Toward Intelligent Flight Control

Robert F. Stengel, *Fellow, IEEE*

Abstract—**Flight control systems can benefit by being designed to emulate functions of natural intelligence. Intelligent control functions fall in three categories. Declarative actions involve decision making, providing models for system monitoring, goal planning, and system/scenario identification. Procedural actions concern skilled behavior and have parallels in guidance, navigation, and adaptation. Reflexive actions are more or less spontaneous and are similar to inner-loop control and estimation. Intelligent flight control systems will contain a hierarchy of expert systems, procedural algorithms, and computational neural networks, each expanding on prior functions to improve mission capability, to increase the reliability and safety of flight, and to ease pilot workload.**

Introduction

HUMAN pilots traditionally have provided the intelligence to fly manned aircraft in numerous ways, from applying manual dexterity through informed planning and coordination of missions. As aircraft characteristics have changed, and more importantly as the technology has allowed, an increasing share of the aircraft's intelligent operation has relied on proper functioning of electromechanical sensors, computers, and actuators. It has become possible to apply machine intelligence to flight control. The transition toward automating control intelligence has been evolutionary. In contemplating the effects of atmospheric turbulence, one of the Wright brothers wrote, "The problem of overcoming these disturbances by automatic means has engaged the attention of many ingenious minds, but to my brother and myself, it has seemed preferable to depend entirely on *intelligent control*" [1], [2]. The Wright brothers' piloting actions depended on proper interpretation of visual and inertial cues, demonstrating biological intelligent control. Later, panel displays of compass heading, pressure altitude, airspeed, aircraft attitude, and bearing to a radio station enhanced the intelligent behavior of human pilots. Stability augmentation systems that fed pitch rate to elevator or yaw rate to rudder could be considered the first intelligent systems that did not rely on the human pilot, while automatic bombing and landing systems carried machine intelligence to the point of "hands-off" flying for portions of the aircraft's mission.

Manuscript received July 11, 1992; revised January 4, 1993.

This work was supported in part 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.

The author is with Princeton University, Princeton, NJ 08544.

IEEE Log Number 9209633.

In a contemporary context, intelligent flight control has come to represent even more ambitious plans to

- make aircraft less dependent on proper human actions for mission completion,
- enhance the mission capability of aircraft,
- improve performance by learning from experience,
- increase the reliability and safety of flight, and
- lower the cost and weight of aircraft systems.

This paper presents concepts for intelligent flight control in the contemporary context, that is, through the aid of what were once called "artificial" devices for sensing, computation, and control. Emphasis is placed on alternatives for analysis and design of control logic rather than on the equipment that makes it possible (i.e., on software rather than hardware). There are many ways to partition and describe intelligent control. Here the intelligent control function is separated into three parts to distinguish between control functions according to a cognitive/biological hierarchy of *declarative, procedural, and reflexive functions*. Declarative functions are performed by the control system's *outer loops*, and reflexive functions are performed by its *inner loops*. An intermediate level of procedural functions have well defined input–output characteristics of a more complicated structure. Traditional design principles suggest that the outer-loop functions should be dedicated to a low-bandwidth, large-amplitude control commands, while the inner-loop functions should have high bandwidths and relatively lower amplitude actions. There is a logical progression from the sweeping, flexible alternatives associated with satisfying mission goals to more local concerns for stability and regulation about a desired path or equilibrium condition.

Foundations for Intelligent Flight Control

Intelligent flight control design draws on two apparently unrelated bodies of knowledge. The first is rooted in classical analyses of aircraft stability, control, and flying qualities. The second derives from human psychology and physiology. The goal considered here is to find new control structures that are consistent with the reasons for flying aircraft, that bring flight control systems to a higher level of overall capability.

Aircraft Flying Qualities and Flight Control

An aircraft requires guidance, navigation, and control so that it can perform its mission. As suggested by Fig.

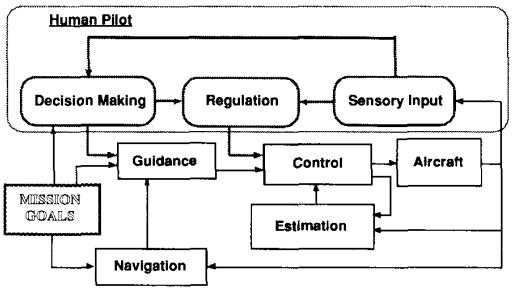


Fig. 1. Guidance, navigation, and control structure, distinguishing between human-pilot and computer-based functions.

1, a human pilot can interact with the aircraft at several levels, and his or her function may be supplanted by electromechanical equipment. The pilot performs three distinct functions: sensing, regulation, and decision making. These three tasks exercise different human characteristics: the ability to see and feel, the ability to identify and correct errors between desired and actual states, and the ability to decide what needs to be done next. The first of these depends on the body's sensors and the neural networks that connect them to the brain. The second relies on motor functions enabled by the neuromuscular system to execute learned associations between stimuli and desirable actions. The third requires more formal, introspective thought about the reasons for taking action, drawing on the brain's deep memory to recall important procedures or data. Sensing and regulation are high-bandwidth tasks with little time for deep thinking. Decision making is a low-bandwidth task that requires concentration. Each of these tasks exacts a workload toll on the pilot.

Pilot workload has become a critical issue as the complexity of systems has grown, and furnishing ideal flying qualities throughout the flight envelope is an imperative. It is particularly desirable to reduce the need to perform high-bandwidth, automatic functions, giving the pilot time to cope with unanticipated or unlikely events. In the future, teleoperated or autonomous systems could find increasing use for missions that expose human pilots to danger.

Research on the *flying (or handling) qualities of aircraft* has identified ways to make the pilot's job easier and more effective, and it provides models on which automatic systems might be based. The first flying qualities specification simply stated, "(the aircraft) must be steered in all directions without difficulty and all time (be) under perfect control and equilibrium" [3], [4]. Further evolution of flying qualities criteria based on dynamic modeling and control theory has resulted in the widely used U.S. military specification [5] and the succeeding military standard, described in [6].

The development of *control theoretic models of piloting behavior* proceeded in parallel with flying qualities research. Most of these models have dealt with reflexive, compensatory tracking tasks using simple time lag and transfer function models [7], [8] or Linear Quadratic Gaussian (LQG) optimal control models [9], [10]. Some of the transfer function approaches go into considerable detail about neuromuscular system dynamics [11], [12]. These models often show good correlation with experimental results, not only in compensatory tracking but in more procedural tasks. The progression of piloting actions from single- to multi-input strategies as the complexity of the task increases is predicted in [11], while test pilot opinion ratings are predicted by a "Paper Pilot" in [13]. These results imply that *computer based control laws can perform procedural and reflexive tasks within the fit error of mathematical human pilot models*. Models of the human pilot's declarative actions have yet to receive the same level of attention; however [14]–[17] introduce the types of decisions that must be made in aerospace scenarios, as well as likely formats for pilot–vehicle interface.

Fig. 1 also portrays a hierarchical structure for stability augmentation, command augmentation, autopilot, and flight management system functions that can be broken into reflexive and declarative parts. Stability augmentation is reflexive control provided by the innermost loop, typically implemented as a linear feedback control law that provides stability and improves transient response through an *Estimation/Compensation* block. Forward loop control provides the shaping of inputs for satisfactory command response through a *Control/Compensation* block, again employing linear models. The combination of control and estimation can be used to change the flying qualities perceived by the pilot, or it can provide a decoupled system for simplified guidance commands [18]–[20]. A basic autopilot merely translates the human pilot's commands to guidance commands for constant heading angle, bank angle, or airspeed, while the *Guidance* block can be expanded to include declarative flight management functions, using inputs from *Navigation* sensors and algorithms.

Intelligent functions have been added to flight control systems in the past. Gain scheduling and switching improve performance in differing flight regimes and mission phases. Control theory, heuristics, and reduced-order optimization have been used to achieve near-optimal trajectory management in many flight phases (e.g., [21]–[23]). The Guidance, Navigation, and Control (GNC) Systems for Project Apollo's Command/Service and Lunar Modules provide an early example of intelligent aerospace control [24]–[26]. The state of the art of aircraft flight control systems has progressed to comparable levels and beyond, as represented by systems installed in modern transport and fighter aircraft (e.g., [27], [28]).

Intelligent flight control[1] is justified only if it materially improves the functions of aircraft, if it saves the time and/or money required to complete a mission, or if it improves the safety and reliability of the system. A number of philosophical problems can be posed. Must machine intelli-

[1]As used here "intelligent flight control" subsumes "intelligent guidance, navigation, and control."

gence be better than the human intelligence it replaces in order for it to be adopted? We accept the likelihood that humans will make mistakes; if a machine has the same likelihood of making a mistake, should it be used? Lacking firm knowledge of a situation, humans sometimes gamble; should intelligent machines be allowed to gamble? When is it acceptable for machine intelligence to be wrong (e.g., during learning)? If the control system adapts, how quickly must it adapt? Must learning occur on-line, or can it be delayed until a mission is completed? Which decisions can the machine make without human supervision, and which require human intervention? How much information should be displayed to the human operator? Should intelligent flight control ever be fully autonomous? The "right" answers depend on the application and the mission.

Cognitive and Biological Paradigms for Intelligence

Intelligence is the "ability involved in calculating, reasoning, perceiving relationships and analogies, learning quickly, storing and retrieving information . . . classifying, generalizing, and adjusting to new situations" [29]. This definition does not deal with the mechanisms by which intelligence is realized, and it makes the tacit assumption that intelligence is a human trait. Intelligence relates not only to intellectuality and cognition but to personality and the environment [30].

The debate over whether or not computers ever will "think" may never be resolved, though this need not restrict our working models for computer-based intelligent control. One argument against the proposition is that computers deal with syntax (form), while minds deal with semantics (meaning), and syntax alone cannot produce semantics [31]. Of course, a computer may *mimic* natural intelligence in a limited domain. Another contention is that thinking is "nonalgorithmic," that the brain evokes consciousness through a process of natural selection and inheritance [32]. Consciousness is required for common sense, judgment of truth, understanding, and artistic appraisal, concepts that are not formal and cannot readily be programmed for a computer (i.e., they are not syntactic).

Conversely, functions that are automatic or "mindless" (i.e., that are *un*conscious), could be programmed, implying that computers have more in common with "unintelligent" functions. Gödel's Theorem[2] is offered in [33] as an example of an accepted proposition that may be considered nonalgorithmic; the statement and proof of the theorem must themselves be nonalgorithmic and, therefore, not computable.

The notion that syntax alone cannot produce semantics is attacked as being an axiom that is perhaps true, but not knowable in any practical sense [34]; therefore, the possibility that a computer can "think" is not ruled out. A further defense is offered in [35], which suggests that human inference may be based, in part, on inconsistent axioms. This could lead to rule-based decisions that are not logically consistent, that are affected by heuristic biases or sensitivities, that may reflect deeper wisdom, or that may be wrong or contradictory.

More to our point, it is likely that a computer capable of passing a flying qualities/pilot workload/control theoretic equivalent of the Turing test[3] [36] could be built even though that computer might not understand what it is doing.[4] For intelligent flight control, the principal objective is improved control performance, that is, for improved input-output behavior. The computer can achieve the operative equivalent of consciousness in a limited domain, even if it does not possess emotions or other human traits.

From an information processing perspective, it is convenient—as well as consistent with empirical data—to identify four types of thought: conscious, preconscious, subconscious, and unconscious [37]. *Conscious thought* is the thought that occupies our attention, that requires focus, awareness, reflection, and perhaps some rehearsal. Conscious thought performs declarative processing of the individual's knowledge or beliefs. It makes language, emotion, artistry, and philosophy possible. *Unconscious thought* "describes those products of the perceptual system that go unattended or unrehearsed, and those memories that are lost from primary memory through display or displacement" [37]. Within the unconscious, we may further identify two important components. *Subconscious thought* is procedural knowledge that is below our level of awareness but central to the implementation of intelligent behavior. It facilitates communication with the outside world and with other parts of the body, providing the principal home for the learned skills of art, athletics, control of objects, and craft. We are aware of perceptions if they are brought to consciousness, but they also may take a subliminal (subconscious) path to memory. *Preconscious thought* is preattentive declarative processing that helps choose the objects of our conscious thought, operating on larger chunks of information or at a more symbolic level. It forms a channel to long-term and implicit memory, and it may play a role in judgment and intuition.

