# Combining Expert System and Analytical Redundancy Concepts for Fault-Tolerant Flight Control

David A. Handelman* and Robert F. Stengel†
*Princeton University, Princeton, New Jersey*

**A technique for rule-based fault-tolerant flight control is presented. The objective is to define methods for designing control systems capable of accommodating a wide range of aircraft failures, including sensor, control, and structural failures. A software architecture that integrates quantitative analytical redundancy techniques and heuristic expert system concepts for the purpose of in-flight, real-time fault tolerance is described. The resultant controller uses a rule-based expert system approach to transform the problem of failure accommodation task scheduling and selection into a problem of search. Control system performance under sensor and control failures using linear discrete-time deterministic simulations of a tandem-rotor helicopter's dynamics is demonstrated. It is found that the rule-based control technique enhances existing redundancy management systems, providing smooth integration of symbolic and numeric computation, a search-based decision-making mechanism, straightforward system organization and debugging, an incremental growth capability, and inherent parallelism for computational speed.**

## Introduction

FAULT tolerance has always been an important aspect of aircraft design. The reduced static stability and increased maneuverability of modern aircraft compound the problem by shortening the amount of time available to detect, identify, and adjust for a given component failure. At the same time, efforts by both the pilot and flight control system to provide the required level of fault tolerance are hampered by a limited amount of sensor and control resources. Dictated by cost considerations, these resource limits translate into limits on the amount of information and capability available for failure accommodation. Flight control research intended to overcome this handicap can draw on advances in redundancy management, artificial intelligence (AI) theory, and digital computer technology.

Significant advances in aircraft fault tolerance have been accomplished through the use of redundancy.[1-8] Voting schemes using direct hardware redundancy among similar sensors, and functional hardware redundancy among related sensors, provide information useful for failure detection, identification, and reconfiguration. More complex schemes also may use some degree of analytical redundancy, which allows the comparison of a sensor measurement with its predicted value. The predicted value is generated with a mathematical model of the physical process being measured. These methods provide an effective, but limited, failure accommodation capability.

Mathematical models that approximately describe pilot behavior exist, but the type of problem-solving actually used by pilots when faced with a failure is far from quantitative.[9-11] This suggests the need for incorporating a humanlike reasoning ability into a fault-tolerant flight control system. Many concepts of AI theory strive to emulate the human thought process by computer and, therefore, could be considered for such an application. Relevant work includes that on qualitative reasoning about physical systems[12] and on expert systems.[13-14] Designed to encode and exercise the knowledge of experts, noteworthy expert systems are those dealing with diagnostics and maintenance[15-17] and those combining qualitative and quantitative reasoning.[18,19] The apparent success of expert system concepts to provide a limited humanlike decision-making capability within a well-defined problem domain gives strong support to their use in fault-tolerant flight control.

The objective of this research is to develop AI-based design techniques for flight control systems that can detect, identify, and accommodate a very large number of potential failures.[20,21] The desired result is not only a technique for fault-tolerant flight control but also an evaluation of how the chosen methodologies assist in the design and implementation of such a complex real-time control system. In particular, the issue of integrated symbolic and numeric computation is addressed. High integration separates this research effort from others in AI-based control; symbolic processing assumes an integral, rather than supervisory, role in control system operation.

The purpose of the controller is to reduce or eliminate abnormal behavior introduced by an unexpected change in the aircraft configuration. These changes may take the form of failures in sensors, controls, and structural components of the aircraft. An overview of a possible solution to the problem of fault-tolerant flight control is represented in Fig. 1. The job of failure accommodation is broken down into five main tasks: executive control, failure detection, failure diagnosis, failure model estimation, and reconfiguration. The executive control task provides continual dynamic state estimation, feedback control calculations, and synchronization of the remaining tasks. The failure detection task monitors aircraft behavior and detects significant abnormalities. Failure diagnosis finds a set of probable causes and effects of the problem, while the failure model estimation task generates a mathematical model of the aircraft dynamics considered to reflect changes due to the failure. Finally, the reconfiguration task determines what action should be taken to correct the situation. In the Rule-Based Flight Control System (RBFCS) presented subsequently, a combination of analytical redundancy and expert system concepts is used to accomplish and coordinate these failure accommodation tasks.

## Rule-Based Approach to Task Scheduling and Selection

The Rule-Based Flight Control System uses *search* to provide fault tolerance. The difference between procedural failure accommodation and that based on search is the way in

Presented as Paper 86-2092 at the AIAA Guidance, Navigation and Control Conference, Williamsburg, VA, Aug. 18–20, 1986; received Oct. 27, 1986; revision received June 30, 1987. 

*Graduate Student, Department of Mechanical and Aerospace Engineering. Student Member AIAA.

†Professor, Department of Mechanical and Aerospace Engineering. Associate Fellow AIAA.

which the required control system actions are scheduled and selected. Unlike a step-by-step procedure, the search technique outlined subsequently attempts to emulate the human decision-making process, performing only those actions deemed necessary and sufficient to keep the aircraft flying safely.

The RBFCS employs a search technique similar to that of the MYCIN expert system.[22] The search process involves a *knowledge base* and an *inference engine*. The knowledge base contains information about the world. The inference engine operates on the knowledge base attempting to infer additional information from that which already exists. Specifically, the knowledge base contains factual information in the form of *parameters* and procedural information in the form of *rules*. Examples of factual information relevant to failure are, "The sensor measurements are reasonable" and "The estimator prediction error is small." In this case, the parameters are SENSOR MEASUREMENTS with possible values REASONABLE and UNREASONABLE, and ESTIMATOR PREDICTION ERROR with possible values SMALL and LARGE. Examples of procedural information are, "If the sensor measurements are reasonable and the estimator prediction error is small, then abnormal behavior is not detected," and, "If the sensor measurements are unreasonable or the estimator prediction error is large, then abnormal behavior is detected." Each rule contains a premise and an action. When a rule is tested, its action is performed only if its premise holds.

