

Figure 1. Air combat maneuvers.

In our implementation, the above air combat maneuver trajectories were broken up into triplets, which qualitatively describe changes in distance, velocity, and acceleration. Building blocks of the continuous maneuver trajectories, "eigencurves," were constructed by considering all possible combinations of (velocity, acceleration) values. The resulting *qualitative* vector sequences describe the transitional behavior of the continuous-time air combat maneuvers. These sequences provided training input (via examples), to a neural network which was successfully trained, after 60,000 iterations, on a Sun-4/260 workstation. In this "teaching/training-by-examples" session, without any knowledge of algorithms or heuristics, the neural network arrived (as expected) at a self-organized solution. The training process involved utilization of back-propagation and of the Cumulative Delta Learning Rule. Given input data (vector sequences describing motion of aircraft) and corresponding output data (candidate countermaneuvers), this rule adjusts the interconnection weights between the processing elements of the artificial neural network by minimizing the output error. Once trained, this neural network provides a robust, fault-tolerant module designed 1) to rapidly identify air combat maneuvers (based on partial information), and 2) to suggest the best possible countermaneuvers.

Expert systems, which initially began as research projects in academia, have found numerous industrial, business, and military applications in the past decade in the forms of decision-aiding, scheduling, and control systems. A typical expert system reasons about incoming information by relating (pattern, decision) and/or (action, decision) pairs. On the other hand, in the late 1980s, expert systems of a new kind, namely neural networks began to be utilized in diverse ways, including the classification of underwater sonar targets, recognition of radar targets, and

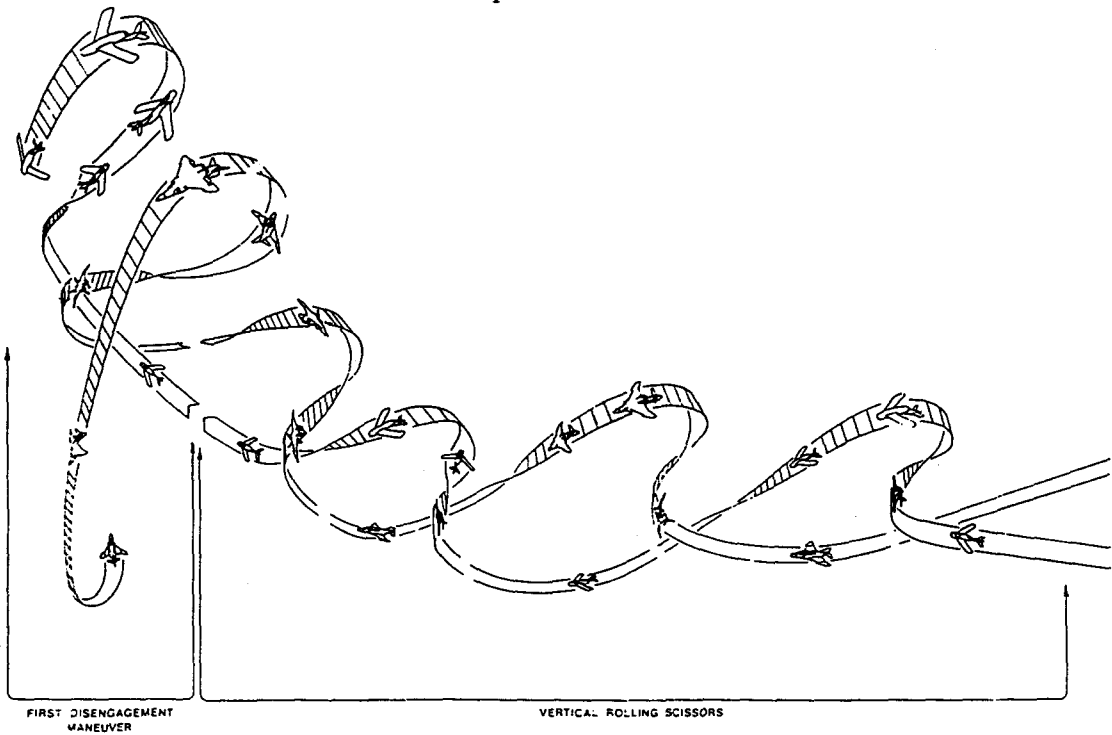


Figure 2. "Duel of the Aces," the Cunningham-Toon dogfight.¹

target shape estimation. Here, however, our goal is the maneuver identification and trajectory prediction of a dynamic aircraft, in the presence of incomplete information.