The central nervous system supports a hierarchy of intelligent and automatic functions with *declarative actions* at the top, *procedural actions* in the middle, and *reflexive actions* at the bottom. We may assume that *declarative thinking* occurs in the brain's cerebral cortex, which accesses the interior limbic system for memory [38]–[40]. Together, they provide the processing unit for conscious thought. Regions of the cerebral cortex are associated with different intellectual and physical functions; the distinc-

[2]As summarized in [32]: Any algorithm used to establish a mathematical truth cannot prove the propositions on which it is based. Or another [33]: Logical systems have to be fixed up "by calling the undecidable statements axioms and thereby declaring them to be true," causing new undecidable statements to crop up.

[3]Turing suggested that a computer could be considered "intelligent" if it could "converse" with a human in a manner that is indistinguishable from a human conversing with a human.

[4]Searle describes such a computer as a "Chinese Room" that translates Chinese characters correctly by following rules while not understanding the language in [31].

tion between conscious and preconscious function may depend on the activation level and duration in regions of the cerebral cortex.

The working memory of conscious thought has access to the spinal cord through other brain parts that are capable of taking *procedural action* (e.g., the brain stem for autonomic functions, the occipital lobes for vision, and the cerebellum for movement). Procedural action can be associated with subconscious thought, which supports voluntary automatic processes like movement and sensing. These voluntary signals are sent over the somatic nervous system, transmitting to muscles through the motor neural system and from receptors through the sensory neural system.

The spinal cord itself "closes the control loop" for *reflexive actions* long before signals could be processed by the brain. Nevertheless, these signals are available to the brain for procedural and declarative processing. We are all aware of performing some task (e.g., skating or riding a bicycle) without effort, only to waver when we focus on what we are doing. Involuntary regulation of the body's organs is provided by the autonomic nervous system, which is subject to unconscious processing by the brain stem. "Bio-feedback" can be learned, allowing a modest degree of higher level control over some autonomic functions.

Declarative, procedural, and reflexive functions can be built into a model of intelligent control behavior (Fig. 2). The *Conscious Thought* module governs the system by performing declarative functions, receiving information and transmitting commands through the *Subconscious Thought* module, which is itself capable of performing procedural actions. Conscious Thought is primed by *Preconscious Thought* [41], which can perform symbolic declarative functions and is alerted to pending tasks by Subconscious Thought. These three modules overlie a bed of deeper *Unconscious Thought* that contains long-term memory.

The Subconscious Thought module receives information from the *Sensory System* and conveys commands to the *Muscular System* through peripheral networks. Voluntary *Reflexive Actions* provide low-level regulation in parallel with the high-level functions, responding to critical stimuli and coordinating control actions. High- and low-level commands may act in concert, or one may block the other. Voluntary Reflexive Actions can be trained by high-level directives from Subconscious Thought, while the learning capabilities of involuntary Reflexive Action are less clear. Control actions produce *Body* motion and can affect an external *Controlled System*, as in piloting an aircraft. In learned control functions, Body motion helps internalize the mental model of Controlled System behavior. The Body and the Controlled Systems are both directly or indirectly subjected to *Disturbances*; for example, turbulence would affect the aircraft directly and the pilot indirectly. The Sensory System observes *External Events* as well as Body and Controlled System motions, and it is subject to *Measurement Errors*.

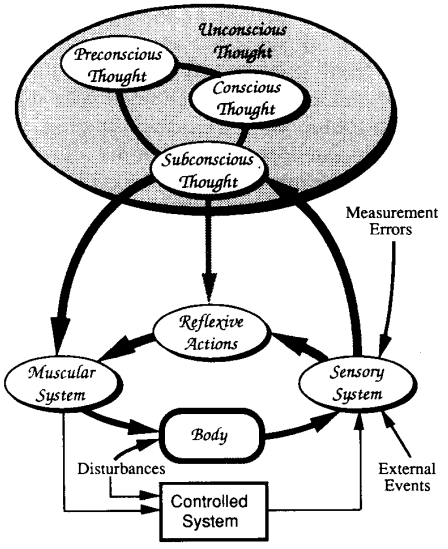


Fig. 2. A Model of cognitive/biological control behavior.

Human cognition provides paradigms for control system design. One important observation is that *learning requires error or incompleteness*. There is nothing new to be gained by observing a known process that is operating perfectly. In a control context, any operation should be started using the best available knowledge of the process and the most complete control resources. Consequently, learning is not always necessary or even desirable in a flight control system. *Biological adaptation is a slow process*, and proper changes in behavior can be made only if there is prior knowledge of alternatives. If adaptation occurs too quickly, there is the danger that misperceptions or disturbance effects will be misinterpreted as parametric effects. *Biological systems need rest*, and this attribute may be an essential element of intelligence. For example, *REM (rapid eye movement) sleep* appears to be a time of learning, consolidating, and pruning knowledge[5] [42]. Computer systems could learn even when they are not functioning by reviewing past performance, perhaps in a repetitive or episodic way.

Human biology provides structural paradigms for control that are worth emulating in machines. First, there is a *richness of sensory information* that is hard to fathom, with millions of sensors providing information to the system. This results in high signal to noise ratio in some cases, and it allows symbolic/image processing in others. Those signals requiring high bandwidth, high resolution channel capacity (vision, sound, and balance) have *short, dedicated, parallel runs* from the sensors to the brain. *Dissimilar but related sensory inputs facilitate interpretation of data*. A single notion can be sensed by the eyes, by the inner ear, and by the "seat-of-the-pants" (i.e., by

[5]"In REM sleep, the brain is barraged by signals from the brain stem. Impulses fired to the visual cortex produce images that may contain materials from the day's experiences, unsolved problems, and unfinished business." [42]

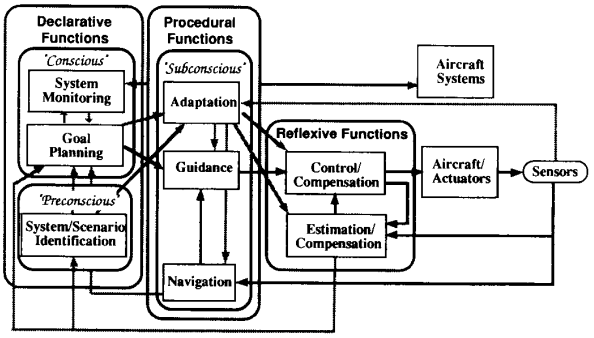


Fig. 3. Intelligent flight control system structure.

sensing forces on the body itself), corroborating each other and suggesting appropriate actions. When these signals are made to disagree in moving-cockpit simulation of flight, a pilot may experience a sense of confusion and disorientation.

There are *hierarchical and redundant structures* throughout the body. The nervous system is a prime example, bringing inputs from myriad sensors (both similar and dissimilar) to the brain, and performing low-level reasoning as an adjunct. Many sensing organs occur in pairs (e.g., eyes, ears, inner ears), and their internal structures are highly parallel. *Pairing allows graceful degradation in the event that an organ is lost.* Stereo vision vanishes with the loss of an eye, but the remaining eye can provide both foveal and peripheral vision, as well as a degree of depth perception through object size and stadiametric processing. Our control effectors (arms, hands, legs, feet) also occur in pairs, and there is an element of "Fail-Op/Fail-Op/Fail-Safe" design [43] in the number of fingers provided for manual dexterity.

Structure for Intelligent Flight Control

The preceding section leads to a control system structure that overlays the cognitive/biological model of Fig. 2 on the flight control block diagram of Fig. 1 and adds new functions. The suggested structure (Fig. 3) has *superblocks* identifying declarative, procedural, and reflexive functions; these contain the classical GNC functions plus new functions related to decision making, prediction, and learning. The black arrows denote information flow for the primary GNC functions, while the gray arrows illustrate the data flow that supports subsidiary adjustment of goals, rules, and laws.

Within the super blocks, higher-level functions are identified as conscious, preconscious, and subconscious attributes as a working analog for establishing a computational hierarchy. The new functions relate to setting or revising goals for the aircraft's mission, monitoring and adjusting the aircraft's systems and subsystems, identifying changing characteristics of the aircraft and its environment, and applying this knowledge to modify the structures and parameters of GNC functions.

In the remainder of the paper, declarative, procedural, and reflexive control functions are discussed from an aerospace perspective. In practice, the boundaries between mission tasks may not be well defined, and there is overlap in the kinds of algorithms that might be applied within each group. A number of practical issues related to human factors, system management, certifiability, maintenance, and logistics are critical to the successful implementation of intelligent flight control, but they are not treated here.

Declarative Systems

Goal planning, system monitoring, and control mode switching are declarative functions that require reasoning. Alternatives must be evaluated, and decisions must be made through a process of *deduction*, that is, by inferring answers from general or domain-specific principles. The inverse process of learning principles from examples is *induction*, and not all declarative systems have this capability. Most declarative systems have fixed structure and parameters, with knowledge induced off-line and before application; declarative systems that learn on-line must possess a higher level of reasoning ability, perhaps through an internal declarative module that specializes in training. Flight control systems that incorporate declarative logic can exhibit intelligent behavior by emulating the functions of an aircraft crew [47], [48].

Expert Systems

Expert Systems are computer programs that use physical or heuristic relationships and facts for interpretation, diagnosis, monitoring, prediction, planning, and design. In principal, an expert system replicates the decision making process of one or more experts who understand the causal or structural nature of the problem [44]. While human experts may employ "nonmonotonic reasoning" and "common sense" to deduce facts that apparently defy simple logic, computational expert systems typically are formal and consistent, basing their conclusions on analogous cases or well defined rules.[6]

A *rule-based expert system* consists of *data*, *rules*, and an *inference engine* [46]. It generates actions predicated on its data base, which contains measurements as well as stored data or operator inputs. An expert system performs deduction using *knowledge* and *beliefs* expressed as parameters and rules. *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 sets values of one or more parameters.

The rule base contains all the cause and effect relationships of the expert system, and the inference engine per-

[6]Expert systems can have *tree* or *graph* structures. In the former, there is a single *root* node, and all final (*leaf*) nodes are connected to their own single *branch*. In the latter, one or more branches lead to individual nodes. Reasoning is *consistent* if an individual node is not assigned differing values by different branches [45].

forms 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 through *side effects*. Side effects provide the principal means for taking measurements, making state estimates, invoking control logic, and commanding actuators.

Both search directions are used in rule-based control systems [47]. 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 (procedural or reflexive functions) to set those parameters to the needed values. Backward chaining also is invoked to learn the value of *ABNORMAL BEHAVIOR DETECTED (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* using ordered lists that identify names and attributes. Specific rules and parameters are represented by lists in which values are given to the names and attributes. 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 [48]. Frames provide useful parameter structures for related productions, such as analyzing the origin of one or more failures in a complex control system [49].

Crew/Team Paradigms for Declarative Flight Control

Logical task classification is a key factor in the development of rule-based systems. To this point, we have focused on the intelligence of an individual as a paradigm for control system design, but it is useful to consider the hypothetical actions of a multi-person aircraft crew as well. In the process, we develop an expert system of expert systems, a hierarchical structure that reasons and communicates like a team of cooperating, well-trained people might. This notion is suggested in [50] and is carried to considerable detail in [51]–[53]. The Pilot's Associate Program initially focused on a four task structure and evolved in the direction of the multiple crew member paradigm [54]–[56].

AUTOCREW is an ensemble of nine cooperating rule-based systems, each figuratively emulating a member of a World War II bomber crew: executive (pilot), co-pilot, navigator, flight engineer, communicator, spoofer (countermeasures), observer, attacker, and defender (Fig. 4) [53]. The executive coordinates mission specific tasks and has knowledge of the mission plan. The aircraft's human pilot can monitor AUTOCREW functions, follow its suggestions, enter queries, and assume full control if confidence is lost in the automated solution. The overall goal is to reduce the pilot's need to regulate the system directly without removing discretionary options. AUTOCREW contains over 500 parameters and over 400 rules.