Figure 2 shows a graphical representation of this sample knowledge base. In this figure, rectangles represent parameters. Slots within a given rectangle contain all values that the corresponding parameter can acquire. With arcs between parameters representing rules, the resultant "and/or" graph can be used to trace the logic path taken by the search process.

The issue of search enters when additional information is to be inferred from the present state of the knowledge base. Before a search begins, all parameters that do not have an initially known value are assumed unknown. This process is called knowledge-base initialization. During a search, parameters acquire values through the testing of rules, although the specific rules tested and parameters set depend on the type of search used.

Two types of search are used in the RBFCS: goal-directed (backward-chaining) and data-driven (forward-chaining). The purpose of a goal-directed search is to find a value for a specified parameter. Using the example given earlier, a goal-directed search would involve the question, "Is abnormal behavior detected?" or, more specifically, the instruction (goal), "Determine the value of ABNORMAL BEHAVIOR DETECTED." When confronted with this task, the inference engine first tests to see if ABNORMAL BEHAVIOR DETECTED already has a value and, if so, returns the value. If not, it tries to infer a value by testing rules that contain ABNORMAL BEHAVIOR DETECTED in their action.

Notice that, when a rule is tested, its premise may require knowledge of parameter values that are as yet unknown. In an attempt to find the value for ABNORMAL BEHAVIOR DETECTED, for example, the inference engine will test the rule "If the sensor measurements are reasonable and the estimator prediction error is small, then abnormal behavior is not detected" (Fig. 2). The premise of this rule is

[(SENSOR MEASUREMENTS = REASONABLE)
AND
(ESTIMATOR PREDICTION ERROR = SMALL)]

The rule requires knowledge of the value of SENSOR MEASUREMENTS and possibly that of ESTIMATOR PREDICTION ERROR if the former is set to REASONABLE. The inference engine will be called recursively to find values for these parameters in the attempt to test the original premise. Hence, the logic flow of a goal-directed search proceeds down the graph of rules, returning back up when parameter values required in tested premises have been determined.

A data-driven search determines the effect of a given parameter having a certain value by testing all rules that contain that parameter in their premise. The search recursively chains forward on any other parameters that acquire a value through the actions of the rules tested. The logic flow thus proceeds up the graph of rules until the full effect of the original parameter value is felt, then returns back down to the starting point. For example, a data-driven search on "Sensor measurements are reasonable" asks the question, "What is the effect of SENSOR MEASUREMENTS = REASONABLE?" The first rule is tested, setting the value of ABNORMAL BEHAVIOR DETECTED to FALSE if ESTIMATOR PREDICTION ERROR = SMALL (Fig. 2). This invokes the question, "What, then, is the effect of ABNORMAL BEHAVIOR DETECTED = TRUE?" The process continues until no more rules remain to be tested.

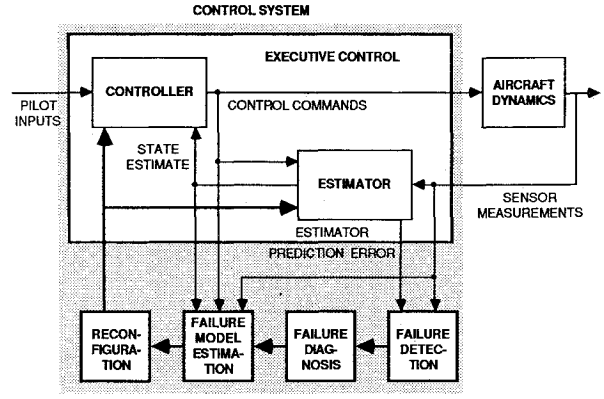



**Fig. 1 Organization of the Rule-Based Flight Control System.**

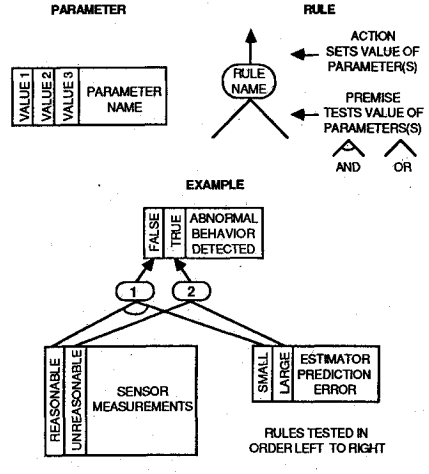



**Fig. 2 Knowledge-base graphical representation.**

The fault-tolerant control system uses search to schedule and select actions that would be done conventionally by a step-by-step algorithm. This scheduling and selection is performed indirectly during the search, occurring through the actions of rules tested in the attempt to determine parameter values. As an example of scheduling, consider some of the rules contained in the executive control knowledge base, depicted graphically in Fig. 3.

*Rule-E05:*
IF estimator and controller are ready
AND dynamic states are estimated
THEN calculate control perturbation commands.

*Rule-E06:*
IF estimator and controller are ready
AND sensor measurements are received
THEN estimate dynamic states.

*Rule-E07:*
IF this rule is tested
THEN receive sensor measurements.