Neural networks are information processing systems which consist of a large number of simple processing elements (PEs) called *units*; each of the PEs is connected to other PEs via information channels called *interconnects*. Each unit can have multiple input signals but only one output signal; units typically take a weighted sum of all their inputs. Neural networks are fault-tolerant, i.e., if a processing element is disabled or destroyed, then the behavior of the network is only slightly altered; this is due to the fact that information is distributed throughout the network. Neural networks have many appealing features; we list a few of them here:

- Sensory interaction (real-time mode);
- Input data may contain errors and uncertainty;
- Approximate solutions based on noisy, fuzzy data are available;
- Exact algorithm for the solution need not be known, but many examples are available (e.g., training the network by using the tactical air combat maneuvers);
- Inherently parallel processing enables all processors to operate simultaneously on incoming information.

The history of intelligent systems goes back to Norbert Wiener, who considered models of neural activity as early as the 1950s. He observed that the behavior of neurons can be described by discontinuous, highly nonlinear equations; these models of biological control systems are important to mention, because they exhibit hierarchical structures which are capable of changing within determined "equivalence classes" in order to adjust to their exosystem. This quality, namely that of slow changes in the interconnections between neurons, is thought of as responsible for learning and memory. These observations, together with the lack of adequate analytical methods for complex problems, provide motivation for the construction of self-organizing (i.e., learning) systems which are capable of robust performance in hostile, unpredictable, or random environments, and which are fault-tolerant, i.e., can cope with component failure. The goal of such a system is to move toward a desired output state, in spite of ignorance of solution algo-

¹With grateful acknowledgement to aviation illustrators Mark and Matt Waki; reprinted in *Top Gun* by George Hall, p. 78 (CA: Presidio Press, 1987).

gorithms, domain heuristics, controls, or process dynamics. This learning (betterment) process can be performed on-line as well as off-line.

The related subjects of adaptation, learning, and self-organization have been interpreted in various ways in several disciplines; these range from systems theory to psychology. Several definitions of these concepts have been proposed, among others, by Wiener, Shannon, and Zadeh, whose description of an adaptive system is one which behaves stably and reliably, responding well with respect to a performance function when faced with a changing environment. This component, i.e., the performance function, determines the character of the system, maintaining the goal-directed behavior. Such a function may contain instructions to "reduce the error between input and output" in a simple feedback system, or "reduce fuel rapidly if temperature rises abruptly" in a linguistic process controller.

2. PREDICTION AND IDENTIFICATION

2.1. Artificial Neural Networks

Artificial neural networks (also known as *parallel/massive distributed processors* or *connectionist architectures*) are computing systems made up of a number of simple, highly interconnected processing elements (PEs), loosely modeled after organic neurons, which process information by their dynamic state response to external inputs. Having no separate memory for storing data, the processing elements operate in parallel, performing simple (typically taking a weighted sum of all inputs, and producing a single output) operations individually. Each processing element is characterized by a *transfer function* and a *firing threshold*. A few examples of commonly used transfer functions are

- (a) *sigmoidal nonlinearities*;
- (b) *hard limiters*;
- (c) *threshold logic*, and
- (d) *time-dependent functions*.

Their performance is weakly sensitive to details of their individual model, or to failure of some of the components, i.e., they are fault-tolerant; this is due to the absence of a symbolic representation in the network, as well as to the distribution of information in the matrix containing the connection weights, which resemble inhibitory and excitatory synaptic weights in biological neuron systems. In an analog electrical circuit model, these weights are represented by conductances, and neurons by small amplifiers.