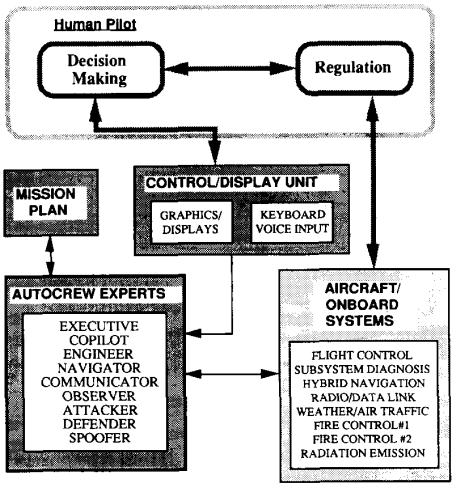


Fig. 4. AUTOCREW configuration with pilot/aircraft interface (adapted from [52]).

AUTOCREW was developed by defining each member expert system as a *knowledge base*, according to the following principles:

- Divide each knowledge base into major task groups: time-critical, routine, and mission-specific.
- Order the task groups from most to least time-critical to quicken the inference engine's search.
- Break major tasks into subtasks according to the detail necessary for communicating system functions.
- Identify areas of cooperation between knowledge bases.

The five task group types for each crew member are: tasks executed during attack on the aircraft, tasks executed during emergency or potential threat, tasks ordered by the EXECUTIVE, tasks executed on a routine basis, and mission-specific tasks. Routine and mission specific tasks are executed on each cycle; emergency tasks are executed only when the situation warrants.

Operation of AUTOCREW was simulated to obtain comparative expert system workloads for two mission scenarios: inbound surface to air missile attack and human pilot incapacitation [52]. In addition to the overall AUTOCREW system, a functioning NAVIGATOR sensor management expert system was developed. Additional perspectives on intelligent flight management functions can be obtained from the literature on decision making by teams, as in [57]–[59]. Alternate approaches to aiding the

pilot in emergencies are given in [60], [61]. Knowledge acquisition for the system presents an interesting challenge, because traditional methods (e.g., domain expert interviews) may not provide sufficiently detailed information for system design [62].

Reasoning Under Uncertainty

Rule-based control systems must make decisions under uncertainty. Measurements are noisy, physical systems are subject to random disturbances, and the environment within which decisions must be made is ambiguous. For procedural systems, the formalism of optimal state estimation provides a rigorous and useful means of handling uncertainty [63]. For declarative systems, there are a number of methods of uncertainty management, including probability theory, Dempster–Shafer theory, possibility theory (fuzzy logic), certainty factors, and the theory of endorsements [64].

Bayesian belief networks [65], which propagate event probabilities up and down a causal tree using Bayes's rule, have particular appeal for intelligent control applications because they deal with probabilities, which form the basis for stochastic optimal control. We have applied Bayesian networks to aiding a pilot who may be flying in the vicinity of hazardous wind shear [66]. Fig. 5 shows a network of the causal relationships among meteorological phenomena associated with microburst wind shear, as well as temporal and spatial information that could affect the likelihood of microburst activity. A probability of occurrence is associated with each node, and a conditional probability based on empirical data is assigned to each arrow. The probability of encountering microburst wind shear is the principal concern; however, each time new evidence of a particular phenomenon is obtained, probabilities are updated throughout the entire tree. In the process, the estimated likelihood of actually encountering the hazardous wind condition on the plane's flight path is refined. Unlike other applications of hypothesis testing, the threshold for advising a go-around during landing or an abort prior to takeoff is a very low probability—typically, 0.01 or less—as the consequences of actually encountering a strong microburst are severe and quite possibly fatal.

The safety of aircraft operations near microburst wind shear will be materially improved by forward-looking Doppler radar, which can sense variations in the wind speed. Procedural functions that can improve the reliability of the wind shear expert system include extended Kalman filtering of the velocity measurements at incremental ranges ahead of the aircraft [67].

Probabilistic reasoning of a different sort has been applied to a problem in automotive guidance that may have application in future Intelligent Vehicle/Highway Systems [68]–[70]. Intelligent guidance for headway and lane control on a highway with surrounding traffic is based on *worst-plausible-case decision making*. It is assumed that the intelligent automobile (IA) has imaging capability as well as on-board motion sensors; hence, it can deduce the speed and position of neighboring automobiles. Each automobile is modeled as a simple discrete-time dynamic system, and estimates of vehicle states are propagated using extended Kalman filters [63]. There are limits on the performance capabilities of all vehicles, and IA strategy is developed using time to collide, braking ratios, driver aggressiveness, and desired security factors. Plausible guidance commands are formulated by minimizing a cost function based on these factors [70]. A general layout of the logic shown in Fig. 6, and a partial decision tree for lateral guidance is presented in Fig. 7. Both normal and emergency expert systems govern the process, supported by procedural calculations for situation assessment, traffic prediction, estimation, and control. Guidance commands are formulated by minimizing a cost function based on these factors [70].

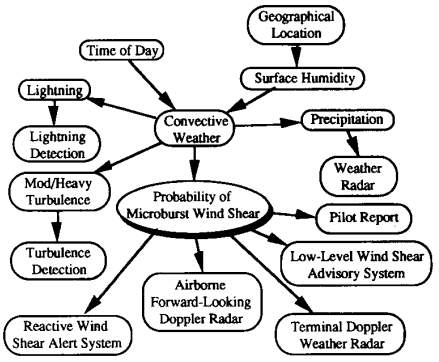


Fig. 5. Bayesian belief network to aid wind shear avoidance (adapted from [67]).

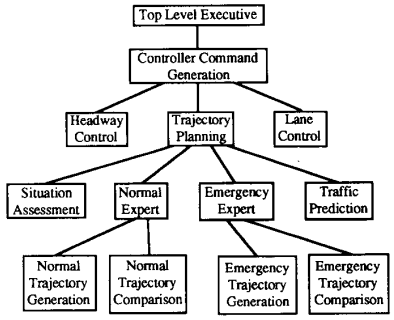


Fig. 6. Intelligent guidance for automative headway and lane control [69].

Alternate plausible strategies for each neighboring automobile are extrapolated, with predictions of both means and covariances. Expected values of *plausibility, belief interval, and hazard functions* are calculated, scores for feasible IA actions are computed, and the best course of action is decided, subject to aggressiveness and security factors, as suggested by Fig. 7. Deterministic and Monte Carlo simulations are conducted to demonstrate system performance and to fine-tune logical parameters.

Each of the expert systems discussed in this section

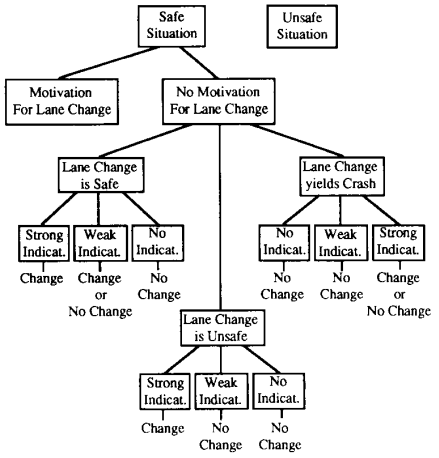


Fig. 7. Partial decision tree used to model lateral behavior in intelligent automative control [69].

performs deduction in a cyclical fashion based on prior logical structures, prior knowledge of parameters, and real-time measurements. It is clear that intelligent flight control systems must deal with unanticipated events, but it is difficult to identify aeronautical applications where on-line declarative learning is desirable. Nevertheless, off-line induction is needed to formulate the initial declarative system and perhaps (in a manner reminiscent of REM sleep) to upgrade declarative logic between missions.

Inducing Knowledge in Declarative Systems

In common usage, "learning" may refer a) to collecting inputs and deducing outputs and b) to inducing the logic that relates inputs and outputs to specific tasks. Here, we view the first process as the normal function of the intelligent system and the second as "learning." Teaching an expert system the rules and parameters that generalize the decision making process from specific knowledge is the inverse of expert system operation. Given all possible values of the parameters, what are the rules that connect them? Perhaps the most common answer is to interview experts in an attempt to capture the logic that they use, or failing that, to study the problem intensely so that one becomes expert enough to identify naturally intelligent solutions. These approaches can be formalized [71], [72], and they were the ones used in [67] and [68]. Overviews of alternatives for induction can be found in [45], [46], [73], [74].

Two approaches are considered in greater detail. The first is called *rule recruitment* [75], and it involves the manipulation of "dormant rules" (or *rule templates*). This method was applied in the development of an intelligent failure-tolerant control system for a helicopter. 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 Monte Carlo simulations of the controlled system with alternate failure scenarios. Learned parameter values then can be defined as "fuzzy functions" [76] contained in rule premises. For example, a typical rule reads, "IF Indicator 1 is near A and Indicator 2 is near B, THEN there is a good chance that Forward Collective Pitch Control is biased from nominal by an amount near C and that Failure Detection Delay is near D."

The second approach is to *construct classification or decision trees* that relate attributes in the data to decision classes [52]. The problem is to develop an Expert Navigation Sensor Management System (NSM) that selects the best navigation aids from available measurements. Several aircraft paths were simulated, and the corresponding measurements that would have been made by GPS, Loran, Tacan, VOR, DME, Doppler radar, air data, and inertial sensors were calculated, with representative noise added. The simulated measurements were processed by extended Kalman filters to obtain optimal state estimates in 200 simulations. Using the root-sum-square error as a decision metric, Analysis of Variance (ANOVA) identifies the factors that make statistically significant contributions to the decision metric, and the Iterative Dichotomizer #3 (ID3) algorithm [77]–[79] extracts a decision tree from the training set by inductive inference. The ID3 algorithm quantifies the *entropy content* of each attribute, that is, the information gained by testing the attribute at a given decision node. It then defines a splitting strategy that minimizes the number of nodes required to characterize the tree.

Over 900 examples were used to develop the NSM decision tree, and performance was assessed at nearly 500 points on two trajectories that differed from the training set. NSM correctly assessed the error class for each nav-aid type (± 1 error class out of 23 possible ranges of RSS error) most of the time (see Fig. 5 of [52]). Differences between NSM and optimal navigation solutions were found to be minimal.

Procedural Systems

Most guidance, navigation, and control systems fielded to date are procedural systems using sequential algorithms on sequential processors. Although optimality of a cost function is not always a necessary or even sufficient condition for a "good" system, linear-optimal stochastic controllers provide a good generic structure for discussing procedural control. They are presented in state-space form, they contain separate control and estimation functions, and they provide an unusual degree of design flexibility. The optimal regulator effectively produces an approximate stable inverse in providing desired response. The nonlinear inverse dynamic controller is a suitable design alternative in some cases.

Control and Estimation

We assume that a nominal (desired) flight path is generated by higher level intelligence, such as the human pilot or declarative machine logic, or as a stored series of waypoints. The procedural system must follow the path, $\boldsymbol{x}^*(t)$

in $t_o < t < t_f$. Control is exercised by a digital computer at time intervals of Δt. The n-dimensional state vector perturbation at time t_k is $\boldsymbol{x}_k$, and the m-dimensional control vector perturbation is $\boldsymbol{u}_k$. The discrete time linear quadratic Gaussian (LQG) control law is formed as [63],

$$\boldsymbol{u}_k = \boldsymbol{u}_k^* - \boldsymbol{C}_B[\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k^*] = \boldsymbol{C}_F \boldsymbol{y}_k^* - \boldsymbol{C}_B \hat{\boldsymbol{x}}_k \quad (1)$$

$\boldsymbol{y}_k^*$ is the desired value of an output vector ($\triangleq \boldsymbol{H}_x \boldsymbol{x}_k + \boldsymbol{H}_u \boldsymbol{x}_k$), and $\hat{\boldsymbol{x}}_k$ is the *Kalman filter* estimate, expressed in two steps:

$$\hat{\boldsymbol{x}}_k(-) = \boldsymbol{\Phi}\hat{\boldsymbol{x}}_{k-1}(+) + \boldsymbol{\Gamma}\boldsymbol{u}_{k-1}$$
$$\hat{\boldsymbol{x}}_k \triangleq \hat{\boldsymbol{x}}_k(+) = \hat{\boldsymbol{x}}_k(-) + \boldsymbol{K}[\hat{\boldsymbol{z}}_k - \boldsymbol{H}_{\text{obs}}\hat{\boldsymbol{x}}_k(-)] \quad (2)$$

the forward and feedback control gain matrices are $\boldsymbol{C}_F$ and $\boldsymbol{C}_B$. Φ and Γ are state-transition and control-effect matrices that describe the aircraft's assumed dynamics. The estimator gain matrix is $\boldsymbol{K}$ and the measurement vector, $\boldsymbol{z}_k$, is a transformation of the state through $\boldsymbol{H}_{\text{obs}}$. The gains $\boldsymbol{C}_B$ and $\boldsymbol{K}$ result from solving two Riccati equations that introduce tradeoffs between control use and state perturbation and between the strengths of random disturbances and measurement error. $\boldsymbol{C}_F$, which provides proper steady state command response, is an algebraic function of $\boldsymbol{C}_B$, Φ, Γ, and $\boldsymbol{H}_{\text{obs}}$. All of the matrices may vary in time, and it may be necessary to compute $\boldsymbol{K}$ on-line. In the remainder, it is not essential that $\boldsymbol{C}_B$ and $\boldsymbol{K}$ be optimal (i.e., they may have been derived from eigen-structure assignment, loop sharing, etc.), although the LQR gains guarantee useful properties of the nominal closed-loop system [63].