A goal-directed search performed at each control system sampling interval begins with the instruction, "Determine the value of EXECUTIVE CONTROL SEARCH COMPLETED." Backward chaining eventually leads to the instruction, "Determine the value of PERTURBATION COMMANDS CALCULATED." Rule-E05 is tested in an attempt to find a value for this parameter. The premise of this rule requires knowledge of the values of two parameters. Assuming that ESTIMATOR AND CONTROLLER READY is set to TRUE, Rule-E06 is tested to determine a value for DYNAMIC STATES ESTIMATED. This rule, in turn, requires knowledge of the value of SENSOR MEASUREMENTS RECEIVED, thereby invoking the testing of Rule-E07. The premise of Rule-E07 is always true (intentionally), thus forcing the action to be performed immediately whenever the rule is tested. In this case, the sensor measurements are taken (an input/output operation), and the parameter SENSOR MEASUREMENTS RECEIVED is set to TRUE. With a positive premise, Rule-E06 then estimates the dynamic states and sets DYNAMIC STATES ESTIMATED to TRUE. In turn, Rule-E05 finally calculates the control perturbation commands and sets PERTURBATION COMMANDS CALCULATED to TRUE.

The side effect of determining the value of a top-level parameter is to perform actions in a specified and necessary order. Similarly, the process of selecting among alternatives, or decision making, can be accomplished using the same search technique. As an example, consider some of the rules of the reconfiguration knowledge base, as shown in Fig. 4.

As with the executive control task, a goal-directed reconfiguration search eventually leads to the instruction "Determine the value of FAILED SENSORS ACCOMMODATED." The premise of Rule-R04 requires knowledge of BIASED SENSORS ACCOMMODATED, which can be set by either Rule-R07 or Rule-R08. Because it appears first in the knowledge base, Rule-R07 is tested. This, in turn, causes Rule-R17 to first calculate the number of biased sensors, then set NUMBER OF BIASED SENSORS to the result. If no sensors are biased, the premise of Rule-R07 holds and the corresponding action sets the value of BIASED SENSORS ACCOMMODATED to TRUE. Otherwise, Rule-R07 fails and the inference engine tests Rule-R08. As a result, Rule-R18 is tested, and the estimator bias factors are calculated. Similar decisions are made throughout the reconfiguration search, with time-consuming procedures being executed only when needed through the actions of relevant rules.

Characterization of the numerical procedures as "building blocks" points out the role to be played by rules within the control system. Rules are meant to augment numerical procedures derived through analytical means. In its simplest form, rule-based search conveniently implements deeply nested IF–THEN–ELSE clauses. The resultant ability to perform complex conditional branching in an organized manner provides an enhanced decision-making mechanism with which to manipulate (initialize, synchronize, schedule, select, execute, etc.) analytically derived numerical procedures.

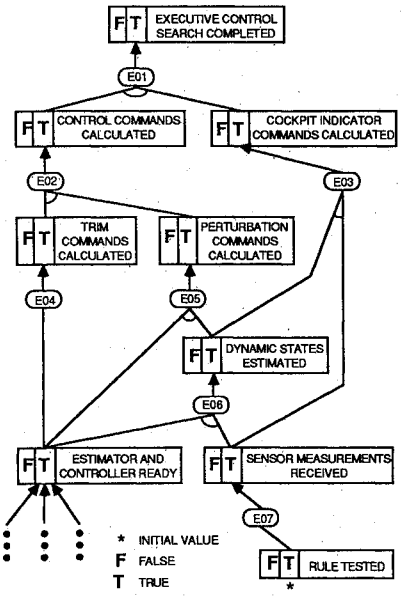



**Fig. 3 Sample of executive control knowledge base.**

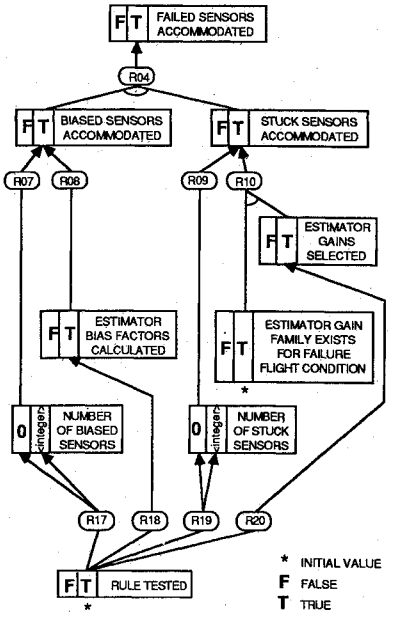



**Fig. 4 Sample of reconfiguration knowledge base.**

## Control System Knowledge-Base Contents

The RBFCS knowledge base contains 209 parameters, 247 rules relating these parameters, and numerous quantitative algorithms residing as procedure calls in the actions of rules. If applied to this knowledge base, an inference engine would conduct a search intent on offsetting the effects of failure on aircraft behavior. As implemented, however, instead of performing a single, time-consuming search on the entire knowledge base, the inference engine is applied to separate pieces of the knowledge base grouped according to the tasks outlined previously: executive control, failure detection, failure diagnosis, failure model estimation, and reconfiguration. Each task is thus a separate entity with its own inference engine and knowledge base, conducting its own search and interacting with other tasks through shared data structures. In this way, separate portions of the overall failure accommodation task may be completed concurrently, decreasing the required computation time.

Described subsequently are the distinct rule groups of the knowledge base and the significant algorithms invoked as a result of these rules. A summary of the knowledge base contents is shown in Table 1.

### Executive Control

The executive control task forms the core of the fault-tolerant control system, providing state estimation and feedback control calculations. Consequently, the executive control search is conducted repeatedly, synchronous with the sensor measurement sampling interval in order to assure timely completion of these critical functions. The remaining task searches are called upon as needed by the executive control search. These other searches are intended to take place on a processor independent of that conducting the executive control search, allowing them to run asynchronous with the sampling interval.