Their operation is modified by *teaching/training* (repeated exposure of a neural net to an input data set), rather than by traditional computer programming. The process of self-adaptation at the PE level, i.e., *learning*, eventually leads to a general betterment of neural network system performance, and *self-organization* via learning is the modification process of the neural network system dynamics, employed in order to arrive at a desired result, or arrive at a specified goal.

Features of a successful neural network application include: high dimension of variable space (many variables); high redundancy in solution space (many almost equivalent acceptable solutions); sensory interaction in a real-time mode; allowance for possible errors, contradictions, and uncertainty of input data; lack of exact solution algorithms; and existence of a large number of constraints. These features are found in many problems encountered in optimization, classification and control.

Research on the subject of neural networks goes back to 1943, when McCulloch and Pitts, and Hebb proposed a multi-state Boolean algebra of neural activity [1,2]. Von Neumann later showed that McCulloch-Pitts networks were capable of functioning reliably, despite the loss of individual processing elements, when redundancies were introduced. The continuation of Von Neumann's research on a bi-state algebra led to a calculus which provides the foundation of today's digital computer logic.

In the mid-1950's, Frank Rosenblatt [3] proposed a neural network model for the human retina. His work at the Cornell Aeronautical Laboratory led to a procedure under which these networks, called *perceptrons*, could learn by systematically adjusting their weights. Perceptrons were capable of solving linearly separable classification problems; these networks consisted of a single layer.

Although perceptrons were by and large successful, they did possess certain limitations, one of which was the necessity of handcrafting each network for a specific type of problem; more importantly, as pointed out by Minsky and Papert [4], perceptrons were unable to solve nonlinearly separable classification problems.

Very little research on neural networks was done in the 1970s due to the scarcity of funds. Among the few scientists who continued to work on the subject were Widrow and Hoff, who developed adaptive filters for signal processing, which were applied to the problem of noise reduction in telecommunication channels. Bernard Widrow had developed the *Adaptive Linear Neurons* model (known as ADALINE) in 1963; the distinctions between ADALINE and perceptrons include

- (a) the former's acceptance of -1 and $+1$ inputs, whereas the latter accepts 0 and 1 . (ADALINE was successfully applied to pattern recognition problems for the classification of linearly separable patterns); and
- (b) the use of threshold logic in Widrow's model.

In 1963, Widrow and Smith published a paper in the *Proceedings of Computer and Information Sciences Symposium*, entitled "Pattern-Recognizing Control Systems," one of the earliest applications of neural network research to control theory. The formulation of a control problem as one of pattern recognition (an approach of *transformation* and *parametrization*), followed by the utilization of neural nets for purposes of classification, is the main strategy, as well as structure, of the research outlined in this paper. Since neural networks, as opposed to statistical classifiers are non-parametric, they do not require Gaussian distributions of data; these distributions can be generated by nonlinear processes. Furthermore, in 1960, Widrow and Hoff developed a learning rule, called the Delta rule or least mean square (LMS), this made it possible to automate the training process for simple networks.

In the early 1980s, John Hopfield of the Biophysical Division of Bell Laboratories, Steven Grossberg of Boston University, T. J. Sejnowski and D. Rosenberg of Johns Hopkins University, as well as David Rumerhart and James McClelland of Carnegie-Mellon University, developed various new neural network topologies which overcame the limitations of earlier perceptrons.

The network proposed by Steven Grossberg in 1976, used for feature discovery, is a hierarchical network, in which the first layer functions as the input layer and each subsequent layer is partitioned into disjoint sets called *clusters*. Within each layer, the nodes in each cluster compete by inhibiting others in the cluster, utilizing a winner-take-all learning strategy [5]. In 1982, John Hopfield proved that for neurons having one of two states, if complete connections and symmetric weights are enforced, the network will converge to a global state which minimizes a cost or an energy function. Neurons can be hardware-implemented by voltage amplifiers; here, the output varies between 0 and 1 . Each amplifier may have a number of inputs, both inhibitory and excitatory. Circuit behavior varies with time, and reaches a steady state when the system energy is at its minimum; the system may, however, converge to a local minimum. The idea behind the Hopfield energy function is the following: a positively weighted connection between two PEs acts as a constraint, in the sense that if one PE is activated, the other must also be activated; a negatively weighted connection corresponds to the opposite. The energy function is given by