The control structure provided by (1) and (2) is flexible. It can represent a scalar feedback loop if $\boldsymbol{z}$ contains one measurement and $\boldsymbol{u}$ one control; or, it can address measurement and control redundancy with $\boldsymbol{z}$ and $\boldsymbol{u}$ dimensions that exceed the dimension of the state $\boldsymbol{x}$. As an alternative, reduced-order modeling can be incorporated in the estimator. Assuming that Φ and Γ have the same dimensions as the aircraft's dynamic model ($n \times n$ and $n \times m$), *the baseline estimator introduces* n*th-order compensation in the feedback control loop*.

The weights of the quadratic control cost function can be chosen not only to penalize state and control perturbations, but to produce *output weighting*, *state rate weighting*, and *implicit model following*, all without modifying the dynamic model [63]. *Integral compensation*, *low-pass filter compensation*, and *explicit model following* can be obtained by augmenting the system model during the design process, increasing the compensation order, and producing the control structures shown in Fig. 8.

These cost weighting and compensation features can be used together, as in the proportional integral/implicit-model-following controllers developed in [80]. Implicit model following is especially valuable when an ideal model can be specified (as identified in flying qualities specifications and standards [5], [6]), and integral compensation provides automatic "trimming" (control that synthesizes $\boldsymbol{u}_k^*$ corresponding to $\boldsymbol{x}_k^*$ to achieve zero steady state command error) and low-frequency robustness. Combining integral and filter compensation produces controllers with good command tracking performance and smooth control actions, as demonstrated in flight tests [81]–[83]. The LQG regulator naturally introduces an *internal model of the controlled plant*, a feature that facilitates control design [84].

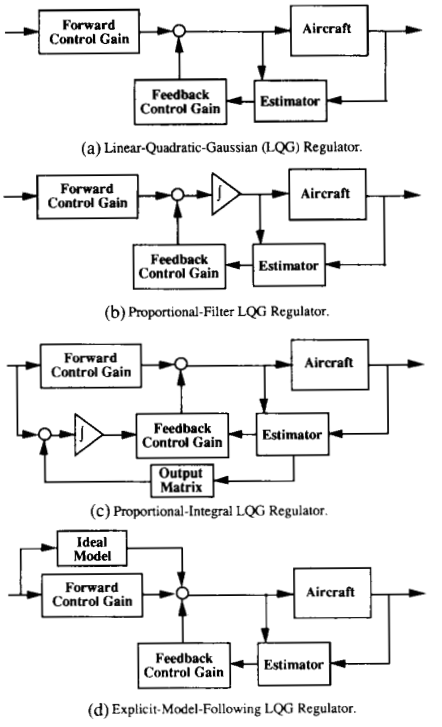


Fig. 8. Structured linear-quadratic-gaussian regulators.

The estimator in the feedback loop presents an efficient means of dealing with uncertainty in the measurements, in the disturbance inputs, and (to a degree) in the aircraft's dynamic model. If measurements are very noisy, the estimator gain matrix $\boldsymbol{K}$ is "small," so that the filter relies on extrapolation of the system model to estimate the state. If disturbances are large, the state itself is more uncertain, and $\boldsymbol{K}$ is "large," putting more emphasis on the measurements. Effects of uncertain parameters can be approximated as "process noise" that increases the importance of measurements, leading to a higher $\boldsymbol{K}$. If the system uncertainties are constant but unknown biases or scale factors, a better approach is to augment the filter state to estimate these terms directly. Parametric uncertainty introduces introduces nonlinearity; hence, an *extended Kalman filter* must be used [63].

Stability and Performance Robustness

Controlled system *robustness* is the ability to maintain satisfactory stability and performance in the presence of parametric or structural uncertainties in either the aircraft or its control system. All controlled systems must possess some degree of robustness against operational parameter

variations. The inherent stability margins of certain algebraic control laws (e.g., the linear-quadratic (LQ) regulator [63], [85]–[87]) may become vanishingly small when dynamic compensation (e.g., the estimator in a linear-quadratic-Gaussian (LQG) regulator) is added [88]. Restoring the robustness to that of the LQ regulator typically requires increasing estimator gains (within practical limits) using the loop transfer recovery method [89].

Subjective judgments must 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 [90]. The designer must know the normal operating ranges and distributions of parameter variations, as well as the specification for system operability with failed components. Otherwise, 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 [91], [92]; gain and phase margins are 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 (e.g., [93]). A 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 system.

Statistical measures of robustness can use knowledge of potential variations in real parameters. The *probability of instability* was introduced in [94] and is further described in [95], [96]. The *stochastic robustness* of a linear, time-invariant system, is judged using Monte Carlo simulation to estimate the probability distributions of closed-loop eigenvalues, given the statistics of the variable parameters in the system's dynamic model. The probability that one or more of these eigenvalues have positive real parts is the scalar measure of robustness, a figure of merit to be minimized by control system design. Because this metric can take one of only two values, it has a *binomial distribution*; hence, *the confidence intervals associated with estimating the metric from simulation are independent of the number or nature of the uncertain parameters* [95].

Considerations of performance robustness are easily taken into account in *Stochastic Robustness Analysis* (SRA). Systems designed using a variety of robust control methods (loop transfer recovery, H_∞ optimization, structured covariance, and game theory) are analyzed in [97], with attention directed to the probability of instability, probability of settling time exceedence, probability of excess control usage, and tradeoffs between them. The analysis uncovers a wide range of system responses and graphically illustrates that gain and phase margins are not good indicators of the probability of instability.[7] Incorporating SRA into the design of an LQG regulator with implicit model following and filter compensation leads to designs that have high levels of stability and performance robustness [98].

[7]Real parameter variations affect not only the magnitude and relative phase angle of the system's Nyquist contour but its *shape* as well [63]. Therefore, the points along the contour that establish gain and phase margin (i.e., the corresponding Bode plot frequencies) are subject to change.

Adaptation and Tolerance to Failures

Adaptation always has been a critical element of stability augmentation. Most aircraft requiring improved stability undergo large variations in dynamic characteristics on a typical mission. Gain scheduling and control interconnects initially were implemented mechanically, pneumatically, and hydraulically; now the intelligent part is done within a computer, and there is increased freedom to use sophisticated scheduling techniques that approach full nonlinear control [81], [99]. It becomes feasible not only to schedule according to flight condition but to account for differences in individual aircraft. Flight control systems that adapt to changes due to wear and exposure, and that report changes for possible maintenance action, can now be built.

Tolerance to system failures, such as plant alterations, actuator and sensor failures, computer failure, and power supply/transmission failure, is an important issue. Multiple failures can occur, particularly as a consequence of physical damage, and they may be intermittent. Factors that must be considered in designing failure-tolerant controls include: 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, system architecture, limits of manual intervention, and life cycle costs [43].

One approach to failure tolerance is *parallel redundancy*: two or more control strings, each separately capable of satisfactory control, are implemented in parallel. A *voting* scheme is used for redundancy management. 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. 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.

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. The principal functions of analytical redundancy are *failure detection*, *failure identification*, and *control system reconfiguration*. These functions use 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 [47].

Procedural adaptation and failure tolerance features will evolve outward, to become more declarative in their supervision and more reflexive in their implementation. Declarative functions are especially important for differentiating between normal and emergency control functions and sensitivities. They can work to reduce trim drag, to increase fatigue life, and to improve handling and ride qualities as functions of turbulence level, passenger loading, and so on. Gain scheduling control can be viewed as *fuzzy control*, suggesting that the latter has a role to play in aircraft control systems [100]–[102]. Reflexive functions can be added by computational neural networks that approximate nonlinear multivariate functions or classify failures.

Nonlinear Control

Aircraft dynamics are inherently nonlinear, but aerodynamic nonlinearities and inertial coupling effects generally are smooth enough in the principal operating regions to allow linear control design techniques to be used. Control actuators impose hard constraints on operation because their displacements and rates are strictly limited. Nonlinear control laws can improve control precision and widen stability boundaries when flight must be conducted at high angles or high angular rates and when the control actuator limits must be challenged.

The general principles of nonlinear inverse control are straightforward [103]. Given a nonlinear system of the form,

$$\dot{x} = f(x) + G(x)u \tag{3}$$

where $G(x)$ is square ($m = n$) and nonsingular, the control law

$$u = -G^{-1}f(x) + G^{-1}v \tag{4}$$

inverts the system, since

$$\dot{x} = f(x) + G(x)[-G^{-1}f(x) + G^{-1}v] = v \tag{5}$$

where v is the command input to the system.

In general, $G(x)$ is not square ($m \neq n$); however, given an m-dimensional output vector,

$$y \triangleq Hx \tag{6}$$

it is possible to define a nonlinear feedback control law that produces *output decoupling* of the elements of y or their derivatives such that $y^{(d)} = v$. The vector $y^{(d)}$ contains *Lie derivatives* of y,

$$y^{(d)} = = f^*(x) + G^*(x)u \tag{7}$$

where d is the *relative degree* of differentiation required to identify a direct control effect on each element of y. $G^*(x)$ and $f^*(x)$ are components of the Lie derivatives, and $G^*(x)$ is guaranteed to be structurally invertible by the condition that defines relative degree [104]. The decoupling control law then takes the form

$$u = -[G^*(x)]^{-1}f^*(x) + [G^*(x)]^{-1}v \tag{8}$$

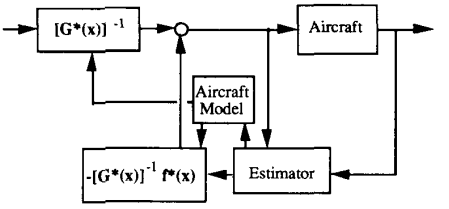


Fig. 9. Decoupling nonlinear inverse control law.

The control law is completed by feeding y back as appropriate to achieve desired transient response and by prefiltering v to produce the desired command response [105]. Because the full state is rarely measured and measurements can contain errors, it may be necessary to estimate x with an extended Kalman filter, substituting $\hat{x}$ for x in control computations.

Evaluating $G^*(x)$ and $f^*(x)$ requires that a full, d-differentiable model of aircraft dynamics be included in the control system; hence the statement of the control law is simple, but its implementation is complex (Fig. 9). Smooth interpolators of the aircraft model (e.g., cubic splines) are needed. Feedforward neural networks with sigmoidal activation functions provide a good means of representing this model, as they are infinitely differentiable and allow for adaptation [106], [107].

There are limitations to the inverse control approach [108]. The principal concerns are pointwise singularity of $G^*(x)$, the effects of control saturation, and the presence of the nonlinear equivalent of nonminimum phase (NMP) zeros in aircraft dynamics. The command vector (6) has a direct effect on the definition of $G^*(x)$. In [105], the singular points of $G^*(x)$ are found to be outside the flight envelope of the subject aircraft for all command vectors. When saturation of a control effector occurs, the control dimension must be reduced by one; hence, the command vector is redefined to exclude the least important element of y in [105]. The command vector is returned to original dimension when the control effector is no longer saturated.