The executive control search estimates the dynamic states, calculates the new control commands, and performs an act of message passing to coordinate the remaining task searches as needed. It first sets a flag requesting failure detection. It then immediately checks another flag to see if a failure is detected, and, if so, requests failure diagnosis. If the failure is diagnosed, it requests failure model estimation. If the failure model is estimated, reconfiguration is requested. Finally, if reconfiguration is determined to be needed, the executive control task adjusts its estimator and controller parameters accordingly, then returns to its initial chores. Notice that the executive control search does not wait for other task searches to finish; it merely sets and tests flags in order to coordinate them. This passing of flags allows the executive control search always to end within one measurement sampling interval as required.

The controller is designed to regulate aircraft motion about a constant flight condition. Assuming linearity within a neighborhood of this nominal operating point, a state-space mathematical model approximates the aircraft dynamics. Based on this no-failure model, a Kalman filter[23] is used for state estimation, and a linear-quadratic regulator[24] is used for feedback control calculations. Figure 5 contains a block diagram representing the resultant linear equations, assuming discrete-time stochastic dynamics. In these equations, the vector $\boldsymbol{x}$ contains the aircraft state variables, $\boldsymbol{u}$ and $\boldsymbol{z}$ correspond to control positions and sensor measurements, respectively, and $\boldsymbol{u}_0$ and $\boldsymbol{z}_0$ correspond to their nominal values. Vectors and matrices with a caret ($\hat{}$) denote assumed, or estimated, values. Finally, $\boldsymbol{w}$ and $\boldsymbol{v}$ represent uncorrelated Gaussian white noise sequences, and $\mathbf{K}$ and $\mathbf{C}$ are steady-state estimator and regulator gain matrices.

A failure is assumed to change significantly the mathematical model representing the actual aircraft dynamics, prompting the need for estimator and regulator reconfiguration. Referring to Fig. 5, the vectors and matrices used to represent failure operations are $\mathbf{A}_z$, $\boldsymbol{z}_b$, $\mathbf{A}_u$, and $\boldsymbol{u}_b$. In the unfailed state, $\mathbf{A}_z$ and $\mathbf{A}_u$ are identity matrices, and all the elements of vectors $\boldsymbol{z}_b$ and $\boldsymbol{u}_b$ are zero. A sensor bias failure is represented by an abrupt change in the appropriate element of $\boldsymbol{z}_b$, while a control bias failure involves a similar change in $\boldsymbol{u}_b$. A stuck sensor involves zeroing out an element of $\mathbf{A}_z$ in addition to changing $\boldsymbol{z}_b$. Similar changes to $\mathbf{A}_u$ and $\boldsymbol{u}_b$ represent a stuck control. Reconfiguration implies changing the appropriate elements of the estimator and controller to offset the effect of such a failure.

### Failure Detection

The failure detection task monitors aircraft behavior, looking for abnormalities indicative of a failure requiring remedial action. Using a measure of behavior abnormality based on analytical redundancy contained in the innovations of the executive control nominal Kalman filter, an algorithm called the normalized innovations monitor (NIM) triggers failure detection. This algorithm, similar in intent to the sequential probability ratio test,[25] provides an effective scalar measure of the success of the Kalman filter to predict aircraft behavior. Because an inability to predict accurately this behavior is indicative of a significant difference between the assumed and actual aircraft configurations, a failure is declared when the scalar measure exceeds a specified threshold.

When a failure is detected, additional indicators of behavior abnormality are calculated to assist the failure diagnosis task. These indicators are average and root-mean-square (rms) values of certain signals over a window of time extending back from the point of failure detection. The signals chosen are the average and rms values of the sensor measurements, the average and rms values of the measurement innovations, and the rms values of the normalized innovations.

### Failure Diagnosis

The failure diagnosis task is broken down into two phases: failure-origin hypothesis generation and failure-model hypothesis generation. Failure-origin hypotheses represent the aircraft elements determined most likely to have caused the problem. Failure-model hypotheses are mathematical models corresponding to aircraft configuration changes assumed due to the hypothesized failure origins.

The failure-origin hypotheses are obtained via a signal dependency search.[26] Rules of the knowledge base relate aircraft elements using a relative measure of influence based on sensitivities of the aircraft equations of motion to worst-case failure modes. Triggered by the behavior abnormality indicators provided by failure detection, a data-driven search on these rules results in a list of elements considered most likely to have failed.

**Table 1 Control system knowledge-base contents**

| Task | Parameters | Rules | Major subtasks |
|---|---|---|---|
| Executive control | 18 | 23 | Kalman filter and linear-quadratic regulator |
| Failure detection | 9 | 15 | Normalized innovations monitor |
| Failure diagnosis | 135 | 147 | Signal dependency search |
| Failure model estimation | 15 | 23 | Multiple-model algorithm |
| Reconfiguration | 32 | 39 | Weighted left pseudoinverse |

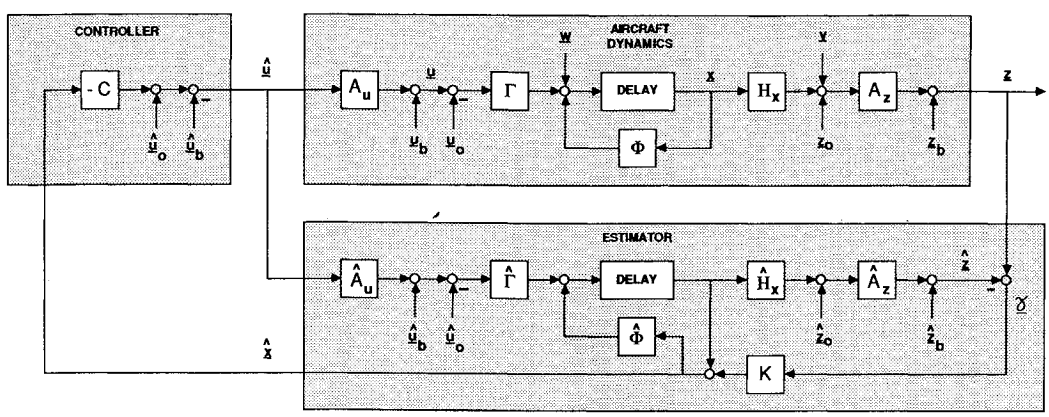



**Fig. 5 Linear model of aircraft dynamics, estimator, and controller with failure-model vectors and matrices.**

Rules responsible for generating failure-model hypotheses attempt to recognize patterns in the behavior abnormality indicators. These patterns were derived through batch simulation runs covering a wide range of failures. For example, the following rule was derived for a tandem-rotor helicopter and is part of the failure diagnosis knowledge base.

