Intelligent Failure-Tolerant Control

Robert F. Stengel*

Princeton University
Department of Mechanical and Aerospace Engineering
Princeton, NJ 08544

Abstract

An overview of failure-tolerant control is presented, beginning with robust control, progressing through parallel and analytical redundancy, and ending with rule-based systems and artificial neural networks. By design or implementation, failure-tolerant control systems are "intelligent" systems. All failure-tolerant systems require some degree of robustness to protect against catastrophic failure; failure tolerance often can be improved by adaptivity in decision-making and control, as well as by redundancy in measurement and actuation. Reliability, maintainability, and survivability can be enhanced by failure tolerance, although each objective poses different goals for control system design. Artificial intelligence concepts are helpful for integrating and codifying failure-tolerant control systems, not as alternatives but as adjuncts to conventional design methods.

Introduction

Many devices depend on automatic control for satisfactory operation, and while assuring stability and performance with all components functioning properly remains the primary design goal, there is increasing need for controlled systems to continue operating acceptably following failures in either the system to be controlled (the plant) or in the control system itself.[1] A distinction should be made between system *failures*, which occur when components break or misbehave, and system *faults*, which include improper design as well. Our attention is directed at the former, as improper design is a separate issue.

Failure-tolerant control systems can be characterized as robust, reconfigurable, or some combination of the two. A well-designed feedback controller typically reduces the plant's output sensitivity to measurement errors and disturbance inputs; if the plant is lightly damped or unstable, it provides closed-loop stability as well. It is designed assuming some nominal physical structure for the plant, expressed by a mathematical model and a set of parameters. A controlled system that retains satisfactory performance in the presence of variations from this model without changes in the control system's structure or parameters is said to be *robust*. The degree of failure that can be accommodated by a fixed control structure is more restricted than that of a variable control structure. If the structure or parameters can be altered following system failure, the control system is *reconfigurable*.

In the latter case, the control system detects, identifies, and isolates failures, and it modifies control laws to maintain acceptable performance. A system that is failure-tolerant through reconfiguration is both adaptive and redundant. It is *adaptive* in its ability to adjust to off-nominal behavior, as occurs from loss or degradation of sensors, actuators, and power supplies, damage to signal and power transmission channels, or unexpected alteration of the plant's characteristics. It is *redundant* in its ability to overcome lost capabilities with remaining resources. Redundancy can be provided by similar parallel channels for measurement and control, or it may result from flexible logic that synthesizes missing measurements or control forces using operable sensors and actuators, effectively invoking dissimilar parallel channels. A reconfigurable control system must be robust enough to preclude controlled system failure while adaptation is taking place.

While there is much debate as to what constitutes true "machine intelligence," it can be argued that adaptivity and redundancy are attributes of intelligence and, in the same light, that feedback control makes use of information in an intelligent fashion. The issue is not that adaptive, feedback controllers pass the seminal Turing test [1] or possess "consciousness" [2]. It is that they exhibit the "ability involved in calculating, reasoning, perceiving relationships and analogies, learning quickly, storing and retrieving information, ... classifying, generalizing, and adjusting to new situations," [3] at least in a symbolic or quantitative sense. To the extent that symbols and instructions reflect knowledge and decisions, a failure-tolerant, feedback control system can be called intelligent, and that context is adopted here.

Controlled Systems

Attention is focused on the control of continuous-time dynamic systems (or plants) whose motions can be represented by integrals of nonlinear ordinary differential equations,

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t),\mathbf{u}(t),\mathbf{w}(t),\mathbf{p}] \qquad (1$$

where $\mathbf{x}(t)$ is the n-dimensional state, $\mathbf{u}(t)$ is the m-dimensional control, $\mathbf{w}(t)$ is an s-dimensional disturbance, and $\mathbf{p}$ is an l-vector of parameters. The state is observed through the measurement r-vector,

$$\mathbf{z}(t) = \mathbf{h}[\mathbf{x}(t),\mathbf{u}(t),\mathbf{w}(t),\mathbf{n}(t),\mathbf{p}] \qquad (2$$

where $\mathbf{n}(t)$ is an r-dimensional measurement-error vector. Along a nominal trajectory specified by $\mathbf{x}_0(t)$, $\mathbf{u}_0(t)$, $\mathbf{w}_0(t)$, and $\mathbf{n}_0(t)$ for t in (t_0, t_f), perturbations of the state and observation vectors are governed approximately by linear, time-varying equations,

---

* Professor. Senior Member, IEEE.

[1] For the purposes of this paper, the *plant* is defined as a dynamic system containing components that impart distinctive physical properties like mass, inertia, elasticity, forces, and moments. The plant's motion (position and velocity) must be controlled for satisfactory operation. The *control system* is an assemblage of additional components -- motion sensors, force and moment actuators, and computers -- that provide this service. A *controlled system* is a plant plus its control system.

Presented at the *5th IEEE International Symposium on Intelligent Control*, Philadelphia, PA, Sept 1990.

$$\Delta\dot{x}(t) = F(t)\Delta x(t) + G(t)\Delta u(t) + L(t)\Delta w(t) \quad (3$$

$$\Delta z(t) = H_x(t)\Delta x(t) + H_u(t)\Delta u(t) + H_w(t)\Delta w(t) + n(t) \quad (4$$

F, G, L, H_x, H_u, and H_w are conformable Jacobian matrices expressing sensitivities to the perturbation variables. At discrete instants of time, t_k, t_{k+1}, and so on, the state and measurement perturbations can be approximated by

$$\Delta x_{k+1} = \Phi_k \Delta x_k + \Gamma_k \Delta u_k + \Lambda_k \Delta w_k \quad (5$$

$$\Delta z_{k+1} = H_{x_k} \Delta x_k + H_{u_k} \Delta u_k + H_{w_k} \Delta w_k + n_k \quad (6$$

where the subscript "k" indicates evaluation at t_k. Here, Φ, Γ, and Λ have the same dimensions as F, G, and L and are derived from the system's state transition properties (e.g., [4]). These models provide a foundation for the remaining discussion.

Control logic for the nonlinear plant (eq. 1 and 2) typically takes the form of a *dynamic compensator*,

$$\Delta u_k = - C_k \xi_k \quad (7$$

$$\xi_{k+1} = \Psi_k \xi_k + \Theta_k \Delta u_k + K_k[z_k - h(\hat{x}_k, u_k)] \quad (8$$

$$\xi_k \triangleq [\Delta\hat{x}_k^T \ \chi_k^T]^T \quad (9$$

$$u_k = u_{o_k} + \Delta u_k \quad (10$$

$$\hat{x}_k \triangleq x_{o_k} + \Delta\hat{x}_k \quad (11$$

This linear, time-varying structure exemplifies estimation and control functions for discussion purposes, but more complex structures -- particularly nonlinear ones -- may be employed. It is equivalent to a feedback control law (eq. 7) operating on the internal state estimate $\Delta\hat{x}$ contained in the $(n + k)$-dimensional ξ_k (eq. 8). χ_k is a k-vector of compensation components, such as integrals of state elements. The control and estimation gains, C_k and K_k, are selected to provide satisfactory nominal response and may vary in time. Ψ_k and Θ_k normally represent nominal values of Φ_k and Γ_k plus integrating (i.e., accumulating) or filtering operations associated with χ_k. The desired state and corresponding control for the nonlinear plant, x_{o_k} and u_{o_k}, enter as in eq. 10 and 11.