Whether or not NMP zero effects are encountered depends on the command vector definition and on the physical model. Some command vector definitions for aircraft control produce no NMP zeros [105]. When NMP zeros occur in conventional aircraft models, they are due to small force effects (e.g., lift due to elevator deflection and pitch rate), and it may be possible to neglect them.

Reflexive Systems

Inner-loop control is a reflexive function. To date, most inner loops have been designed as procedural pointwise linear control structures. Computational neural networks may extend prior results to facilitate nonlinear control and adaptation. Neural networks can be viewed as *nonlinear generalizations of sensitivity, transformation, and gain matrices*. Consequently, compensation dynamics can be incorporated by following earlier models and control structures. Nonlinear proportional-integral and model-

following controllers, as well as nonlinear estimators, can be built using computational neural networks.

Computational Neural Networks

Computational neural networks are motivated by input–output and learning properties of biological neural systems, though in mathematical application the network becomes an abstraction that may bear little resemblance to its biological model. Computational neural networks consist of *nodes* that simulate the *neurons* and *weighting factors* that simulate the *synapses* of a living nervous system. The nodes are nonlinear basis functions, and the weights contain knowledge of the system. Neural networks are good candidates for performing a variety of reflexive functions in intelligent 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" may be softened to "saturation," not only to facilitate learning of the synaptic weights but to admit the modeling of continuous, differentiable functions.

The neural networks receiving most current attention are static expressions that perform one of two functions. The first is to *approximate multivariate functions* of the form

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) \tag{9}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are input and output vectors and $\boldsymbol{h}(\cdot)$ is the (possibly unknown) relationship between them. Neural networks can be considered *generalized spline functions* that identify efficient input–output mappings from observations [109], [110]. The second application is to *classify attributes*, much like the decision trees mentioned earlier. (In fact, decision trees can be mapped to neural networks [111].) The following discussion emphasizes the first of these two applications.

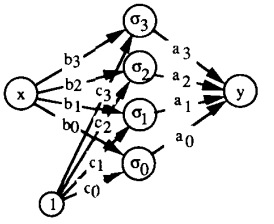
An N-layer *feedforward neural network* (FNN) represents the function by a sequence of operations,

$$\boldsymbol{r}^{(k)} = \boldsymbol{s}^{(k)}[\boldsymbol{W}^{(k-1)}\boldsymbol{r}^{(k-1)}] \triangleq \boldsymbol{s}^{(k)}[\eta^{(k)}], \qquad k = 1 \text{ to } N \tag{10}$$

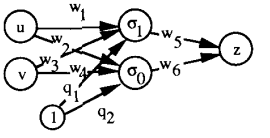
where $\boldsymbol{y} = \boldsymbol{r}^{(N)}$ and $\boldsymbol{x} = \boldsymbol{r}^{(0)}$. $\boldsymbol{W}^{(k-1)}$ is a matrix of weighting factors determined by the learning process, and $\boldsymbol{s}^{(k)}[\cdot]$ is an activation function vector whose elements normally are identical, scalar, nonlinear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

$$\boldsymbol{s}^{(k)}[\eta^{(k)}] = [\sigma_1(\eta_1{}^{(k)}) \cdots \sigma_n(\eta_n^{(k)})]^T. \tag{11}$$

One of the inputs to each layer may be a unity threshold element that adjusts the bias of the layer's output. Networks consisting solely of linear activation functions are of little interest, as they merely perform a linear transformation $\boldsymbol{H}$, thus limiting (9) to the form, $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}$.



(a) Single-Input/Single-Output Network.



(b) Double-Input/Single-Output Network.

Fig. 10. Two feedforward neural networks.

Fig. 10 represents two simple feedforward neural networks. Each circle represents an arbitrary, scalar, nonlinear function $\sigma_i(\cdot)$ operating on the sum of its inputs, and each arrow transmits a signal from the previous node, multiplied by a weighting factor. A scalar network with a single hidden layer of four nodes and a unit threshold element [Fig. 10(a)] is clearly parallel, yet its output can be written as the series

$$y = a_0\sigma_0(b_0x + c_0) + a_1\sigma_1(b_1x + c_1) + a_2\sigma_2(b_2x + c_2) + a_3\sigma_3(b_3x + c_3) \tag{12}$$

illustrating that parallel and serial processing may be equivalent.

Consider a simple example. Various nodel activation functions, σ_i, have been used, and there is no need for each node to be identical. Choosing $\sigma_0(\cdot) = (\cdot)$, $\sigma_1 = (\cdot)^2$, $\sigma_2 = (\cdot)^3$, $\sigma_3 = (\cdot)^4$, (9) is represented by the truncated power series,

$$y = a_0(b_0x + c_0) + a_1(b_1x + c_1)^2 + a_2(b_2x + c_2)^3 + a_3(b_3x + c_3)^4. \tag{13}$$

It is clear that network weights are redundant (i.e., that the (a, b, c) weighting factors are not independent). Consequently, more than one set of weights could produce the same functional relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$. Training sessions starting at different points could produce different sets of weights that yield identical outputs. This simple example also indicates that the unstructured feedforward network may not have compact support (i.e., its weights may have global effects) if its basis functions do not vanish for large magnitudes of their arguments.

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 ($N = 2$) [112], [113]. Symmetric functions like the *Gaussian radial basis*

function ($\sigma(\eta) = e^{-\eta^2}$) have better convergence properties for many functions and have more compact support as a consequence of near orthogonality [109], [114]. Classical *B-splines* [115] could be expressed in parallel form, and it has been suggested that they be used in multilayered networks [116]. Adding hidden layers strengthens the analogy to biological models, though additional layers are not necessary for approximating continuous functions, and they complicate the training process.

In control application, neural networks perform functions analogous to gain scheduling or nonlinear control. Consider the simple two-input network of Fig. 10(b). The scalar output and derivative of a single sigmoid with unit weights are shown in Fig. 11. If u is a fast variable and v is a slow variable, choosing the proper weights on the inputs and threshold can produce a gain schedule that is approximately linear in one region and nonlinear (with an inflection point) in another. More complex surfaces can be generated by increasing the number of sigmoids. If u and v are both fast variables, then the sigmoid can represent a generalization of their nonlinear interaction.

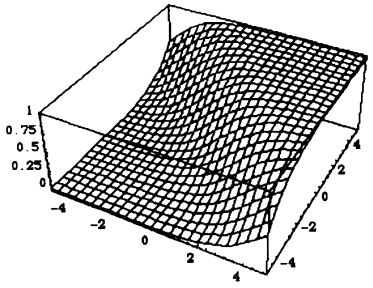
For comparison, a typical radial basis function produces the output shown in Fig. 12. Whereas the sigmoid has a preferred input axis and simple curvature, the RBF admits more complex curvature of the output surface, and its effect is more localized. The most efficient nodal activation function depends on the general shape of the surface to be approximated. There may be cases best handled by a mix of sigmoids and RBF in the same network.

The cerebellar model articulation controller (CMAC) is an alternate network formulation with somewhat different properties but good promise for application in control systems [117], [118]. The CMAC performs table lookup of a nonlinear function over a particular region of function space. CMAC operation can be split into two mappings. The first maps each input into an *association space* $\mathfrak{A}$. The mapping generates a *selector vector* $\boldsymbol{a}$ of dimension n_A, with c nonzero elements (usually ones) from overlapping *receptive regions* for the input. The second mapping, $\mathfrak{R}$, goes from the selector vector $\boldsymbol{a}$ to the scalar output y through the weight vector $\boldsymbol{w}$, which is derived from training:

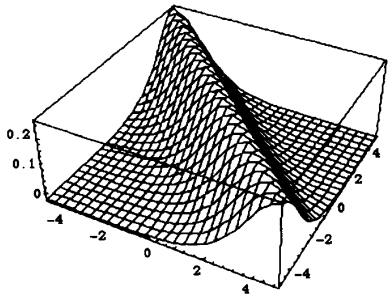
$$y = \boldsymbol{w}^T\boldsymbol{a}. \tag{14}$$

Training is inherently local, as the extent of the receptive regions is fixed. The CMAC has quantized output, producing "staircased" rather than continuous output. A recent paper proposes to smooth the output using B-spline receptive regions [119].

The FNN and CMAC are both examples of *static networks*, that is, their outputs are essentially instantaneous: given an input, the speed of output depends only on the speed of the computer. *Dynamic networks* rely on stable resonance of the network about an equilibrium condition to relate a fixed set of initial conditions to a steady-state output. Bidirectional Associative Memory (BAM) networks [120] are nonlinear dynamical systems that subsume Hopfield networks [121], Adaptive Resonance Theory (ART) networks [122], and Kohonen networks [123]. Like FNN, they use binary or sigmoidal neurons and store knowledge in the weights that connect them; however, the

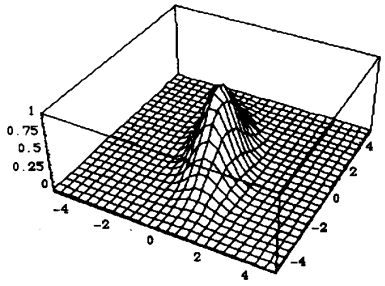


(a) Sigmoid.

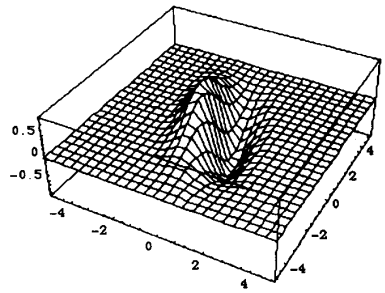


(b) x-Derivative of Sigmoid.

Fig. 11. Example of sigmoid output with two inputs.



(a) Radial Basis Function (RBF).



(b) x-Derivative of RBF.

Fig. 12. Example of radial basis function output with two inputs.

"neural circuits" take time to stabilize on an output. While dynamic networks may operate more like biological neurons, which have a *refractory period* between differing outputs, computational delay degrades aircraft control functions.

Although neural networks performing function approximation may gain little from multiple hidden layers, networks used for classification typically require multiple layers, as follows from the ability to map decision trees to neural networks [111]. The principal values of performing such a mapping are that it identifies an efficient structure for parallel computation, and it may facilitate incremental learning and generalization.

Neural networks can be applied to *failure detection and identification* (FDI) by mapping data patterns (or *feature vectors*) associated with failures onto detector/identification vectors (e.g., [124]–[126]). To detect failure, the output is a scalar, and the network is trained (for example) with "1" corresponding to failure and "0" corresponding to no failure. To identify specific failures, the output is a vector, with a training value of "1" in the ith element corresponding to the ith failure mode and zeros elsewhere. 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*, preprocessing that transforms the input time series into a feature vector. In [124], this was done by computing two dozen Fourier coefficients of the input signal in a moving temporal window. As an alternative, the feature vector could be specified as a *parity vector* [127], and the neural network could be used for the decision making logic in FDI. When assessing the efficiency of neural network FDI logic, feature extraction must be considered part of the total process.

Reflexive Learning and Adaptation

Training neural networks involves either supervised or unsupervised learning. In *supervised learning*, the network is furnished typical histories of inputs and outputs, and the training algorithm computes the weights that minimize fit error. FNN and CMAC require this type of training. In *unsupervised learning*, the internal dynamics are self-organizing, tuning the network to home on different cells of the output *semantic map* in response to differing input patterns [128]. Dynamic networks learn rapidly and are suitable for pattern matching, as in speech or character recognition. The remaining discussion focuses on supervised learning, which is more consistent with control functions.