*Rule-I41:*

IF control failure candidates are determined
AND forward collective pitch control
is a candidate
AND the largest element of the normalized
innovations rms is pitch rate
AND the ratio of pitch rate (rad/s)
to vertical velocity (m/s)
normalized innovations rms is
within 10% of 6.01
THEN hypothesize forward collective pitch control
stuck at
sign (pitch rate innovations average) ×
[(35.8 × pitch rate innovations rms) − 1.48]
cm at
(−2.85 × pitch rate innovations rms) + 0.890 s
prior to failure detection

Similar rules exist for all failure modes of all aircraft elements capable of failure.

#### Failure Model Estimation

The failure model estimation task selects from among the failure-model hypotheses the model that best represents the postfailure aircraft dynamics. It does this by choosing the model that most accurately predicts aircraft behavior over a certain interval of time. The quantitative procedure used for model selection is the multiple-model algorithm (MMA).[25]

The MMA contains a bank of Kalman filters based on dynamic models that correspond to the failure-model hypotheses. Using a buffer containing a history of control commands, sensor measurements, and state estimates, each filter is initialized with the state estimate obtained at the failure model's estimated failure time. An iterative search then implements the MMA. Each iteration begins with the calculation (for each filter) of the conditional probability that its model is correct given the control commands and observed measurements. Bayes' rule is then used to compute the probability of each filter's model being correct given the existence of all other filters. The failure model estimation search oversees operation of the algorithm, ensuring that the number of MMA iterations falls between specified minimum required and maximum allowed values. If the maximum allowed number of MMA iterations is reached, the hypothesis with the highest probability is chosen as the estimated failure model. If, during an earlier iteration, the highest probability far exceeds all others, the corresponding model is chosen, and the iterative search terminates.

#### Reconfiguration

Given an estimate of the mathematical model representing the failed aircraft, the reconfiguration task suggests remedial changes to the estimator and regulator. This suggestion takes the form of a reconfigured set of Kalman filter and linear-quadratic-regulator gains. Bias failures are accommodated easily through artificial unbiasing of the suspect sensor or control signal, as represented by $z_b$ and $u_b$ in Fig. 5. Stuck sensors and controls involve retrieving from memory gains that were calculated off-line assuming that the faulty element never existed. Relying on system redundancy, these gains effectively eliminate dependence of the estimator or regulator on the stuck sensor or control. Forces and moments produced by a control stuck in an off-trim position also must be reduced as much as possible. The reconfiguration search uses heuristics and a weighted left pseudoinverse operation to generate adjustments to the remaining operative controls in the attempt to restore trim.

## Description of the Control System

The RBFCS is configured to provide simulated failure accommodation for a U.S. Army CH-47 tandem-rotor helicopter traveling at 80 knots airspeed and sea-level altitude. Assuming a sampling rate of 10/s, a fully coupled eighth-order linear model is used to represent helicopter dynamics.[27] Inputs to the control system are body-axis sensor measurements: longitudinal velocity ($u$), lateral velocity ($v$), vertical velocity ($w$), roll rate ($p$), pitch rate ($q$), yaw rate ($r$), pitch angle ($\theta$), and roll angle ($\phi$). Outputs are commands to two actuators of each rotor: forward cyclic pitch, forward collective pitch, aft cyclic pitch, and aft collective pitch. The knowledge base presently contains rules and associated algorithms capable of handling a significant bias or stuck failure in any one of these sensors or controls.

A combination of LISP and Pascal is used to develop and implement the control system software. Considered a requirement for realization of a flight-capable implementation of the RBFCS, the use of a high-order language with fast execution speed guided the selection of Pascal as the language for run-time source code. The symbolic manipulation capabilities of LISP are used not only to provide a convenient prototyping search environment but also to translate a high-level knowledge-base description into a low-level one required by the Pascal environment.