Figure 1 represents an idealized controlled system, with disturbance and noise inputs not shown. While the figure identifies the elements of nominal control system design, it provides little insight about control system components, all of which may fail. Tangible components are needed for measurement and actuation (Fig. 2), the control logic described by eq. 7 to 11 is executed in a computer, these components are enabled by a power supply, and the power supply also is subject to failure. An ancillary issue is that sensors and actuators -- themselves physical systems -- have scale factors, biases, and dynamic characteristics to be considered during failure detection and identification. The simplest means of doing this is to incorporate these characteristics in the plant model (eq. 1 and 2), with estimation and control logic modified accordingly.

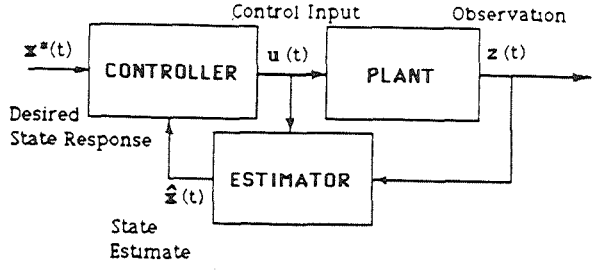

Figure 1. Idealization of a controlled system.

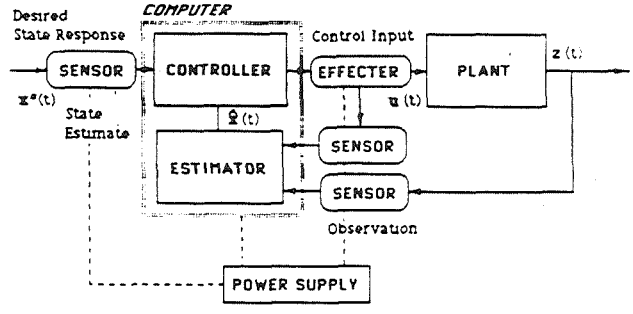

Figure 2. Components of a controlled system.

Objectives and Issues for Failure-Tolerant Control

Failure tolerance may be called upon to improve system reliability, maintainability, and survivability. The requirements for failure tolerance are different in these three cases. *Reliability* deals with the ability to complete a task satisfactorily and with the period of time over which that ability is retained. A control system that allows normal completion of tasks after component failure improves reliability. *Maintainability* concerns the need for repair and the ease with which repairs can be made, with no premium placed on performance. Failure tolerance could increase time between maintenance actions and allow the use of simpler repair procedures. *Survivability* relates to the likelihood of conducting an operation safely (without danger to human operators or the controlled system), whether or not the task is completed. Degraded performance following failure might be permitted, as long as the system can be brought to an acceptable state of rest.

Improving the reliability of individual components clearly helps in all three categories; however, it does not follow that what aids one objective aids another. For example, replacing a single string of control system components by three parallel strings of identical components (plus selection or averaging logic) may improve reliability, but it also increases the likelihood of component failures, degrading maintainability. Conversely, redundancy within line-replaceable units (LRUs) could improve maintainability if it allows LRUs to be changed less often. Adding a separate string of less-capable components may improve survivability without improving reliability while decreasing maintainability.

The principal categories of failure are plant alterations, actuator and sensor failures, computer failure, and power sup-

ply/transmission failure. Actuators, sensors, and other analog components are subject to many failure types, some of which may be subtle but nonetheless damaging: parameter variation, abrupt or random bias shift, abrupt or random scale factor shift, change in saturation limits, drift, open circuit, hardover (or stuck), and noise. Digital computer hardware failures have entirely different characteristics, but it can be argued that they are never subtle, as internal clock rates are high and the loss of coherent output is obvious [5]. Computer software does not fail *per se*, but it is susceptible to programming faults that may surface unexpectedly and that may be hard to detect. Multiple failures can occur, particularly as a consequence of physical damage, and they may be intermittent; hence, reconfiguration logic must do more than just accommodate isolated failures. While not strictly system failures, operator blunders and power transients may produce system states that require prompt response.

Many factors must be considered in designing failure-tolerant controls, including: allowable performance degradation in the failed state, criticality and likelihood of the failure, urgency of response to failure, tradeoffs between correctness and speed of response, normal range of system uncertainty, disturbance environment, component reliability vs. redundancy, maintenance goals (mean-time-between failures, mean-time-to-failure, mean-time-to-repair, maintenance-hours/operation-hours, etc.), size and cost of LRUs, system architecture, limits of manual intervention, and life-cycle costs. Assessing each of these factors requires detailed knowledge of the plant and its control objectives.

Robust Control

Controlled system *robustness* is the ability to maintain satisfactory stability and performance in the presence of parameter variations, which could be due to component failures in either the plant or the control system. All practical controlled systems must possess some degree of robustness against operational parameter variations. Maintaining stability with component failures is a particular challenge when the plant is open-loop-unstable, as control-system failure may mean that the system becomes partially "open-loop." Alternatively, a plant alteration (e.g., the breaking of a stabilizing spring or the loss of an aircraft's stabilizing surface) may force an ordinarily stable system to become unstable. In either case, reconfiguration may offer the only recourse for stable control. It also is possible for an open-loop-stable plant to be destabilized by a feedback controller with failed control loops [6]. This lack of robustness is most likely to occur in high-gain controllers, where open- and closed-loop dynamics are substantially different; robustness recovery typically requires lowering the control gains in systematic fashion [4,6,7]. The inherent stability margins of certain algebraic control laws (e.g., the linear-quadratic (LQ) regulator [4,8-10]) may become vanishingly small when dynamic compensation (e.g., the estimator in a linear-quadratic-Gaussian (LQG) regulator) is added [11]. Restoring the robustness to that of the LQ regulator typically requires increasing estimator gains using the loop-transfer-recovery method [4,12].

Subjective judgments have to be made in assessing the need for robustness and in establishing corresponding control system design criteria, as there is an inevitable tradeoff between robustness and nominal system performance [13]. The designer must know the normal operating ranges and distributions of parameter variations, as well as the specifications for system operability with failed components, else the final design may afford too little robustness for possible parameter variations or too much robustness for satisfactory nominal performance. Robustness traditionally has been assessed deterministically [14]; it is an inherent part of the classical design of single-input/single-output systems, and there are multi-input/multi-output equivalents based on singular-value analysis of various frequency-domain matrices [e.g., 4,10,12,15]. The most critical difficulty in applying these techniques is relating singular-value bounds on return-difference and inverse-return-difference matrices to real parameter variations in the controlled system.

There is increasing interest in statistical alternatives that make full use of knowledge about potential system variations and that work directly with real parameter variations. The *probability of instability* was introduced in [16] and is further described in [17,18]. This method determines the *stochastic robustness* of a linear, time-invariant system by the probability distributions of closed-loop eigenvalues, given the statistics of the variable parameters in the controlled system's dynamic model. The probability that any of these eigenvalues have positive real parts is the scalar measure of robustness, a figure of merit to be minimized by control system design. Extensions to the analysis of performance robustness and of nonlinear, time-varying systems are direct. This approach provides logical connections to reliability analysis of control systems, discussed below.