Backpropagation learning algorithms for the elements of $\boldsymbol{W}^{(k)}$ typically involve a *gradient search* (e.g., [129], [130]) that minimizes the mean-square output error

$$\mathcal{E} = [\boldsymbol{r}_d - \boldsymbol{r}^{(N)}]^T[\boldsymbol{r}_d - \boldsymbol{r}^{(N)}] \tag{15}$$

where $\boldsymbol{r}_d$ is the desired output. The error gradient with respect to the weight matrix is calculated for each input–output example presented to the network, and the weights are updated by

$$\boldsymbol{W}^{(k)}_{\text{new}} = \boldsymbol{W}^{(k)}_{\text{old}} + \beta \boldsymbol{r}^{(k-1)}[\boldsymbol{d}^{(k)}]^T \tag{16}$$

β is the learning rate, and $\boldsymbol{d}$ is a function of the error between desired and actual outputs. For the output layer, the error term is

$$\boldsymbol{d}^{(N)} = \boldsymbol{S}'[\boldsymbol{W}^{(N-1)}\boldsymbol{r}^{(N-1)}]\, \boldsymbol{r}_d - \boldsymbol{r}^{(N)} \tag{17}$$

where the prime indicates differentiation with respect to $\boldsymbol{r}$. For interior layers, the error from the output layer is propagated from the output error using

$$\boldsymbol{d}^{(k)} = \boldsymbol{S}'[\boldsymbol{W}^{(k-1)}\boldsymbol{r}^{(k-1)}]\, [\boldsymbol{W}^{(k-1)}]^T\boldsymbol{d}^{(k-1)} \tag{18}$$

Search rate can be modified by adding momentum or conjugate-gradient terms to (16).

The CMAC network learning algorithm is similar to backpropagation. The weights and output are connected by a simple linear operation, so a learning algorithm is easy to prescribe. Each weight contributing to a particular output value is adjusted by a fraction of the difference between the network output and the desired output. The fraction is determined by the desired learning speed and the number of receptive regions contributing to the output.

Learning speed and accuracy for FNN can be further improved using an *extended Kalman filter* [106], [107], [131]. The dynamic and observation models for the filter are

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \boldsymbol{q}_{k-1} \tag{19}$$

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{w}_k, \boldsymbol{r}_k) + \boldsymbol{n}_k \tag{20}$$

where $\boldsymbol{w}_k$ is a vector of the matrix $\boldsymbol{W}_k$'s elements, $\boldsymbol{h}(\cdot)$ is an observation function, and $\boldsymbol{q}_k$ and $\boldsymbol{n}_k$ are noise processes. If the network has a scalar output, then z_k is scalar, and the extended Kalman filter minimizes the fit error between the training hypersurface and that produced by the network (15). The fit error can be dramatically reduced by considering the *gradients* of the surfaces as well [106], [107]. The observation vector becomes

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{w}_k, \boldsymbol{r}_k) \\ \dfrac{\partial \boldsymbol{h}}{\partial \boldsymbol{r}}(\boldsymbol{w}_k, \boldsymbol{r}_k) \end{bmatrix} + \boldsymbol{n}_k \tag{21}$$

with concomitant increase in the complexity of the filter. The relative significance given to function and derivative error during training can be adjusted through the measurement-error covariance matrix used in filter design.

Recursive estimation of the weights is useful when smooth relationships between fit errors and the weights are expected, when the weight-vector dimension is not high, and when local minima are global. When one of these is not true, it may speed the computation of weights to use a *random search*, at least until convergent regions are identified. Such methods as *simulated annealing* or *genetic algorithms* can be considered (and the latter has

philosophic appeal for intelligent systems) [132]–[134]. The first of these is motivated by statistical mechanics and the effects that controlled cooling has on the ground states of atoms (which are analogous to the network weights). The second models the reproduction, crossover, and mutation of biological strings (e.g., chromosomes, again analogous to the weights), in which only the fittest combinations survive.

Statistical methods can go hand in hand with SRA to train *robust neural networks*. Following [98], the random search could be combined with Monte Carlo variation of system parameters during training, numerically minimizing the *expected value of fit error* rather than a deterministic fit error.

Problems that may be encountered in neural network training include proper choice of the input vector, local vs. global training, speed of learning and forgetting, generalization over untrained regions, and trajectory-dependent correlations in the training sets. We envision an aerodynamic model that spans the entire flight envelope of an aircraft, including poststall and spinning regions. The model contains six neural networks with multiple inputs and scalar outputs, three for force coefficients and three for moment coefficients (for example, the pitch moment network takes the form $C_m = g(\boldsymbol{x}, \boldsymbol{u})$, where $\boldsymbol{x}$ represents the state and $\boldsymbol{u}$ the control). If input variables are not restricted to those having plausible aerodynamic effect, false correlations may be created in the network; hence, attitude Euler angles and horizontal position should be neglected, while physically meaningful terms like elevator deflection, angle of attack, pitch rate, Mach number, and dynamic pressure should be included [107].

The aircraft spends most of its time flying within normal mission envelopes. Unless it is a trainer, the aircraft does not normally enter poststall and spinning regions; consequently, on-line network training focuses on normal flight and neglects extreme conditions. This implies not only that networks must be pretrained in the latter regions but that normal training must not destroy knowledge in extreme regions (which may be required in an emergency) while improving knowledge in normal regions. Therefore, radial basis functions appear to be a better choice than sigmoid activation functions for adaptive networks.

Elements of the input vector may be strongly correlated with each other through the aircraft's equations of motion; hence, networks may not be able to distinguish between highly correlated variables (e.g., pitch rate and normal acceleration). This is problematical only when the aircraft is outside its normal envelope. Pretraining should provide inputs that are rich in frequency content, that span the state and control spaces, and that are as uncorrelated as possible. Generalization between training points may provide smoothness, but it does not guarantee accuracy.

Control Systems Based on Neural Networks

Neural networks can find application in logic for control, estimation, system identification, and physical modeling. In addition to work already referenced, additional examples can be found in [135]–[140]. Fig. 13(a) illustrates an application in which the neural network forms the aircraft model for a nonlinear inverse control law. The aircraft model of Fig. 9 is implemented with a neural network that is trained by a dedicated (weight) extended Kalman filter (the thick gray arrow indicating training). The extended Kalman filter for state estimation is expanded to estimate histories of forces and moments as well as the usual motion variables.

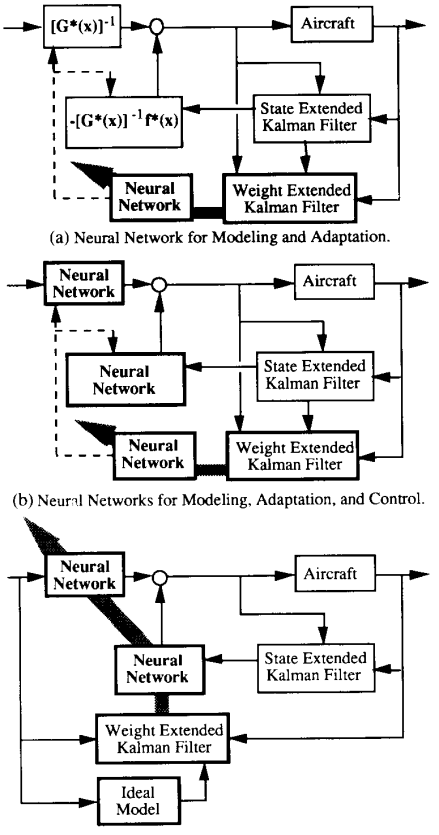


Fig. 13. Adaptive control structures using neural networks.

It is possible to conduct supervised learning on-line without interfering with normal operation because the state Kalman filter produces both the necessary inputs and the desired outputs for the network training algorithm. There is no need to provide an ideal control response for training, as the form of the control law is fixed. Procedural and reflexive functions are combined in this control implementation, under the assumption that the direct expression of inversion is the most efficient approach.

Fig. 13(b) shows a logical extension in which the inverse control law is implemented by neural networks. Inversion is an implicit goal of neural network controllers [135], [136], and the formal existence of inversion networks has been explored [141]. Although Fig. 13(b) implies that the inversion networks are pretrained and fixed,

they, too, can be trained with the explicit help of the network that models the system [136]. Here, it is assumed that the control networks have been pretrained, as no desired output has been specified.

If a desired control output is specified [Fig. 13(c)], then the formal model of the aircraft is no longer needed. The control networks learn implicit knowledge of the aircraft model. Referring to Fig. 8 and (1) and (2), control and estimation gains, state transition and control-effect matrices, and measurement transformations can be implemented as static neural networks with either off-line or on-line learning.

It can be useful to divide control networks into separate feedback and forward parts, as this may facilitate training to achieve design goals. A feedback neural network has strongest effect on homogeneous modes of motion, while a forward neural network is most effective for shaping command (forced) response. This structure is adopted in [139], where the forward and feedback networks are identified as *reason* and *instinct networks*. In pretraining, it is plausible that the feedback network would be trained with initial condition responses first, to obtain satisfactory transient response. The forward network would be trained next to achieve desired steady states and input decoupling. A third training step could be the addition of command-error integrals while focusing on disturbance inputs and parameter variations in training sets.

Once baselines have been achieved, it could prove useful to admit limited coupling between forward and feedback networks to enable additional nonlinear compensation. In adaptive applications, networks would be pretrained with the best available models and scenarios to establish satisfactory baselines; on-line training would slowly adjust individual systems to vehicle and mission characteristics.

Including the integral of command-vector error as a neural network input produces a *proportional-integral* structure [140], while placing the integrator beyond the network gives a *proportional filter* structure (Fig. 8). The principal purpose of these structures is, as before, to assure good low and high-frequency performance in a classical sense. Extension of neural networks to state and weight filters is a logical next step that is interesting in its own right as a means of more nearly optimal nonlinear estimation.

Conclusion

Intelligent flight control systems can do much to improve the operating characteristics of aircraft, and they provide an interesting, practical, and well documented framework within which intelligent control concepts can be developed. An examination of cognitive and biological models for human control of systems suggests that they exhibit a declarative, procedural, and reflexive hierarchy of functions; this structure can be applied to aircraft control. Top level aircraft control functions are analogous to conscious and preconscious thoughts that are transmitted to lower level subsystems through subconscious, neural, and reflexlike activities. Human cognition and biology also suggest models for automated learning and adaptation, not only during operation but between periods of activity.

The computational analogs of the three cognitive/biological paradigms are expert systems, stochastic controllers, and neural networks. Expert systems organize decision making efficiently, stochastic controllers optimize estimation and control, and neural networks provide rapid, nonlinear, input-output functions. It appears that many control functions at *all* levels could be implemented as neural networks. While this may not always be necessary or even desirable using sequential processors, mapping declarative and procedural functions as neural networks may prove useful as a route to new algorithms for the massively parallel flight control processors of the future.

References

[1] *The Papers of Wilbur and Orville Wright*, vol. 1, pp. 1898-1905; vol. 2, pp. 1906-1948.

[2] R. P. Harper, Jr. and G. E. Cooper, "Handling qualities and pilot evaluation," *J. Guid., Cont., Dyn.*, vol. 9, no. 5, pp. 515-529, Sept.-Oct. 1986.

[3] D. J. Moorhouse, "The history and future of U.S. military flying qualities specification," *AIAA Aero. Sci. Mtg.*, New Orleans, Jan. 1979.

[4] R. O. Anderson, "Flying qualities yesterday today and tomorrow," *AIAA Atmos. Flgt. Mech. Conf.*, Danvers, MA, Aug. 1980.

[5] ——, "Military Specification, Flying Qualities of Piloted Airplanes," MIL-F-8785C, WPAFB, OH, Nov. 1980.

[6] R. H. Hoh, *et al.*, *Proposed MIL Standard and Handbook—Flying Qualities of Air Vehicles*, AFWAL-TR-82-3081, WPAFB, OH, Nov. 1982.

[7] A. Tustin, "The nature of the operator's response in manual control and its implications for controller design," *Proc. IEE*, vol. 94, part IIA, pp. 190-202, 1947.

[8] D. T. McRuer, "Development of pilot-in-the-loop analysis," *J. Aircraft*, vol. 10, no. 9, pp. 515-524, Sept. 1973.

[9] D. L. Kleinman, S. Baron, and W. Levison, "An Optimal Control Model of Human Response," *Automatica*, vol. 9, no. 3, pp. 357-383, May 1970.

[10] R. F. Stengel and J. R. Broussard, "Prediction of pilot-aircraft stability boundaries and performance contours," *IEEE Trans. Syst., Man, Cyber.*, vol. SMC-8, no. 5, pp. 349-356, May 1978.

[11] D. T. McRuer, *et al.*, "New approaches to human pilot/vehicle dynamic analysis," AFFDL-TR-67-150, WPAFB, OH, Feb. 1968.

[12] L. Stark, *Neurological Control Systems*. New York: Plenum Press, 1968.