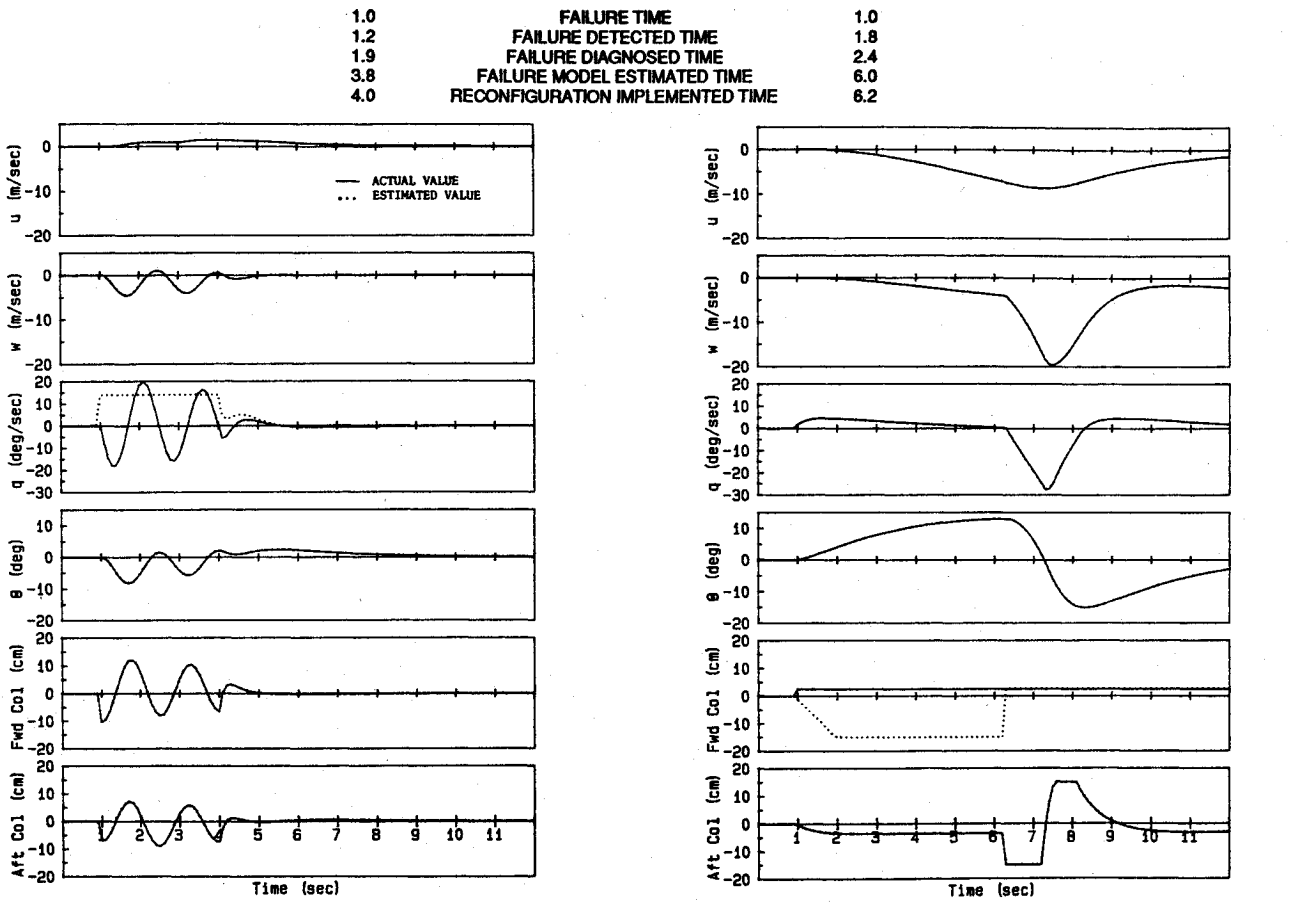
In order to carry out goal-directed and data-driven searches, the inference engine must have, for each parameter of the knowledge base, a list of names of rules containing that parameter in their action and a list of names of rules containing that parameter in their premise. These lists are obtained off-line from high-level rule descriptions with a LISP program. The same program translates this and other knowledge-base information into the numerous constant, type, and variable declarations needed by the Pascal compiler. Because rule premises are implemented in Pascal as functions and rule actions as procedures, the LISP program also creates Pascal source code files of rules. In this way, the capabilities of LISP are used for off-line control system preparation, while the speed advantages of Pascal are used for on-line realization.

In addition to increasing search speed dramatically, knowledge-base translation enables the integration of search with numeric processing by allowing rules and procedures to communicate through common data structures. Although rules and parameters are developed in a symbolic LISP environment, numeric procedures are coded in Pascal. Because parameters are implemented as Pascal variables following knowledge-base translation, their values can be tested and modified by procedures. In this manner, the result of a procedure can be "returned" to its calling rule if necessary. Similarly, rules are free to access variables other than parameters, such as the elements of matrices used routinely by procedures.

Real-time RBFCS performance has been obtained using knowledge-base translation and a multiprocessor hardware architecture. Described subsequently is the emulated operation of a five-processor control system (one processor per task) performing simulated failure accommodation during regulation of a linear discrete-time deterministic CH-47 dynamic model (Fig. 5). Each processor is assumed to contain an 8-MHz 80286 central processing unit and a 5.33-MHz 80287 math coprocessor. Documented elsewhere is real-time concurrent RBFCS operation using a three-processor implementation.[28]

## Example of Control System Performance

The effects of a stuck pitch rate sensor on aircraft longitudinal control commands and the resultant motion are shown in Fig. 6a. At the start of the simulation, the NIM of the failure detection task search is given time to "warm up." At $t = 1.0$ s, the pitch rate sensor becomes stuck at 14 deg/s from its nominal reading, and the aircraft begins to pitch. The abrupt jump in pitch rate normalized innovations triggers failure detection




a) Pitch rate sensor stuck 14 deg/s from nominal

b) Forward collective pitch control stuck 2.5 cm from nominal (controls saturate at $\pm 15$ cm)