It is easy to pose unreachable or irrelevant goals for control robustness. Problems that must be addressed in robust control system design include: retaining controllability and observability following component failure, achieving satisfactory off-design performance (including steady-state and tracking response as well as stability), minimizing compromises to on-design performance, and relating robustness criteria to real component failures.

Parallel Redundancy

In principle, tolerance to control system failures can be improved if two or more strings of sensors, actuators, and computers, each separately capable of satisfactory control, are implemented in parallel (Fig. 3). A *voting* scheme is used for redundancy management, comparing control signals to detect and overcome failures. With two identical channels, a comparator can determine whether or not control signals are identical; hence, it can detect a failure but cannot identify which string has failed. Using three identical channels, the control signal with the middle value can be selected (or voted), assuring that a single failed channel never controls the plant. A 2-channel system is considered *fail-safe* because the presence of a failure can be determined, but it is left to additional in-line (or "built-in test") logic to select the unfailed channel for control. The 3-channel system is *fail-operational*, as the task can be completed following a single failure. Systems with four identical control channels are called "fail-op/fail-op" because they can tolerate two failures and still yield nominal performance. In any voting system, it remains for additional logic to declare unselected channels failed. Given the vectorial nature of control, this declaration may be equivocal, as middle values of control-vector elements can be drawn from different strings.

Of course, the voting logic itself has some probability of failure, and a *single-point failure* of a voting component could be catastrophic. Consequently, it may be preferable to let each channel remain independent through the application of control force, letting *force averaging* mediate failures. If control outputs are averaged, small variations among the parallel channels tend to cancel, and the net output is smooth; however, a runaway failure can bias the net signal away from its desired value. Voting and isolation of failed channels then can be carried out as an auxiliary process whose own failure would not disable the entire system. Once a failed channel has been disengaged, the total available

control force is reduced, changing the performance characteristics of the controlled system.

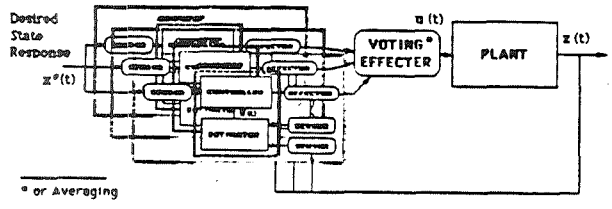

Figure 3. A triply redundant controlled system.

For perfect output voting of M identical parallel channels each with N serial components, the failure probability P_f of the overall control system is,

$$P_f = \prod_{j=1}^{M}\left[1 - \prod_{i=1}^{N} e^{-\lambda_i t}\right] \approx [(\lambda_s + \lambda_c + \lambda_a)(t_f - t_0)]^M \triangleq (1 - R)^M \qquad (12$$

Sensor, computer, and actuator failure rates[1] are λ_s, λ_c, and λ_a (assumed to be small and uncorrelated), $(t_f - t_0)$ is the mission duration, and R is the single-string reliability [19,20]. If the components can be *cross-strapped* perfectly (i.e., if a failed component from one string can be connected to an unfailed string), the overall probability of failure is reduced to

$$P_f = 1 - \prod_{j=1}^{M}\left[1 - \prod_{i=1}^{N}(1 - e^{-\lambda_i t})\right] \approx (\lambda_s{}^M + \lambda_c{}^M + \lambda_a{}^M)(t_f - t_0)^M \qquad (13$$

Unfortunately, failures cannot be detected perfectly, and cross-strapping itself is subject to failure. The probability of detecting, isolating, and recovering from a failure -- called *coverage* -- is a more meaningful measure than P_f. For a 3-channel control system with output voting alone, the coverage C [21], or net reliability, is

$$C = R^3 + 3R^2(1 - R)P_{r_1} + 3(1 - R)^2 R P_{r_2} \qquad (14$$

where P_{r_1} is the probability of recovering from the first failure and P_{r_2} is the probability of recovering from a second failure. These probabilities are not necessarily the same, as different processes may be used for failure detection: voting for the first failure, in-line detection for the second. Unless the recovery probabilities are very nearly one, the maximum benefits of redundancy will not be realized.

Problems encountered in implementing parallel redundancy include: selection logic, nuisance trips, generic failures, reliability of voting/selection units, control force contention, cross-strapping, increased cost and maintenance, number of operating channels required for dispatch, and connectors. Failure-detection logic must be sensitive to failures yet insensitive to small operational errors, including those due to non-colocation of sensors or actuators. Nuisance trips (false indications of failure) must be minimized to assure that useful resources are kept on-line and missions are not aborted prematurely. Redundancy does not preclude identical damage to parallel systems, especially when they are located in close proximity. Cross-strapping implies complex, "intelligent" interconnections; however, if it is not implemented, a single component failure brings down an entire control string. Voting can be done in all operating control computers, but arbitration is required when these computers disagree. For the ideal parallel system, the probability P_c that some component will fail is,

$$P_c \approx M[(\lambda_s + \lambda_c + \lambda_a)(t_f - t_0)] \qquad (15$$

so the likelihood of component failure is increased by redundancy. It is necessary to establish rules for dispatching the controlled system: if one control string is not operational but the others are, should the process be initiated? For a manufacturing system, the answer might be "yes," while for a transport aircraft, it might be "no." A non-trivial aspect of redundant control is the need for more electrical connectors, the components most likely to cause trouble!

One insidious problem associated with parallel redundancy is the lack of controllability of internal state components [22]. Consider the dual-redundant controlled system of Fig. 4, where the individual control outputs are averaged by $M_1 = M_2$, and $F_1 = F_2$, $G_1 = G_2$, and $N_1 = N_2$. The dynamic equations can be expressed as

$$\begin{bmatrix}\dot{x}_A\\ \dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}F_A & G_A M_1 & G_A M_1\\ G_1 N_1 & F_1 & 0\\ G_1 N_1 & 0 & F_1\end{bmatrix}\begin{bmatrix}x_A\\ x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ G_1\\ G_1\end{bmatrix} u_D \qquad (16$$

The controllability matrix C of this system is

$$\mathbb{C} = \begin{bmatrix}0 & 2G_A M_1 G_1 & 2(F_A G_A M_1 + G_A M_1 F_1)G_1 & \cdots\\ G_1 & F_1 G_1 & (2N_1 G_A M_1 + F_1{}^2)G_1 & \cdots\\ G_1 & F_1 G_1 & (2N_1 G_A M_1 + F_1{}^2)G_1 & \cdots\end{bmatrix} \qquad (17$$

Complete controllability requires that $\mathbb{C}$ be of maximal rank; however, that is not possible because the bottom two rows are repeated. In other words, the compensator state elements are not controllable. If the corresponding modes are stable, then small variations between the two controllers tend to decay; however, if the modes are unstable or neutrally stable (as in the case of integral compensation), uncontrollable drift can occur, leading to divergent control outputs, nuisance trips, and possible isolation of otherwise operable channels.

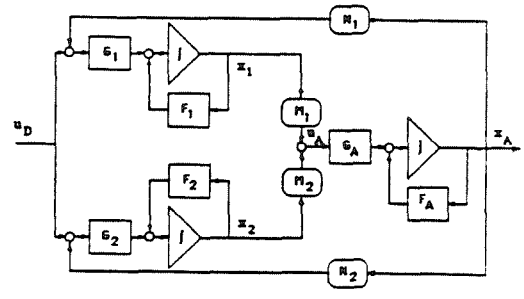

Figure 4. Model of a dual-redundant controller.