[13] J. R. Stone and G. J. Gerken, "The prediction of pilot acceptance for a large aircraft," *Proc. 1973 Joint Auto. Cont. Conf.*, Columbus, OH, pp. 807-809, June 1973.

[14] W. M. Hollister, Ed., *Improved Guidance and Control Automation at the Man-Machine Interface*, AGARD-AR-228, Neuilly-sur-Seine, Dec. 1986.

[15] B. O. Hartman, ed., *Higher Mental Functioning in Operational Environments*, AGARD-CP-181, Neuilly-sur-Seine, Apr. 1986.

[16] R. M. Patton, T. A. Tanner, Jr., and J. A. Swets, *Applications of Research on Human Decisionmaking*, NASA SP-209, Washington, DC, 1970.

[17] E. L. Wiener and D. C. Nagel, *Human Factors in Aviation*, Academic Press, San Diego, 1988.

[18] C. D. Perkins, "Development of airplane stability and control technology," *J. Aircraft*, vol. 7, no. 4, pp. 290-301, Jul.-Aug. 1970.

[19] D. McRuer, I. Ashkenas, and D. Graham, *Aircraft Dynamics and Automatic Control*. Princeton, NJ: Princeton Univ. Press, 1973.

[20] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*, J. Wiley & Son, New York, 1991.

[21] A. E. Bryson, Jr., M. N. Desai, and W. L. Hoffman, "The energy

state approximation in performance optimization of supersonic aircraft," *J. Aircraft*, vol. 6, no. 6, pp. 481-487, Nov.-Dec. 1969.
[22] H. Erzberger, J. D. McLean, and J. F. Barman, "Fixed-range optimum trajectories for short haul aircraft," NASA TN D-8115, Washington, DC, Dec. 1975.
[23] R. F. Stengel and F. J. Marcus, "Energy management for fuel conservation in transport aircraft," *IEEE Trans. Aero. Elec. Sys.*, vol. AES-12, no. 4, pp. 464-470, July 1976.
[24] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA, New York, 1987.
[25] W. S. Widnall, "Lunar module digital autopilot," *J. Space Rockets*, vol. 8, no. 1, Jan. 1971.
[26] R. F. Stengel, "Manual attitude control of the lunar module," *J. Space Rockets*, vol. 7, pp. 56-62, no. 8, Aug. 1970, pp. 941-948.
[27] A. A. Lambregts, "Integrated system design for flight and propulsion control using total energies principles," AIAA Paper No. 83-2561, New York, Oct. 1983.
[28] S. M. Wagner and S. W. Rothstein, "Integrated control and avionics for air superiority: Computational aspects of real-time flight management," *AIAA Guid., Nav., Cont. Conf.*, Boston, pp. 321-326, Aug. 1989.
[29] W. H. Harris and J. S. Levey, eds., *New Columbia Desk Encyclopedia*. New York: Columbia Univ. Press, 1975.
[30] R. J. Sternberg, "Human Intelligence: The model is the message," *Science*, vol. 230, pp. 1111-1118, Dec. 6, 1985.
[31] J. R. Searle, "Is the brain's mind a computer program?" *Scient. Amer.*, vol. 262, no. 1, pp. 26-31, Jan. 1990.
[32] R. Penrose, *The Emperor's New Mind*. New York: Penguin, 1989.
[33] G. Kolata, "Does Gödel's theorem matter to mathematics?" *Science*, vol. 218, pp. 779-780, Nov. 19, 1982.
[34] P. M. Churchland and P. S. Churchland, "Could a machine think?" *Scient. Amer.*, vol. 262, no. 1, pp. 32-37, Jan. 1990.
[35] M. Minsky, "Re: penrose," NetNews communication, May 13, 1992.
[36] A. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433-460, Oct. 1950.
[37] J. F. Kihlstrom, "The cognitive unconscious," *Science*, vol. 237, pp. 1445-1452, Sept. 18, 1987.
[38] J. R. Anderson, *Language, Memory, and Thought*, Erlbaum, Hillsdale, NJ, 1976.
[39] D. E. Hinto and J. A. Anderson, *Parallel Models of Associative Memory*. Hillsdale, NJ: Erlbaum, 1981.
[40] J. Mitchell, Ed., *The Random House Encyclopedia*, Random House, New York, 1990.
[41] E. Tulving and D. L. Schacter, "Priming and human memory systems," *Science*, vol. 247, pp. 301-306, Jan. 10, 1990.
[42] S. Blakeslee, "Scientists unraveling chemistry of dreams," *The New York Times*, pp. C1, C10, Jan. 7, 1992.
[43] R. F. Stengel, "Intelligent failure-tolerant control," *IEEE Cont. Sys. Mag.*, vol. 11, no. 4, pp. 14-23, June 1991.
[44] M. M Waldrop, "Causality, structure, and common sense," *Science*, vol. 237, pp. 1297-1299, Sept. 11, 1987.
[45] P. R. Cohen and E. A. Feigenbaum, *The Hand. Art. Intell.*, William Kaufmann, Los Altos, CA, 1982.
[46] E. Charniak and D. McDermott, *Intro. Art. Intell.*, Addison-Wesley, Reading, MA, 1985.
[47] D. A. Handelman and R. F. Stengel, "Combining expert system and analytical redundancy concepts for fault-tolerant flight control," *J. Guid., Cont., Dyn.*, vol. 12, no. 1, pp. 39-45, Jan.-Feb. 1989.
[48] D. A. Handelman and R. F. Stengel, "An architecture for real-time rule-based control," in *Proc. Amer. Cont. Conf.*, Minneapolis, NM, pp. 1636-1642, June 1987.
[49] C. Y. Huang and R. F. Stengel, "Failure model determination in a knowledge-based control system," in *Proc. Amer. Cont. Conf.*, Minneapolis, MN, pp. 1643-1648, June 1987.
[50] K. Frankovich, K. Pedersen, and S. Bernsteen, "Expert system Applications to the Cockpit of the '90s," *IEEE Aero. Elec. Sys. Mag.*, pp. 13-19, Jan. 1986.
[51] B. L. Belkin and R. F. Stengel, "Cooperative rule-based control systems for aircraft navigation and control," in *Proc. IEEE Conf. Dec. Cont.*, Los Angeles, pp. 1934-1940, Dec. 1987.
[52] B. L. Belkin and R. F. Stengel, "Systematic methods for knowledge acquisition and expert system development," *IEEE Aero. Elec. Sys. Mag.*, vol. 6, no. 6, pp. 3-11, June 1991.
[53] B. L. Belkin and R. F. Stengel, "AUTOCREW: A Paradigm for Intelligent Flight Control," *An Introduction to Intelligent and Autonomous Control*, P. Antsaklis and K. Passino, Ed., Kluwer, Norwell, MA, pp. 371-400, 1993.
[54] K. J. Maxwell and J. A. Davis, "Artificial intelligence implications for advanced pilot/vehicle interface design," AIAA 84-2617, New York, 1984.
[55] C. A. Leavitt and D. M. Smith, "Integrated dynamic planning in the pilot's associate," in *Proc. AIAA Guid., Nav., Cont. Conf.*, Boston, pp. 327-331, Aug. 1989.
[56] M. Broadwell, Jr. and D. M. Smith, "Associate systems technology issues," in *Proc. AIAA Guid., Nav., Cont. Conf.*, New Orleans, pp. 1451-1457, Aug. 1991.
[57] X. Miao, P. B. Luh, D. L. Kleinman, and D. A. Castanon, "Distributed stochastic resource allocation in teams," *IEEE Trans. Syst., Man, Cyber.*, vol. 21, no. 1, pp. 61-69, Jan.-Feb. 1991.
[58] P. Kapsouris, D. Serfaty, J. C. Deckert, J. G. Wohl, and K. R. Pattipati, "Resource allocation and performance evaluation in large human-machine organizations," *IEEE Trans. Syst., Man, Cyber.*, vol. 21, no. 3, pp. 521-531, May-June 1991.
[59] R. Mallubhatla, K. R. Pattipati, D. L. Kleinman, and Z. B. Tang, "A model of distributed team information processing under ambiguity," *IEEE Trans. Syst., Man, Cyber.*, vol. 21, no. 4, pp. 713-725, July-Aug. 1991.
[60] E. Wagner, "On-board automatic aid and advisory for pilots of control-impaired aircraft," in *Proc. AIAA Guid., Nav., Cont. Conf.*, Boston, pp. 306-320, Aug. 1989.
[61] B. M. Anderson, N. L. Cramer, M. Lineberry, G. S. Lystad, and R. C. Stern, "Intelligent automation of emergency procedures in advanced fighter aircraft," in *Proc. IEEE Conf. Art. Intell. App.*, Silver Spring, MD, pp. 496-501, Dec. 1984.
[61] S. Berning, D. P. Glasson, and J. L. Pomarede, "Knowledge engineering for the adaptive tactical navigator," *Proc. IEEE Nat. Aero. Elec. Conf.*, Dayton, pp. 1266-1273, May 1988.
[63] R. F. Stengel, *Stochastic Optimal Control: Theory and Applications*, Wiley, New York, 1986.
[64] K.-C. Ng and Abramson, B., "Uncertainty management in expert systems," *IEEE Expert*, vol. 5, no. 2, pp. 29-47, Apr. 1990.
[65] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Palo Alto, CA: Morgan Kaufmann, 1988.
[66] D. A. Stratton and R. F. Stengel, "Probabilistic reasoning for intelligent wind shear avoidance," *J. Guid., Cont., Dyn.*, vol. 15, no. 1, pp. 247-254, Jan.-Feb. 1992.
[67] D. A. Stratton and R. F. Stengel, "Real-time decision aiding: Aircraft guidance for wind shear avoidance," AIAA 92-0290, Reno, Jan. 1992.
[68] A. Niehaus and R. F. Stengel, "An expert system for automated highway driving," *IEEE Cont. Sys. Mag.*, vol. 11, no. 3, pp. 53-61, Apr. 1991.
[69] A. Niehaus and R. F. Stengel, "Rule-based guidance for vehicle highway driving in the presence of uncertainty," in *Proc. Amer. Cont. Conf.*, Boston, pp. 3119-3124, June 1991.
[70] A. Niehaus and R. F. Stengel, "Probability-based decision making for automated highway driving," in *Proc. Veh. Nav. Info. Sys. '91 Conf.*, SAE912869, Dearborn, MI, pp. 1125-1136, Oct. 1991.
[71] M. Perez, L. Gemoets, and R. G. McIntyre, "Knowledge extraction methods for the development of expert systems," *Knowledge Based System Applications for Guidance and Control*, AGARD-CP-474, pp. 26-1-26-10, Apr. 1991.
[72] P. M. Ryan and A. J. Wilkinson, "Knowledge acquisition for ATE diagnosis," *IEEE Aerosp. Elec. Syst., Mag.*, pp. 5-12, July 1986.
[73] S. M. Weiss and C. A. Kulikowski, *Computer Systems That Learn*. San Mateo, CA: Morgan Kaufmann, 1991.
[74] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cyber.*, vol. 21, no. 3, pp. 660-674, May-June 1991.
[75] D. A. Handelman and R. F. Stengel, "Rule-based mechanisms of learning for intelligent adaptive flight control," in *Proc. Amer. Cont. Conf.*, Atlanta, pp. 208-213, June 1988.
[76] R. Tong, "A control engineering review of fuzzy systems," *Automatica*, vol. 13, no. 6, pp. 559-569, Nov. 1977.
[77] J. R. Quinlan, "Discovering rules by induction from large collections of samples," in *Expert Systems in the Micro Electronic Age*, D. Michie, Ed. Edinburgh: Edinburgh University Press, pp. 169-201, 1979.
[78] B. Thompson and W. Thompson, "Finding rules in data," *Byte*, Nov. 1986, pp. 149-158.
[79] J. Durkin, "Induction· · ·," *AI Expert*, pp. 48-53, Apr. 1992.

