=1
19              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1

                Sensor one functional operation.: Probability=1
20              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1
```

ERROR IN PITCH RATE SENSOR

Time Tick	Diagnosis
	Sensor one is not changing: Probability=1
3	Sensor two is not changing: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one is not changing: Probability=1
4	Sensor two is not changing: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one is not changing: Probability=1
5	Sensor two is not changing: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one functional operation.: Probability=1
6	Sensor two is not changing: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
7	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
8	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
9	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
10	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
11	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
12	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Inconsistency in pitch rate and angle of attack.: Probability=1
	Sensor one functional operation.: Probability=1
13	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Inconsistency in pitch rate and angle of attack.: Probability=1
	Sensor one functional operation.: Probability=1
14	Sensor two functional operation.: Probability=1

```
                Sensor three functional operation.: Probability=1
                Inconsistency in pitch rate and
                                angle of attack.: Probability=1

                Sensor one functional operation.: Probability=1
15              Sensor two functional operation.: Probability=1
                Sensor three functional operation.: Probability=1
                Inconsistency in pitch rate and
                                angle of attack.: Probability=1

                Sensor one functional operation.: Probability=1
16              Sensor two functional operation.: Probability=1
                Sensor three functional operation.: Probability=1

                Sensor one functional operation.: Probability=1
17              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1

                Sensor one functional operation.: Probability=1
18              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1

                Sensor one functional operation.: Probability=1
19              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1

                Sensor one functional operation.: Probability=1
20              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1
```

ERROR IN ANGLE-OF-ATTACK SENSOR

Time Tick	Diagnosis
	Sensor one is not changing: Probability=1
3	Sensor two functional operation.: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one is not changing: Probability=1
4	Sensor two functional operation.: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one is not changing: Probability=1
5	Sensor two functional operation.: Probability=1
	Sensor three is not changing: Probability=1
	Inconsistency in pitch rate and angle of attack.: Probability=1
	Sensor one functional operation.: Probability=1
6	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Inconsistency in pitch rate and angle of attack.: Probability=1
	Sensor one functional operation.: Probability=1
7	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
8	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Inconsistency in pitch rate and angle of attack.: Probability=1
	Sensor one functional operation.: Probability=1
9	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
10	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
11	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
12	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
13	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1

```
14          Sensor two functional operation.: Probability=1
            Sensor three functional operation.: Probability=1

            Sensor one functional operation.: Probability=1
15          Sensor two functional operation.: Probability=1
            Sensor three functional operation.: Probability=1

            Sensor one functional operation.: Probability=1
16          Sensor two functional operation.: Probability=1
            Sensor three functional operation.: Probability=1

            Sensor one functional operation.: Probability=1
17          Sensor two functional operation.: Probability=1
            Sensor three is not changing: Probability=1

            Sensor one functional operation.: Probability=1
18          Sensor two functional operation.: Probability=1
            Sensor three is not changing: Probability=1

            Sensor one functional operation.: Probability=1
19          Sensor two functional operation.: Probability=1
            Sensor three is not changing: Probability=1

            Sensor one functional operation.: Probability=1
20          Sensor two functional operation.: Probability=1
            Sensor three is not changing: Probability=1
```

PITCH RATE SENSOR DIES

Time Tick	Diagnosis
	Sensor one is not changing: Probability=1
3	Sensor two is not changing: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one is not changing: Probability=1
4	Sensor two is not changing: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one is not changing: Probability=1
5	Sensor two is not changing: Probability=1
	Sensor three is not changing: Probability=1
	Sensor one functional operation.: Probability=1
6	Sensor two is not changing: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
7	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
8	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
9	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
10	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
11	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one functional operation.: Probability=1
12	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Sensor one is not changing: Probability=1
13	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Inconsistency in pitch rate and elevator deflection.: Probability=1
	Sensor one is not changing: Probability=1
14	Sensor two functional operation.: Probability=1
	Sensor three functional operation.: Probability=1
	Inconsistency in pitch rate and

```
                                angle of attack.: Probability=1
                Inconsistency in pitch rate and
                                  elevator deflection.: Probability=1

                Sensor one is not changing: Probability=1
15              Sensor two functional operation.: Probability=1
                Sensor three functional operation.: Probability=1
                Inconsistency in pitch rate and
                                angle of attack.: Probability=1
                Inconsistency in pitch rate and
                                  elevator deflection.: Probability=1

                Sensor one is not changing: Probability=1
16              Sensor two functional operation.: Probability=1
                Sensor three functional operation.: Probability=1
                Inconsistency in pitch rate and
                                angle of attack.: Probability=1
                Inconsistency in pitch rate and
                                  elevator deflection.: Probability=1