**Fig. 6 Longitudinal state and control time histories for simulated failures.**

**Table 2 Failure diagnosis and failure model estimation search results for simulated failures ($z_q$ = pitch rate sensor, $\delta_{fc}$ = forward collective pitch control, $\delta_{ac}$ = aft collective pitch control)**

| Failure | Failure-origin hypotheses | Failure-model hypotheses | Fairlure-model hypothesis probability per multiple-model algorithm iteration | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| $z_q$ stuck 14 deg/s at $t = 1.0$ | $z_q$ | No failure at $t = 1.2$ | 0.333 | 0.001 | 0.000 | 0.000 | 0.000 | 0.000 | — | — |
| | $\delta_{fc}$ | $z_q$ biased 14 deg/s at $t = 1.0$ | 0.333 | 0.500 | 0.019 | 0.001 | 0.000 | 0.000 | — | — |
| | $\delta_{ac}$ | $z_q$ stuck 14 deg/s at $t = 1.0$ | 0.333 | 0.500 | 0.981 | 0.999 | 1.000 | 1.000 | — | — |
| $\delta_{fc}$ stuck 2.5 cm at $t = 1.0$ | $z_q$ | No failure at $t = 1.8$ | 0.200 | 0.004 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | $\delta_{fc}$ | $z_q$ stuck 9.2 deg/s at $t = 1.5$ | 0.200 | 0.082 | 0.022 | 0.003 | 0.000 | 0.000 | 0.000 | 0.000 |
| | $\delta_{ac}$ | $\delta_{fc}$ biased 11 cm at $t = 1.4$ | 0.200 | 0.273 | 0.287 | 0.296 | 0.291 | 0.256 | 0.185 | 0.096 |
| | | $\delta_{fc}$ stuck 2.6 cm at $t = 1.0$ | 0.200 | 0.325 | 0.357 | 0.368 | 0.379 | 0.404 | 0.450 | 0.507 |
| | | $\delta_{fc}$ stuck 3.7 cm at $t = 1.3$ | 0.200 | 0.315 | 0.334 | 0.332 | 0.330 | 0.340 | 0.366 | 0.398 |

within two sampling intervals at $t = 1.2$ s. A 0.7-s failure diagnosis search then occurs, producing a list of three failure-origin hypotheses (out of a possible 12: eight sensors and four controls), followed by a set of three failure-model hypotheses as shown in Table 2. The failure model estimation task selects the best model within 1.9 s using the minimum required number of MMA iterations, prompting a request for reconfiguration instructions at $t = 3.8$ s. As the estimated failure model contains a disabled pitch rate sensor, a 0.2-s reconfiguration search retrieves from memory a set of estimator gains calculated off-line with the assumption that no pitch rate sensor ever existed. This eliminates dependence of the state estimates on direct measurement of pitch rate. After the executive control task replaces its estimator gains, aircraft motion finally settles back to its nominal operating condition.

Figure 6b and Table 2 show the effects of the forward collective pitch control actuator being stuck 2.5 cm from its trim position at $t = 1.0$ s. Failure detection, mainly due to deviations in the pitch rate and vertical velocity normalized innovations, occurs within 0.8 s of the failure time. A 0.6-s failure diagnosis search generates three failure-origin and five failure-model hypotheses. The higher number of model hypotheses compared to the stuck sensor simulation described earlier means that each iteration of the MMA takes longer to complete. In addition, because no model probability far exceeds all others during the iterations, the estimated failure model is chosen only after the maximum allowed number of MMA iterations has been reached (Table 2). This results in a 3.6-s failure model estimation search. The reconfiguration search then supplies not only new controller gains, but also a new aft collective pitch control trim position to help offset the effect of a failure-induced pitching moment. Finally, 5.2 s after occurrence of the failure, the executive control task implements the required reconfiguration.

The preceding simulations demonstrate the ability of the rule-based control system to accommodate certain types of

failures given the assumption of regulation of a deterministic dynamic system about a constant nominal trajectory. The significance of this result can be seen in the role such a control system can play in relation to existing systems. The controller presently considers sensors and control actuators to be failure-origin candidates. In the real world, however, a problem with a sensor or actuator often can be traced back to something upon which it depends, such as an electrical or hydraulic support system. It is precisely this type of problem that prior concepts, using additional sensors and possibly some level of functional and analytical redundancy, are effective at handling. Unfortunately, this effectiveness is greatly diminished when one or more failures corrupt the complement of sensors used to provide this nominal level of performance. Because the RBFCS performs failure accommodation using as little direct and functional hardware redundancy as possible, its techniques can be used to provide a level of fault tolerance otherwise degraded as a result of these initial failures. In this way, the control system is intended to augment state-of-the-art systems either by providing a higher level of performance given an existing amount of hardware, or an equivalent level of performance with a reduction in hardware.

## Conclusions

A software architecture based on concepts of artificial intelligence theory has been applied to the problem of aircraft fault tolerance. Using goal-directed and data-driven search techniques, the fault-tolerant flight control system schedules and selects failure accommodation tasks conventionally coordinated by step-by-step algorithms. Rules are used to encode common sense relations and expert advice on specific situations and to manipulate quantitative mathematical procedures. Failure accomodation occurs as a side effect of search through this knowledge base of rules.

The indirect nature of control system task manipulation is not an inherent limitation of the software architecture; it simply means that the system designer must adjust to the change in program control flow. When matched against desired control system characteristics, the rule-based approach provides many advantages including a straightforward system organization, an incremental growth capability, and inherent parallelism for computational speed. In addition, debugging is greatly facilitated. A completed search implies that a conclusion was drawn from a supposedly logical progression of inferred facts. Programming errors can be found by following the search process until an illogical step is made. Most importantly, however, the software architecture provides for the efficient coupling of symbolic and numeric computation. Commonality of data structures between rules and procedures, resulting from LISP to Pascal knowledge-base translation, allows rule-based search to become an integral, rather than supervisory, part of control system operation.

The Rule-Based Flight Control System helps solve a restricted portion of the failure accommodation problem. While simulating regulation of the aircraft state about a constant nominal flight condition, the controller presently can accommodate an abrupt and significant bias or stuck failure in an aircraft sensor or control. Prospects for enhancing the capability of the control system, i.e., accommodation of more failure modes (including structural failures) under varying flight conditions, appear promising due to the beneficial characteristics of the chosen rule-based technique.

## Acknowledgment

This project was sponsored by the U.S. Army Research Office under Contract DAAG29-84-K-0048.