[80] C. Y. Huang and R. F. Stengel, "Restructurable control using proportional-integral implicit model following," *J. Guid., Cont., Dyn.*, vol. 13, no. 2, pp. 303-309, Mar.-Apr. 1990.
[81] R. F. Stengel, J. R. Broussard, and P. W. Berry, "Digital flight control design for a tandem-rotor helicopter," *Automatica*, vol. 14, no. 4, pp. 301-311, July 1978.
[82] J. A. Foxgrover, *Design and Flight Test of a Digital Flight Control System for General Aviation Aircraft*, Princeton Univ. M.S.E. Thesis, MAE 1559-T, June 1982.
[83] J. R. Broussard, "Design, implementation and flight testing of PIF autopilots for general aviation aircraft," NASA CR-3709, Washington, DC, July 1983.
[84] C. E. Garcia and M. Morari, "Internal model control. 1. A. unifying review and some new results," *I&EC Proc. Des. & Devel.*, vol. 21, pp. 308-323, 1982.
[85] R. E. Kalman, "When is a linear control system optimal?" *ASME J. Basic Eng.*, vol. 86, pp. 51-60, Mar. 1964.
[86] B. D. O. Anderson and J. B. Moore, *Linear Optimal Control*. Englewood Cliffs, NJ: Prentice Hall,1971.
[87] N. A. Lehtomaki, N. R. Sandell, and M. Athans, "Robustness results in linear-quadratic-Gaussian based multivariable control designs," *IEEE Trans. Auto. Cont.*, vol. AC-26, no. 1, pp. 75-93, Feb. 1981.
[88] J. C. Doyle, "Guaranteed margins for LQC regulators," *IEEE Trans. Auto. Cont.*, vol. AC-23, no. 4, pp. 756-757, Aug. 1978.
[89] J. C. Doyle and G. Stein, "Multivariable feedback design: Concepts for a classical/modern synthesis," *IEEE Trans. Auto. Cont.*, vol. AC-26, no. 1, pp. 4-16, Feb. 1981.
[90] M. G. Safonov, A. J. Laub, and G. L. Hartmann, "Feedback properties of multivariable systems: The role and use of the return difference matrix," *IEEE Trans. Auto. Cont.*, AC-26(1), pp. 47-65, Feb. 1981.
[91] P. Dorato, ed., *Robust Control*. New York: IEEE Press, 1987.
[92] P. Dorato and R. K. Yedavalli, Ed., *Recent Advances in Robust Control*. New York: IEEE Press, 1990.
[93] J. C. Doyle, "Analysis of feedback systems with structured uncertainties," *IEEE Proc.*, vol. 129, part D, no. 6, pp. 242-250, Nov. 1982.
[94] R. F. Stengel, "Some effects of parameter variations on the lateral-directional stability of aircraft," *J. Guid., Cont.*, vol. 3, no. 2, pp. 124-131, Apr. 1980.
[95] R. F. Stengel and L. Ryan, "Stochastic robustness of linear-time-invariant control systems," *IEEE Trans. Auto. Cont.*, vol. 36, no. 1, pp. 82-87, Jan. 1991.
[96] L. R. Ray and R. F. Stengel, "Application of stochastic robustness to aircraft control," *J. Guid., Cont., Dyn.*, vol. 14, no. 6, pp. 1251-1259, Nov.-Dec. 1991.
[97] R. F. Stengel and C. I. Marrison, "Robustness of solutions to a benchmark control problem," *J. Guidance, Control, and Dynamics*, vol. 15, no. 5, pp. 1060-1067, Sept.-Oct. 1992.
[98] R. F. Stengel and C. I. Marrison, "Stochastic robustness synthesis for a benchmark problem," in *Proc. Amer. Cont. Conf.*, Chicago, pp. 2421-2422, June 1992.
[99] R. F. Stengel, P. W. Berry, and J. R. Broussard, "Command augmentation control laws for maneuvering aircraft," AIAA 77-1044, New York, Aug. 1977.
[100] M. Sugeno, "An introductory survey of fuzzy control," *Info. Sci.*, vol. 36, 1985, pp. 59-83.
[101] S. Chand and S. Chiu, "Robustness analysis of fuzzy control systems with application to aircraft roll control," *Proc. Guid., Nav., Cont. Conf.*, New Orleans, pp. 1676-1679, Aug. 1991.
[102] M. Steinberg, "Potential role of neural networks and fuzzy logic in flight control design and development," AIAA 92-0999, Washington, DC, Feb. 1992.
[103] S. N. Singh and W. J. Rugh, "Decoupling in a class of nonlinear systems by state feedback," *ASME J. Dyn. Syst. Meas. Cont.*, series G, vol. 94, pp. 323-329, Dec. 1972.
[104] A. Isidori, *Nonlinear Control Systems*. Berlin: Springer-Verlag, 1989.
[105] S. H. Lane and R. F. Stengel, "Flight control design using nonlinear inverse dynamics," *Automatica*, vol. 24, no. 4, pp. 471-483, July 1988.
[106] D. J. Linse and R. F. Stengel, "A system identification model for adaptive nonlinear control," in *Proc. Amer. Cont. Conf.*, Boston, pp. 1752-1757, June 1991.
[107] D. J. Linse and R. F. Stengel, "Identification of aerodynamic coefficients using computational neural networks," AIAA 92-0172, Washington, DC, Jan. 1992.
[108] E. Sentoh and A. E. Bryson, Jr., "Inverse and optimal control for desired outputs," *J. Guid., Cont., Dyn.*, vol. 15, no. 3, pp. 687-691, May-June 1992.
[109] T. Poggio and F. Girosi, "Regularization algorithms for learning that are equivalent to multilayer networks," *Science*, vol. 247, no. 4945, pp. 978-982, Feb. 23, 1990.
[110] D. J. Linse and R. F. Stengel, "Neural networks for function approximation in nonlinear control," in *Proc. Amer. Cont. Conf.*, San Diego, May 1990, pp. 675-679.
[111] I. K. Sethi, "Entropy nets: From decision trees to neural networks," in *Proc. IEEE*, vol. 78, no. 10, pp. 1605-1613, Oct. 1990.
[112] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," in *Neural Networks*, vol. 2, pp. 183-192, 1989.
[113] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Math. Cont., Sig., Sys.*, vol. 2, no. 4, pp. 303-314, 1989.
[114] T. Holcomb and M. Morari, "Local training for radial basis function networks: Towards solving the hidden unit problem," in *Proc. Amer. Cont. Conf.*, pp. 2331-2336, June 1991.
[115] M. G. Cox, "Data approximation by splines in one and two independent variables," in *The State of the Art in Numerical Analysis*, A. Iserles and M. J. D. Powell, Ed. Oxford: Clarendon, pp. 111-138, 1987.
[116] S. H. Lane, M. G. Flax, D. A. Handelman, and J. J. Gelfand, "Multi-layered perceptrons with B-spline receptive field functions," to appear in *Neural Information Processing Systems*, Morgan Kaufmann, Palo Alto, CA.
[117] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *ASME J. Dyn. Sys., Meas., Cont.*, vol. 97, pp. 220-227, Sept. 1975.
[118] W. T. Miller, "Sensor-based control of robotic manipulators using a general learning algorithm," *J. Robot. Auto.*, vol. RA-3, no. 2., pp. 157-165, Apr. 1987.
[119] S. H. Lane, D. A. Handelman, and J. J. Gelfand, "Theory and development of higher-order CMAC neural networks," *IEEE Cont. Sys. Mag.*, vol. 12, no. 3, pp. 23-30, Apr. 1992.
[120] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst., Man, Cyber.*, vol. 18, no. 1, pp. 49-60, Jan.-Feb. 1988.
[121] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Acad. Sci.*, vol. 79, pp. 2554-2558, Apr. 1982.
[122] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 815-826, Sept.-Oct. 1983.
[123] T. Kohonen, "Adaptive, associative, and self-organizing functions in neural computing," *Appl. Opt.*, vol. 26, no. 23, pp. 4910-4918, Dec. 1987.
[124] S. Naidu, E. Zafiriou, and T. McAvoy, "Use of neural networks for sensor failure detection in a control system," *IEEE Cont. Sys. Mag.*, vol. 10, no. 3, pp. 49-55, Apr. 1990.
[125] K. Watanabe, *et al.*, "Incipient fault diagnosis of chemical processes via artificial neural networks," *AIChE J.*, vol. 35, no. 11, pp. 1803-1812, Nov. 1989.
[126] T. Sorsa, H. N. Koivo, and H. Koivisto, "Neural networks in process fault diagnosis," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 4, pp. 815-825, July-Aug. 1991.
[127] E. Y. Chow and A. S. Willsky, "Analytical redundancy and the design of robust failure detection systems," *IEEE Trans. Auto. Cont.*, vol. AC-29, no. 7, pp. 603-614, July 1984.
[128] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464-1480, Sept. 1990.
[129] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognitions, Vol. 1: Foundations*, D. Rumelhart and J. McClelland, Ed. Cambridge: MIT Press, 1986.
[130] P. J. Werbos, "Backpropagation through time: What it does and how to do it," in *Proc. IEEE*, vol. 78, no. 10, pp. 1550-1560, Oct. 1990.
[131] S. Singhal and L. Wu, "Training feed-forward networks with the extended Kalman algorithm," in *Proc. Int. Conf. Acous., Speech, Sig. Proc.*, Glasgow, pp. 1187-1190, May 1989.
[132] E. Levin, N. Tishby, and A. A. Solla, "A statistical approach to

learning and generalization in layered neural networks," *Proc. IEEE*, vol. 78, no. 10, pp. 1568–1574, Oct. 1990.
[133] L. Davis, ed., *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann, Palo Alto, CA, 1987.
[134] G. L. Bilbro and W. E. Snyder, "Optimization of functions with many minima," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 4, pp. 840–849, July–Aug. 1991.
[135] W. Goldenthal and J. Farrell, "Application of neural networks to automatic control," *Proc. AIAA Guid., Nav., Cont. Conf.*, Portland, OR, pp., 1108–1112, Aug. 1990.
[136] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Cont. Sys. Mag.*, vol. 10, no. 3, pp. 18–23, Apr. 1990.
[137] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4–27, Mar. 1990.
[138] M. Fadali, *et al.*, "Minimum-time control of robotic manipulators using a back propagation neural network," in *Proc. Amer. Cont. Conf.*, San Diego, pp. 2997–3000, May 1990.
[139] S. Nagata, M. Sekiguchi, and K. Asakawa, "Mobile robot control by a structural hierarchical neural network," *IEEE Cont. Sys. Mag.*, vol. 10, no. 3, pp. 69–76, Apr. 1990.
[140] T. Troudet, S. Garg, and W. C. Merrill, "Neural network application to aircraft control system design," in *Proc. Guid., Nav., Cont. Conf.*, pp. 993–1009, Aug. 1991.
[141] Y.-L. Gu, "On nonlinear system invertibility and learning approaches by neural networks," in *Proc. Amer. Cont. Conf.*, San Diego, pp. 3013–3017, May 1990.

Robert Stengel (F'93, SM'85, M'77) received the S.B. degree in aeronautics and astronautics from M.I.T., Cambridge, MA, in 1960. He received the M.S.E., M.A., and Ph.D. degrees in aerospace and mechanical sciences from Princeton University, NJ, in 1965, 1966, and 1968, respectively.

He was with the Analytical Sciences Corporation, the Charles Stark Draper Laboratory, U.S. Air Force, and the National Aeronautics and Space Administration. He was a principal designer of the Project Apollo Lunar Module manual attitude control logic. Also, he contributed to the design of the Space Shuttle guidance and control system. Since 1977, he has been a Professor of Mechanical and Aerospace Engineering at Princeton University, where he directs the Topical Program on Robotics and Intelligent Systems and the Laboratory for Control and Automation. His current research focuses on system dynamics, control, and machine intelligence. He teaches courses on control and estimation, aircraft dynamics, space flight engineering, and aerospace guidance. He is the author of *Stochastic Optimal Control: Theory and Application*. He is also Associate Editor at large of the *IEEE Transactions on Automatic Control*. Currently, he is writing a book on aircraft dynamics and control. He has authored or co-authored over 150 technical papers and reports.

Dr. Stengel is an Associate Fellow of the AIAA and a Member of the SAE Aerospace Control and Guidance Systems Committee. He is a Member of the Program Council for the New Jersey Space Grant Consortium. He was Vice Chairman of the Congressional Aeronautical Advisory Committee and Chairman of the AACC Awards Committee. Also, he has served on numerous governmental advisory committees.