[1] In the present context, "sensor" implies the entire suite of sensors needed for control, and "computer" and "actuator" are defined similarly.

If there are sufficient cues to warn a human operator of system failure and plausible failure effects are slow enough to allow manual intervention, many of the benefits of parallel redundancy can be obtained by operating with a single control string, keeping an idle backup control string at the ready. The backup system can be similar or dissimilar to the primary system; however, if it is less capable, ability to perform the task will be degraded.

Parallel redundancy can protect against control-system component failures, but it does not address failures of plant components. Analytical redundancy provides a capability to improve tolerance to failures of both types. It does this with fewer additional components, flexible cross-strapping, and increased computation; as a consequence, there is greater reliance on the control computer, producing even greater need for computer reliability.

Analytical Redundancy

The principal functions of analytical redundancy are failure detection (through built-in-test alarms or off-nominal operation), failure identification (recognition of which components are failed), and control-system reconfiguration (adaptation to sensed or estimated failures). Detection and identification may be combined in built-in test functions. Although in-line monitors provide direct and rapid response to specific failures, it is impossible to provide full coverage of all failures by specialized instrumentation (which itself is subject to failure). A practical failure detection, identification and reconfiguration (FDIR) solution can be found in the control computer's ability to compare expected response to actual response, inferring component failures from the differences and changing either the structure or the parameters of the control system as a consequence.

Failure detection is exemplified by the generalized likelihood ratio test (Fig. 5) [23], which uses a Kalman-filter-like recursive equation to sense discrepancies in system response. The test compares the probability of the estimator's actual measurement residual [z - h(•)] with its expected value, detecting a jump that can be related to failure. It is very sensitive to off-nominal performance and is easy to implement; however, the test does not produce a tight indication of the failed element, and modeling errors can hamper detection [24].

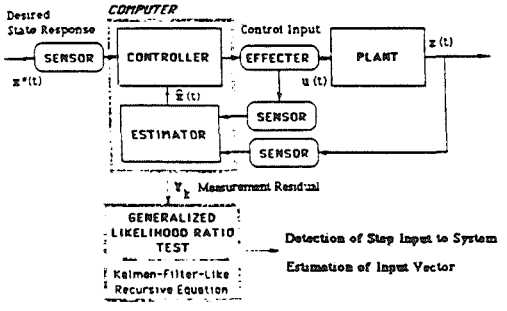

Figure 5. Failure detection: generalized likelihood ratio test.

Failure identification may require a more specific test, such as multiple-model hypothesis testing (Fig. 6) [25,26]. Each failure hypothesis (including that of no failure) is modeled in a Kalman filter, and the most likely hypothesis (based on probability estimates [4]) indicates the failure state. This is a computationally intensive technique, as not only the failed device must be hypothesized but the type, magnitude, and (if taken to the extreme) even the time of the failure must be modeled as well.

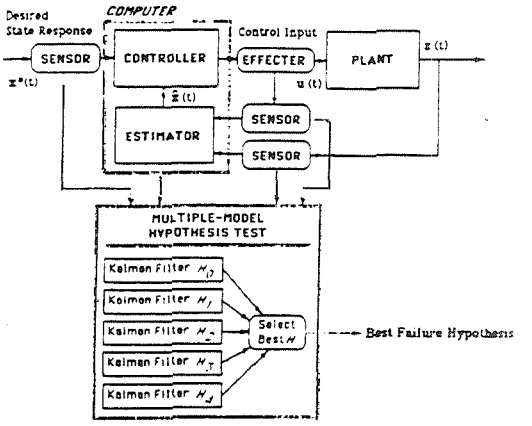

Figure 6. Failure identification: multiple-model hypothesis test.

Consider a modified form of the generic control structure:

$$\Delta u_k = - S_c C_k[\xi_k] + b_c \qquad (18$$

$$\xi_{k+1} = \Psi_k \xi_k + \Theta_k \Delta u_k + K_k[S_s z_k - h(\hat{x}_k, u_k) + b_s] \qquad (19$$

S_s and S_c are scale-factor matrices on the measurements and control, and b_s and b_c are bias vectors. Within this framework, we can identify the elements of the control system that need to be modified following various failures, as in Table 1. If the plant is altered, it may be necessary to change the internal model (Ψ, Θ), as well as the estimation and control gains (K, C), and so on for the remaining failure types. Precise failure identification is an important antecedent of control reconfiguration. Both "hard" (fast) and "soft" (slow) failures must be expected, and logic must accommodate command inputs (set-point transients), disturbances, and measurement noise [27].

Table 1
Failure Types and Related Control-Law Parameters

Failure	Parameter
Plant Alteration	Ψ, Θ, K, C
Actuator Failure	u, Θ, C
Sensor Failure	z, h, K
Bias Shift	b_s or b_c
Scale Factor Shift	S_s or S_c
Saturation Limit Change	K or C
Drift	b_s or b_c
Open Circuit	u, Θ, C, and/or z, h, K
Hardover/Stuck	Open Circuit, plus b_s and/or b_c
Noise	K

Reconfiguration attempts to retain nominal stability and performance characteristics. At a minimum, this requires that on-design controllability and observability (e.g., [4]) be preserved. There is a tradeoff between speed of reconfiguration, computer storage requirements, and flexibility of reaction. Controller structures and parameters for all conceivable failed states can be generated off-line and stored for eventual use; however, this ap-

proach could require an enormous memory. Conversely, on-line design requires minimal storage and (in principle) can adjust to unanticipated failures, but design algorithms must be executed and their results accepted soon enough to provide sufficient failure tolerance. With failed sensors, reconstruction of missing measurements may increase state-estimate errors; with failed effecters, the remaining actuators may have to operate with larger displacements and rates [28]. If the plant is open-loop-unstable, higher control activity combined with existing control-saturation limits may reduce the state space within which closed-loop stability can be assured [29,30].

Artificial Intelligence

Control theory and artificial intelligence both strive to harness mathematics and logic for practical problem solving, but control theory finds its origins in dynamics and electronics, while artificial intelligence springs from biology, psychology, and computer science. Failure-tolerant control systems can benefit from blending these perspectives. Two approaches have been followed in the field of artificial intelligence. *Artificial neural networks* are motivated by input-output and learning properties of living neural networks, although in application the network becomes an abstraction that may bear little resemblance to its biological namesake. *Expert systems* mimic the intelligent functions of an expert or group of experts. Initially, artificial neural networks appeared impractical because computers of the day were too slow and massive, and methods for training neural networks (e.g., *perceptrons* and *adalines*) were thought to be unworkable [31,32]. In the intervening years, the expert system approach proved to be quite achievable; hence, it received major emphasis in both theoretical development and applications. New insights about learning and improved electronics have restored interest in neural networks.

Expert Systems

Expert systems are computer programs that use heuristic relationships and facts as human experts do. The tasks and requirements of such systems (Table 2 [33]) are important for reconfigurable control systems, but there is a need to go beyond the usual limitations of static expert systems. Interpretation, diagnosis, monitoring, prediction, planning, and design must be cyclical, dynamic processes that can reconfigure the control system in "real time" (i.e., with negligible delay).