                Sensor one is not changing: Probability=1
17              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1
                Inconsistency in pitch rate and
                                angle of attack.: Probability=1
                Inconsistency in pitch rate and
                                  elevator deflection.: Probability=1

                Sensor one is not changing: Probability=1
18              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1
                Inconsistency in pitch rate and
                                angle of attack.: Probability=1
                Inconsistency in pitch rate and
                                  elevator deflection.: Probability=1

                Sensor one is not changing: Probability=1
19              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1
                Inconsistency in pitch rate and
                                angle of attack.: Probability=1
                Inconsistency in pitch rate and
                                  elevator deflection.: Probability=1

                Sensor one is not changing: Probability=1
20              Sensor two functional operation.: Probability=1
                Sensor three is not changing: Probability=1
                Inconsistency in pitch rate and
                                angle of attack.: Probability=1
                Inconsistency in pitch rate and
                                  elevator deflection.: Probability=1
```

References

[1] M. Krishnamurthi, R.D. Mayer, and P.G. Friel, "Robust Fault Diagnosis Using Deep and Shallow Modeling Approaches," *Proceedings of ROMEXS '86*, Second Annual Workshop on Robotics and Expert systems, NASA/Johnson Space Center, June 4–6 1986.

[2] R.A. Walker, M.E. Badgett, R.L. Kosut, and S.C. Shah, "An Expert MIMO Control System Design Advisor," *AIAA Guidance and Control Conference*, Willamsburg, VA, August 16–18, 1986.

[3] N.K. Gupta, R.A. Walker, "Robust Fault Detection Techniques, " AIAA Guidance, and Control Conf., Seattle, WA, August 20–22 1984, pp. 758–760

[4] E.Y. Chow and A.S. Willsky,"Analytical Redundancy and the Design of Robust Failure Detection Systems," IEEE Trans. on Auto. Control, Vol. AC-29, No. 7, July 1984

[5] R.L. Kosut, R.A. Walker and S.C. Shah,"Robust Fault Detection, Isolation and Accommodation to Support Integrated Aircraft Control," AIAA Guidance and Control Conf., Gatlinburg, TN, August 15–17, 1983

[6] D.P. Looze, S.M. Krolewski, J.L. Weiss, J.S. Eterno, and S.W. Gully,"An Approach to Restructurable Control System Design," Proc. of the 23rd Conf. on Dec. and Control, Los Vegas, NV, December 1984

[7] D.A. Waterman,"A Guide to Expert Systems," Addison-Wesley Publishing Co., Menlo Park, 1986

[8] R.E. Korf,"Real-Time Heuristic Search:First Results," AAAI Sixth National Conf. on A.I., Seattle, WA, July 13–17 1987, pp. 133–138

[9] P.E. Hart, N.J. Nilsson, and B. Raphael,"A Formal Basis for the Heuristic Determination of Minimum Cost Path," IEEE Trans. on Systems Science and Cybernetics, SSC-4, No. 2, 1968, pp. 100–107

[10] R.E. Korf,"Depth-First Iterative Deepening:AN Optimal Admissible Tree Search," Artificial Intelligence, Vol. 27, No. 1, 1985, pp. 97–109

[11] H.A. Simon,"The Science of the Artificial," 2nd edition, M.I.T. Press, Cambridge, MA, 1981

[12] W.B. Gevarter,"The Nature and Evaluation of Commercial Expert System Building Tools," Computer, IEEE, May 1987, pp. 24–41

[13] A. Wolfe,"An Eaiser Way to Build a Real-Time Expert System, " Electronic Magazine, March 5, 1987

NASA National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
CR-179433		

4. Title and Subtitle

Automation Tools for Demonstration of Goal Directed and Self-Repairing Flight Control Systems

5. Report Date

August 1988

6. Performing Organization Code

7. Author(s)

A.K. Agarwal

8. Performing Organization Report No.

H-1498

10. Work Unit No.

RTOP 505-66-11

9. Performing Organization Name and Address

Integrated Systems, Inc.
2500 Mission College Boulevard
Santa Clara, CA 95054

11. Contract or Grant No.

NAS2-12588

13. Type of Report and Period Covered

Contractor Report—Final

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, D.C. 20546

14. Sponsoring Agency Code

15. Supplementary Notes

NASA Technical Monitor: Eugene L. Duke, NASA Ames Research Center, Dryden Flight Research Facility, Edwards, California 93523-5000.

16. Abstract

This report documents a phase I Small Business Innovative Research (SBIR) contract in the coupling of expert systems and control design and analysis techniques to provide a realizable self-repairing flight control system. Key features of such a flight control system are identified and a limited set of "rules" for a simple aircraft model are presented.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Artificial intelligence Expert systems Flight controls Real time systems	Unclassified—Unlimited Subject category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	75	A04

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

End of Document