## References

[1]*Fault Tolerance Design and Redundancy Management Techniques*, AGARD-LS-109, Neuilly sur Seine, France, Oct. 1980.

[2]Montoya, R., Howell, W., Bundick, W., Ostroff, A., Hueschen, R., and Belcastro, C., eds., "Restructurable Controls," NASA CP-2277, Sept. 1982.

[3]Rubertus, D., "Self-Repairing Flight Control Systems Overview," *Proceedings of the 1983 National Aerospace Electronics Conference*, May 1983, pp. 1280–1286.

[4]Friedland, B., "Maximum Likelihood Failure Detection of Aircraft Flight Control Sensors," *Journal of Guidance, Control, and Dynamics*, Vol. 5, Sept.–Oct. 1982, pp. 498–503.

[5]Chow, E. and Willsky, A., "Analytical Redundancy and the Design of Robust Failure Detection Systems," *IEEE Transactions on Automatic Control*, Vol. AC-29, July 1984, pp. 603–614.

[6]Walker, B., Gai, E., and Desai, M., "A Fault-Tolerant Approach to Helicopter Swashplate Control," *Journal of Guidance, Control, and Dynamics*, Vol. 8, Jan.–Feb. 1985, pp. 62–70.

[7]Gelderloos, H. and Young, D., "Redundancy Management of Shuttle Flight Control Rate Gyroscopes and Accelerometers," *Proceedings of the 1982 American Control Conference*, IEEE, Piscataway, NJ, June 1982, pp. 808–811.

[8]*Active Control Systems—Review, Evaluation and Projections*, AGARD-CP-384, Toronto, Canada, Oct. 1984.

[9]van de Graaff, R. and Wewerinke, P., "Theoretical and Experimental Analysis of Pilot Failure Detection Behavior During Various Automatic Approach Conditions," National Aerospace Laboratory, Amsterdam, NLR-MP-83025-U, April 1983.

[10]Rockwell, T., Giffin, W., and Romer, D., "Combining Destination Diversion Decisions and Critical In-Flight Event Diagnosis in Computer Aided Testing of Pilots," *Proceedings of the Second Symposium on Aviation Psychology*, Ohio State Univ., Columbus, OH, April 1983, pp. 343–351.

[11]Johannsen, G. and Rouse, W., "Studies of Planning Behavior of Aircraft Pilots in Normal, Abnormal, and Emergency Situations," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 3, May–June 1983, pp. 267–278.

[12]Bobrow, D. and Hayes, P., eds., "Special Volume on Qualitative Reasoning About Physical Systems," *Artificial Intelligence*, Vol. 24, No. 1–3, Dec. 1984.

[13]Hayes-Roth, F., Waterman, D., and Lenat, D., *Building Expert Systems*, Addison-Wesley, Reading, MA, 1983.

[14]Buchanan, B. and Shortliffe, E., *Rule-Based Expert Systems*, Addison-Wesley, Reading, MA, 1984.

[15]"Artificial Intelligence in Maintenance," *Proceedings of the Joint Services Workshop*, AFHRL-TR-84-25, Denver Research Inst., Denver, CO, June 1984.

[16]Ames, K., "A Relational Approach to the Development of Expert Diagnostic Systems," NASA TM-86288, Oct. 1984.

[17]Fink, P., Lusth, J., and Duran, J., "A General Expert System Design for Diagnostic Problem Solving," *IEEE Workshop on Principles of Knowledge-Based Systems*, IEEE Computer Society Press, Silver Spring, MD, 1984, pp. 45–52.

[18]de Kleer, J., "Qualitative and Quantitative Reasoning in Classical Mechanics," *Artificial Intelligence: An MIT Perspective*, Vol. 1, MIT Press, Cambridge, MA, 1979, pp. 11–30.

[19]*AAAI Workshop on Coupling Symbolic and Numerical Computing in Expert Systems*, Boeing Computer Services Co., Bellevue, WA, Aug. 1985.

[20]Stengel, R., "Artificial Intelligence Theory and Reconfigurable Control Systems," Dept. of Mechanical and Aerospace Engineering, Princeton Univ., Princeton, NJ, Rept. 1664, June 1984.

[21]Handelman, D., "An Application of Artificial Intelligence Theory to Reconfigurable Flight Control," Dept. of Mechanical and Aerospace Engineering, Princeton Univ., Princeton, NJ, Rept. 1665, June 1984.

[22]Davis, R., Buchanan, B., and Shortliffe, E., "Production Rules as a Representation for a Knowledge-Based Consultation Program," *Artificial Intelligence*, Vol. 8, No. 1, Feb. 1977, pp. 15–45.

[23]Gelb, A., *Applied Optimal Estimation*, MIT Press, Cambridge, MA, 1974.

[24]Stengel, R., *Stochastic Optimal Control: Theory and Application*, Wiley, New York, 1986.

[25]Willsky, A., "Failure Detection in Dynamic Systems," *Fault Tolerance Design and Redundancy Management Techniques*, AGARD-LS-109, Neuilly sur seine, France, Oct. 1980, pp. 2.1–2.14.

[26]Handelman, D. and Stengel, R., "Combining Quantitative and Qualitative Reasoning in Aircraft Failure Diagnosis," *Proceedings of the 1985 AIAA Guidance, Navigation, and Control Conference*, AIAA, New York, 1985, pp. 366–375.

[27]Ostroff, A., Downing, D., and Rood, W., "A Technique Using a Nonlinear Helicopter Model for Determining Trims and Derivatives," NASA TN-D-8159, May 1976.

[28]Handelman, D. and Stengel, R., "An Architecture for Real-Time Rule-Based Control," *Proceedings of the 1987 American Control Conference*, IEEE, Piscataway, NJ, June 1987, pp. 1636–1642.