Table 2
Functions of an Expert System

Task	Requirements
Interpretation	Correct, consistent, complete analysis of data
Diagnosis	Fault finding
Monitoring	Recognition of alarm conditions
Prediction	Reasoning about time, forecasting the future
Planning	Defining actions to achieve goals
Design	Creating objects that satisfy requirements

The expert system offers a useful formalism for failure-tolerant control because it can consider diverse data sources and subproblem abstractions. The expert system can combine qualitative and quantitative reasoning, heuristics and statistics [34]. Failure indicators may be continuous variables generated by measurements or estimators, or they may be discrete variables from in-line monitors or discrete-event models. Indicators are the outputs of *productions*, routines with unique input-output characteristics that produce goal conditions from initial conditions. Hence, the expert system can be implemented as a *production system* or a *rule-based system* consisting of a *data base*, a *rule base*, and a rule interpreter (or *inference engine*) [35]. A production system generates actions predicated on the data base, which contains measurements as well as stored data or operator inputs.

A rule-based failure-tolerant control system contains FDIR logic in expert-system format (Fig. 7). The expert system is an adjunct to the nominal control structure, which remains the most efficient means of effecting precise control. From the control perspective, the expert system performs its decision-making tasks in a concentric *outer loop*; from the expert-system perspective, control activity is a *side effect* that supports decision making.

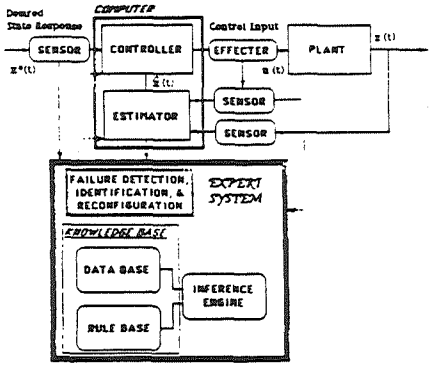

Figure 7. Expert-system approach to analytical redundancy.

An expert system performs *deduction* using *knowledge* and *beliefs* expressed as parameters and rules (Fig. 8). *Parameters* have values that either are external to the expert system or are set by rules. An "IF-THEN" *rule* evaluates a *premise* by testing values of one or more parameters related by logical "ANDs" or "ORs," as appropriate, and it specifies an *action* that set values of one or more parameters. The rule base contains all the rules of the expert system, and the inference engine performs its function by searching the rule base. Given a set of premises (evidence of the current state), the logical outcome of these premises is found by a data-driven search (*forward chaining*) through the rules. Given a desired or unknown parameter value, the premises needed to support the fixed or free value are identified by a goal-directed search (*backward chaining*) through the rules. Querying (or firing) a rule when searching in either direction may invoke procedures that produce parameter values as side effects.

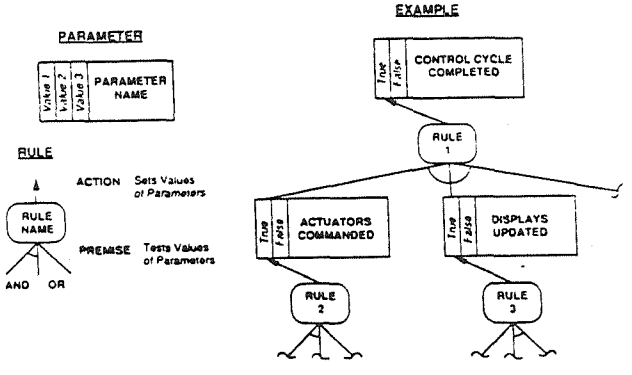

Figure 8. Graphical representation of expert system knowledge.

Both search directions are used in a rule-based control system [36]. Backward chaining drives the entire process by demanding that a parameter such as *CONTROL CYCLE COMPLETED* have a value of *true*. The inference engine works back through the rules to identify other parameters that allow this and, where necessary, triggers side effects like estimation and control to set these parameters to the needed values. Backward chaining also is invoked to learn the value of *ABNORMAL BEHAVIOR DETECTED*, be it *true* or *false*. Conversely, forward chaining indicates what actions can be taken as a consequence of the current state. If *SENSOR MEASUREMENTS REASONABLE* is *true*, and *ALARM DETECTED* is *false*, then failure identification and reconfiguration side effects can be skipped on the current cycle.

Rules and parameters can be represented as *objects* or *frames* that have identities and attributes. For example, a rule can be expressed as the ordered list (*NAME, STATUS, PREMISE, ACTION, ACTION PARAMETERS, PREMISE PARAMETERS, TRANSLATION*), while a parameter may take the form (*NAME, USING RULES, UPDATING RULES, ALLOWABLE VALUES, TRANSLATION*). Most of these attributes are self-explanatory. *STATUS* indicates the state of the rule, such as "not been tested," "being tested," "tested, and premise is *true*," "tested, and premise is *false*," or "tested, and premise is *unknown*." *ALLOWABLE VALUES* provides a mechanism for detecting false logic. *TRANSLATION* provides a natural-language explanation for display to the operator. Specific rules and parameters are represented by lists in which names and attributes are replaced by their values. The attribute lists contain not only values and logic but additional information for the inference engine. This information can be used to compile *parameter-rule-association lists* that speed execution [37].

Frames provide useful parameter structures for related productions, such as analyzing the origin of one or more failures in a complex, connected system [38]. The *dependency graph* of Fig. 9 showing relationships between actuators and their power supplies can be represented by the random-order list ((*OBJECT Name*) (*ATTRIBUTE*$_1$ *Value*$_1$) (*ATTRIBUTE*$_2$ *Value*$_2$) (...)), a more flexible form than the previous structure. In this application, the (*ATTRIBUTE Value*) lists are (*A-KIND-OF Device*), (*ANTERIOR <-OR> Device<s>*), (*POSTERIOR<-OR> Device<s>*), (*CRITICALITY Number*), and (*UNITS Number*). Frames possess an inheritance property; thus the object ((*OBJECT Pivoting Actuator*) (*A-KIND-OF Actuator*) (*ANTERIOR Hydro-Reservoir*) (*POSTERIOR-OR* (*Swashplate Pitching-Link*))) lays claim to the properties of ((*OBJECT Actuator*) (*A-KIND-OF Hydraulic Device*) (*UNITS* (*I* 2))). A two-step process estimates the failure state. In local failure analysis, forward chaining assesses the impact of known malfunctioning units, and backward chaining finds possible causes of the anomalies. In global failure analysis, local failure models are combined, an inclusion property prunes redundant models, and a heuristic evaluation based on criticality, reliability, extensiveness, implications, level of backtracking, and severity produces a list of most likely failure models.

Expert systems process lists, so it is not surprising that LISP (LISt Processing) is the computer language of choice for preliminary development. However, LISP is not a fast, efficient language and is ill-suited to real-time applications. Moreover, a rule-based control system uses numerical algorithms that are most effectively coded in languages like Pascal, C, or FORTRAN. Consequently, *knowledge-base translation* from LISP to a procedural language is a useful (if not necessary) adjunct of rule-based control system design. This not only speeds program execution, it integrates control and decision-making processes, revealing new possibilities for incorporating diagnostic procedures in failure detection and identification [39].

Rule-based control systems must make decisions under uncertainty, and they can do so either by invoking *certainty-equivalent logic*, which is analogous to a well-known concept of stochastic optimal control, or by uncertainty management in the decison-making process. In the LQG regulator, uncertainties due

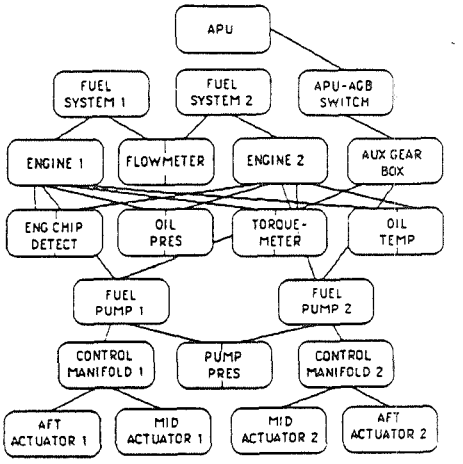

Figure 9. Dependency graph of a hydraulic control system.

to disturbances and measurement error are processed in the estimator, and the feedback control law operates on the state estimate as if it were the actual state [4]. The optimal control gains for the stochastic and deterministic cases are identical. Because the rule-based control system described above makes its best estimates of the failure state in the control logic, the expert system controlling FDIR can treat these results deterministically, realizing little or no improvement from further uncertainty processing. If inner-loop estimation is decidedly sub-optimal, uncertainty management can help, using probability theory, Dempster-Shafer theory, possibility theory, certainty factors, or the theory of endorsements [40]. Bayesian belief networks [41], which propagate event probabilities up and down a causal tree, have particular appeal for failure-tolerant control and are being applied in a related program to assist aircraft crews in avoiding hazards [42].

Teaching the expert system the rules and parameters that generalize the decision-making process from specific knowledge (the process of *induction*) is another concern. Here, we have followed two approaches at Princeton. The first is called *rule recruitment* [43], and it involves the manipulation of "dormant rules" (or *rule templates*). Each template possesses a fixed premise-action structure and refers to parameters through "pointers." Rules are constructed and incorporated in the rule base by defining links and modifying parameter-rule-association lists. Learning is based on repeated simulations of the controlled system with alternative failure scenarios. Learned parameter values then can be defined as "fuzzy functions" [44] contained in rule premises. The second approach [45] has two parts: analysis of variance identifies the factors that make statistically significant contributions to the decision metric, and the "ID3" algorithm [46] extracts rules from the training set by inductive inference. The rules take the form of *decision trees* that predict the performance of alternative strategies.

Expert systems are incorporated in the FDIR process to accommodate declarative functions, leaving reflexive functions to the estimation and control laws [43]. *Declarative action* requires a deep understanding of cause and possible effect. *Reflexive action* is automatic, quickly relating stimulus to response. Both are needed in intelligent failure-tolerant control.

Artificial Neural Networks

Artificial neural networks consist of nodes that simulate the neurons and weighting factors that simulate the synapses of a living nervous system. They are good candidates for performing a variety of reflexive functions in failure-tolerant control systems because they are potentially very fast (in parallel hardware implementation), they are intrinsically nonlinear, they can address problems of high dimension, and they can learn from experience. From the biological analogy, the neurons are modeled as switching functions that take just two discrete values; however, "switching" is softened to "saturation" in common usage, not only to facilitate learning of the synaptic weights but to admit the modeling of continuous functions.

The neural networks receiving most current attention are memoryless expressions that approximate functions of the form

$$y = f(x) \qquad (20$$

where x and y are input and output vectors and $f(\cdot)$ is the (possibly unknown) relationship between them. Neural networks can be considered *generalized spline functions* that identify efficient input-output mappings from observations [47,48]. Rather than approximating eq. 20 by a series, an N-layer neural network (Fig. 10) represents the function by recursive operations,

$$\mathbf{x}^{(k)} = \mathbf{s}^{(k)}[\mathbf{W}^{(k-1)}\mathbf{x}^{(k-1)}] \triangleq \mathbf{s}^{(k)}[\boldsymbol{\eta}^{(k)}] \text{ , } k = 1 \text{ to } N \text{ .} \qquad (21$$

where $y = \mathbf{x}^{(N)}$ and $x = \mathbf{x}^{(0)}$. $\mathbf{W}^{(k-1)}$ is a matrix of weighting factors determined by the learning process, and $\mathbf{s}^{(k)}[\cdot]$ is an activation-function vector whose elements are scalar, nonlinear functions $\sigma_i(\eta_i)$ appearing at each network node:

$$\mathbf{s}^{(k)}[\boldsymbol{\eta}^{(k)}] = [\sigma_1(\eta_1^{(k)}) \ldots \sigma_n(\eta_n^{(k)})]^T \qquad (22$$

One of the inputs to each layer may be a unity threshold element that biases the activation-function output.

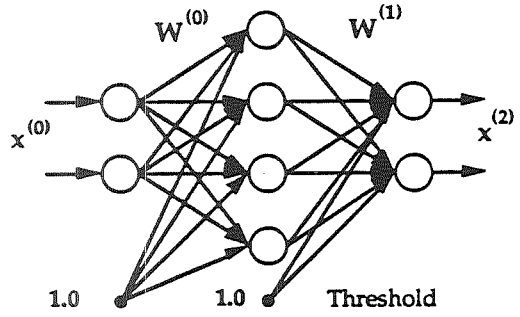

Figure 10. Backpropagation Feed-Forward Neural Network.

The *sigmoid* is commonly used as the artificial neuron. It is a saturating function defined variously as $\sigma(\eta) = 1/(1 + e^{-\eta})$ for output in (0,1) or $\sigma(\eta) = (1 - e^{-2\eta})/(1 + e^{-2\eta}) = \tanh \eta$ for output in (-1,1). Recent results indicate that any continuous mapping can be approximated arbitrarily closely with sigmoidal networks containing a single hidden layer (N = 2) [49,50]. It appears that certain symmetric functions, such as the *radial basis function* ($\sigma(\eta) = e^{-\eta^2}$) or the derivative of the sigmoid have even better convergence properties. Backpropagation learning algorithms for the elements of $\mathbf{W}^{(k)}$ typically involve a gradient search (e.g., [51]), although learning speed and accuracy are improved using the extended Kalman filter [52]. The Cerebellar Model Articulation Controller (CMAC) is an alternative neural network formulation with somewhat different properties but similar promise for application in control systems [53].

Equation 20 can represent many functions of importance in dynamics and control. For example, defining x as [$\mathbf{x}(t)$, $\mathbf{u}(t)$, $\mathbf{w}(t)$, $\mathbf{p}$], eq. 1 takes that form; together with the implied integration, neural networks can model plant dynamics. A discrete-time model of truck dynamics is demonstrated in [54], and a means of using neural networks in system identification is described in [55]. With x = [$\mathbf{x}(t)$, $\mathbf{u}(t)$, $\mathbf{w}(t)$, $\mathbf{n}(t)$, $\mathbf{p}$], the measurement vector (eq. 2) also could be represented. There is little advantage to expressing a linear control law like eq. 7 by a neural network; however, if the control gain matrix C is scheduled by operating point or time, that relationship could be modeled by a neural network. If a nonlinear control function such as $\mathbf{u} = \mathbf{u}(\mathbf{x}, \mathbf{x}_{desired}, t)$ is generated by optimization, nonlinear inversion, or model matching, it can be represented by a neural network [e.g., 54,56,57]. Consequently, neural networks can be incorporated in most of the control and FDIR techniques mentioned above.

Neural networks can be applied to failure detection and identification by mapping data patterns (or *feature vectors*) associated with failures onto detector/identification vectors (e.g., [58,59]). The network is trained to detect failure with the scalar output "1" corresponding to all failure patterns and "0" corresponding to no failure. During operation, a failure is indicated when the output exceeds some threshold near "1." To identify specific failures, the output is a vector, with a training value of "1" in the i^{th} element corresponding to the i^{th} failure mode. For M failure modes, either M neural networks with scalar outputs are employed or a single neural network with M-vector output is used; there are evident tradeoffs related to efficiency, correlation, and so on. The data patterns associated with each failure may require *feature extraction*, pre-processing that transforms the input time series into a feature vector. In [59], this was done by computing 24 Fourier coefficients of the input signal in a moving temporal window. When assessing FDI logic, feature extraction must be considered part of the neural-network computation.

Of course, not all of the suggested neural nets can learn on-line, as a training set must contain desired outputs as well as available inputs. In the cited examples, [54] and [56,58,59] use *off-line learning*, while [55] and [57] allow *on-line learning*. Reference 60 trains a neural network using an expert system that previously learned the desired control strategy. Once an initially trained system is on-line, the "off-line" training process could be executed in parallel with the on-line operation, allowing updates to be made. If the control process that generates on-line training data performs satisfactory control, the need for the neural network must be questioned. The goal should be to provide satisfactory failure tolerance with minimum hardware and software.

Neural networks intended to detect failures would learn little from monitoring normally operating plants. In any case, the neural-network learning rate is slow, probably too slow to expect neural networks of appreciable dimension to adapt to system failures in real time. Hence, the immediate application of neural networks in failure-tolerant control systems is to approximating nonlinear functions used by the FDIR methods introduced earlier. On-line learning can fine-tune this logic over a period of time.

Conclusions

Intelligent failure-tolerant control can improve the operating characteristics of systems. These improvements depend upon a good knowledge of the plant, reliable control elements, and sufficient observability and controllability following failures. Inherent robustness, the ability to accommodate failures without adaptation, is a highly desirable attribute, but it may not be sufficient to contain all system failures. Because split-second decision and reconfiguration may be required, a high degree of pre-training should be assumed; even intelligent systems cannot learn about new failure modes and respond to them properly at the same time (except by chance). Failure-tolerant systems must be able to distinguish between failures, disturbances, and modeling errors, responding to each in the proper way. Probability theory provides an underlying theme that unifies failure-tolerant design, from the probability of instability of robust systems, through the probability of failure of redundant systems, to the probability of correct FDIR response in analytical redundancy. Artificial intelligence is a useful adjunct to parallel and analytical redundancy, as expert systems and artificial neural networks offer new alternatives for both declarative and reflexive response to system failures.

Acknowledgment

This paper expands on a talk given at the General Motors Research Laboratories, Warren, MI, October, 1988. The present work has been supported by the Federal Aviation Administration and the National Aeronautics and Space Administration under Grant No. NGL 31-001-252 and by the Army Research Office under Contract No. DAAL03-89-K-0092.

References

1. Turing, A., "Computing Machinery and Intelligence," *Mind*, Vol. 59, No. 536, Oct 1950, pp. 433-460.

2. Searle, J.R., "Is the Brain's Mind a Computer Program?," *Scientific American*, Vol. 262, No. 1, Jan 1990, pp. 26-31,

3. Harris, W.H., and Levey, J.S., *New Columbia Desk Encyclopedia*, Columbia University Press, New York, 1975.

4. Stengel, R.F., *STOCHASTIC OPTIMAL CONTROL: Theory and Application*, J.Wiley & Sons, New York, 1986.

5. Osder, S., "The DC-9-80 Digital Flight Guidance System's Monitoring Techniques," AIAA Paper No. 79-1704, *Proceedings of the 1979 AIAA Guidance and Control Conference*, Boulder, CO, Aug 1979, pp. 64-79.

6. Soroka, E., and Shaked, U., "On the Robustness of LQ Regulators," *IEEE Transactions on Automatic Control*, Vol. AC-29, No.7, July 1984, pp. 664-665.

7. Wong, P.K., and Athans, M., "Closed-Loop Structural Stability for Linear-Quadratic Optimal Systems," *IEEE Transactions on Automatic Control*, Vol. AC-22, No.1, Feb 1977, pp. 94-99.

8. Kalman, R.E., "When is a Linear Control System Optimal?" *Transactions of the ASME, Journal of Basic Engineering*, Vol. 86, Mar 1964, pp. 51-60.

9. Anderson, B.D.O., and Moore, J.B., *Linear Optimal Control*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

10. Lehtomaki, N.A., Sandell, N.R., and Athans, M., "Robustness Results in Linear-Quadratic-Gaussian Based Multivariable Control Designs," *IEEE Transactions on Automatic Control*, Vol. AC-26, No.1, Feb 1981, pp. 75-93.

11. Doyle, J.C., "Guaranteed Margins for LQG Regulators," *IEEE Transactions on Automatic Control*, Vol. AC-23, No.4, Aug 1978, pp. 756-757.

12. Doyle, J.C., and Stein, G., "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis," *IEEE Transactions on Automatic Control*, Vol. AC-26, No.1, Feb 1981, pp. 4-16.

13. Safonov, M.G., Laub, A.J., and Hartmann, G.L., "Feedback Properties of Multivariable Systems: The Role and Use of the Return Difference Matrix," *IEEE Transactions on Automatic Control*, Vol. AC-26, No.1, Feb 1981, pp. 47-65.

14. Dorato, P., ed., *Robust Control*, IEEE Press, New York, 1987.

15. Doyle, J. C., "Analysis of Feedback Systems with Structured Uncertainties," *IEE Proceedings*, Vol. 129, Part D, No. 6, pp. 242-250, Nov 1982.

16. Stengel, R.F., "Some Effects of Parameter Variations on the Lateral-Directional Stability of Aircraft," *Journal of Guidance and Control*, Vol. 3, No. 2, Apr 1980, pp. 124-131.

17. Stengel, R.F., and Ryan, L., "Stochastic Robustness of Linear-Time-Invariant Control Systems," to appear in *IEEE Transactions on Automatic Control*.

18. Ray, L.R., and Stengel, R.F., "Stochastic Stability and Performance Robustness of Linear Multivariable Systems," *Proceedings of the 1990 American Control Conference*, San Diego, May 1990, pp. 462-467.

19. Miller, I., and Freund, J.E., *Probability and Statistics for Engineers*, Prentice-Hall, Englewood Cliffs, 1977.

20. Westermeier, T.F., "Redundancy Management of Digital Fly-By-Wire Systems," *Proceedings of the 1977 Joint Automatic Control Conference*, San Francisco, June 1977, pp. 272-277.

21. Bouricius, W.G., *et al*, "Reliability Modeling for Failure-Tolerant Computers," *IEEE Transactions on Computers*, Vol. C-20, No. 11, Nov 1971, pp. 1306-1311.

22. Stengel, R.F., "Some Effects of Bias Errors in Redundant Flight Control Systems," *Journal of Aircraft*, Vol. 10, No. 3, Mar 1973, pp. 150-156.

23. Willsky, A., and Jones, H.L., "A Generalized Likelihood Ratio Approach to the Detection and Estimation of Jumps in Linear Systems," *IEEE Transactions on Automatic Control*, Vol. AC-21, No. 1, Feb 1976, pp. 108-112.

24. Horak, D.T., "Failure Detection in Dynamic Systems with Modeling Errors," *Journal of Guidance, Control, and Dynamics*, Vol. 11, No. 6, Nov-Dec 1988, pp. 508-516.

25. Willsky, A., "A Survey of Design Methods for Failure Detection in Dynamic Systems," *Automatica*, Vol. 12, No. 6, Nov 1976, pp. 601-611.

26. Basseville, M., "Detecting Changes in Signals and Systems -- A Survey," *Automatica*, Vol. 24, No. 3, May 1988, pp.309-326.

27. Merrill, W.C., DeLaat, J.C., and Bruton, W.M., "Advanced Detection, Isolation, and Accommodation of Sensor Failures -- Real-Time Evaluation," *Journal of Guidance, Control, and Dynamics*, Vol. 11, No. 6, Nov-Dec 1988, pp. 517-526.

28. Huang, C.Y., and Stengel, R.F., "Restructurable Control Using Proportional-Integral Implicit Model Following," *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 2, Mar-Apr 1990, pp. 303-309.

29. Hanson, G., and Stengel, R.F., "Effects of Displacement and Rate Saturation on the Control of Statically Unstable Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 7, No. 2, Mar-Apr 1984, pp. 197-205.

30. Shrivastava, P.C., and Stengel, R.F., "Stability Boundaries for Aircraft with Unstable Lateral-Directional Dynamics and Control Saturation," *Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 1, Jan-Feb 1989, pp. 62-70.

31. Minsky, M., and Papert, S., *Perceptrons: An Introduction to Computational Geometry*, M.I.T. Press, Cambridge, 1968 (rev. 1988).

32. McCorduck, P., *Machines Who Think*, W.H. Freeman, San Francisco, 1979.

33. Stefik, M., *et al*, "The Organization of Expert Systems, A Tutorial," *Artificial Intelligence*, Vol. 18, No. 2, Mar 1982, pp. 135-174.

34. Handelman, D.A., and Stengel, R.F., "Combining Quantitative and Qualitative Reasoning in Aircraft Failure Diagnosis," *AIAA Guidance, Navigation, and Control Conference*, Snowmass, CO, AIAA Paper No. 85-1905CP, Aug 1985.

35. Charniak, E., and McDermott, D., *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, MA, 1985.

36. Handelman, D.A., and Stengel, R.F., "Combining Expert System and Analytical Redundancy Concepts for Fault-Tolerant Flight Control," *Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 1, Jan-Feb 1989, pp. 39-45.

37. Handelman, D.A., and Stengel, R.F., "An Architecture for Real-Time Rule-Based Control," *Proceedings of the 1987 American Control Conference*, Minneapolis, MN, June 1987, pp. 1636-1642.

38. Huang, C.Y., and Stengel, R.F., "Failure Model Determination in a Knowledge-Based Control System," *Proceedings of the 1987 American Control Conference*, Minneapolis, MN, June 1987, pp. 1643-1648.

39. Handelman, D.A., and Stengel, R.F., "Perspectives on the Use of Rule-Based Control," in *Artificial Intelligence in Real-Time Control*, M.G. Rodd and G.J. Suski, ed., Pergamon Press, New York, 1989, pp.27-32.

40. Ng, K.-C., and Abramson, B., "Uncertainty Management in Expert Systems," *IEEE Expert*, Vol. 5, No. 2, Apr 1990, pp. 29-47.

41. Pearl, J., *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, Palo Alto, CA, 1988.

42. Stratton, D.A., and Stengel, R.F., "Probabilistic Reasoning for Intelligent Wind Shear Avoidance," AIAA Paper No. 90-3437, *1990 AIAA Guidance, Navigation & Control Conference*, Portland, OR, Aug 1990.

43. Handelman, D.A., and Stengel, R.F., "Rule-Based Mechanisms of Learning for Intelligent Adaptive Flight Control," *Proceedings of the 1988 American Control Conference*, Atlanta, June 1988, pp. 208-213.

44. Tong, R., "A Control Engineering Review of Fuzzy Systems," *Automatica*, Vol. 13, No. 6, Nov 1977, pp. 559-569.

45. Belkin, B., and Stengel, R.F., "Quantitative Knowledge Acquisition for Expert Systems," *Proceedings of the Space Operations, Applications, and Research Symposium*, Albuquerque, NM, June 1990.

46. Quinlan, J.R., "Discovering Rules by Induction from Large Collections of Samples," in *Expert Systems in the Micro Electronic Age*, D. Michie, ed., Edinburgh University Press, Edinburgh, 1979, pp. 169-201.

47. Poggio, T., and Girosi, F., "Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks," *Science*, Vol. 247, No. 4945, Feb 23, 1990, pp. 978-982.

48. Linse, D.J., and Stengel, R.F., "Neural Networks for Function Approximation in Nonlinear Control, *Proceedings of the 1990 American Control Conference*, San Diego, May 1990, pp. 675-679.

49. Funahashi, K.-I., "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, Vol. 2, 1989, pp.183-192.

50. Cybenko, G., "Approximation by Superposition of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, Vol. 2, No. 4, 1989, pp. 303-314.

51. Rumelhart, D., Hinton, G., and Williams, R., "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognitions, Vol. 1: Foundations*, D. Rumelhart and J. McClelland, ed., MIT Press, Cambridge, 1986.

52. Singhal, S., and Wu, L., "Training Feed-Forward Networks with the Extended Kalman Algorithm," *Proceedings of the 1989 International Conference on Acoustics, Speech, and Signal Processing*, Glasgow, May 1989, pp. 1187-1190.

53. Albus, J.S., "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, Vol. 97, Sept 1975, pp. 220-227.

54. Nguyen, D.H., and Widrow, B., "Neural Networks for Self-Learning Control Systems," *IEEE Control Systems Magazine*, Vol. 10, No. 3, Apr 1990, pp. 18-23.

55. Stengel, R.F., and Linse, D.J., "System Identification for Nonlinear Control Using Neural Networks," *Proceedings of the 1990 Conference on Information Sciences and Systems*, Princeton University, Mar 1990.

56. Fadali, M., *et al*, "Minimum-Time Control of Robotic Manipulators Using a Back Propagation Neural Network," *Proceedings of the 1990 American Control Conference*, San Diego, May 1990, pp. 2997-3000.

57. Miller, W.T., "Sensor-Based Control of Robotic Manipulators Using a General Learning Algorithm," *Journal of Robotics and Automation*, Vol. RA-3, No. 2, Apr 1987, pp. 157-165.

58. Passino, K., Sartori, M., and Antsaklis, P., "Neural Computing for Numeric-to-Symbolic Conversion in Control Systems," *IEEE Control Systems Magazine*, Vol. 9, No. 3, Apr 1989, pp. 44-52.

59. Naidu, S., Zafiriou, E., and McAvoy, T., "Use of Neural Networks for Sensor Failure Detection in a Control System," *IEEE Control Systems Magazine*, Vol. 10, No. 3, Apr 1990, pp. 49-55.

60. Handelman, D.A., Lane, S.H., and Gelfand, J.J., "Integrating Neural Networks and Knowledge-Based Systems for Intelligent Robotic Control," *IEEE Control Systems Magazine*, Vol. 10, No. 3, Apr 1990, pp. 77-87.