$$E = - \sum_{i < j} x_i x_j \omega_{ij} + \sum_i x_i \Theta_i, \quad (1)$$

where ω_{ij} is the weight of connection between units i and j , and x_i is 1 if unit i is activated, and zero otherwise. Θ_i is a threshold for unit i . With this definition, the energy gap due to one processing element in the global energy function is:

$$\Delta E_i = E_{i \text{ off}} - E_{i \text{ on}} = \sum_j x_j \omega_{ij}. \quad (2)$$

If ΔE_i is greater than zero, the unit must be activated (or remain activated); if ΔE_i is less than zero, the unit should turn off (or remain off). Such systems have found numerous applications in artificial intelligence.

Another class of systems motivated by biological models is formulated via nonlinear difference equations. Because the equilibria of these systems usually has a non-trivial basin of attraction, it is possible for the network weights to converge to an equilibrium if the input is inside a region of attraction. Such stability considerations for neural networks were developed by Cohen and Grossberg [6], who showed that global stability results depend upon interconnection weights; the network admits a global Liapunov function whose equilibrium set can be analyzed. Recently, Bart Kosko proved the property of structural stability, i.e., insensitivity to perturbations, for a class of unsupervised nonlinear feedback neural networks called adaptive bidirectional associative memory models (ABAM models). Kosko accomplished this by formulating ABAM models as a system of stochastic differential equations, including Brownian diffusions. He then obtained the equilibria of random ABAM models, demonstrating their global stability.

In 1984, Hinton and Sejnowski [7] proposed a neural network model, the *Boltzmann machine*, where the network is treated as a metal which is cooling into crystalline structure (i.e., *simulated annealing*); in such a network, the activation function is a thermodynamical probability function; learning is accomplished when the temperature reaches a thermal equilibrium state determined by the Boltzmann distribution. Boltzmann machines are essentially identical to Hopfield nets; in the former, however, each processing element is activated with a probability given by the Boltzmann distribution. Again, the system may converge to a local minimum. The main objective of simulated annealing is to escape from high local minima by adding a random component to the decision process at each unit, i.e., each unit i computes ΔE_i as given by Equation 2. The probability of unit i at state 1 is given by:

$$P_i = \left[1 + e^{-\Delta E_i/T} \right]^{-1}, \quad (3)$$

where T is a scaling factor to control temperature (noise). If T is large, p is approximately .5, and the network assumes different states randomly. If T is zero (i.e., p_i is nonrandom), the network behaves like a Hopfield net. At any given T , after the system has reached thermal equilibrium, the relative probability of being at an energy surface with the local minimum at A , vs. a global minimum at B , is given by:

$$\frac{P_A}{P_B} = e^{-(E_A - E_B)/T}. \quad (4)$$

The most recent work by Y. Akiyama *et al.*, presented at the Fifth Aerospace Applications of AI Conference (October 1989), involves a new artificial neural network called the Gaussian machine, which includes the McCulloch-Pitts model, the Hopfield net, and the Boltzmann machine as special cases. Each neuron has several inputs, one output, and a random noise term ε ; this term helps the system escape from local minima when used in combination with a simulated annealing technique.

Another network proposed in 1984 by T. Kohonen, the *Kohonen self-organizing network* [8,9], consists of a single layer of highly interconnected elements; these elements receive inputs from the exosystem and from each other. The winner-take-all learning rule is then employed: for each presentation of input, activation in the layer is concentrated around the neighborhood of a winning processing element. In short, a Kohonen network models the probability distribution of the input vectors during training.

Also taking part in the recent renewal of interest in neural networks is Robert Hecht-Nielsen [10,11] who showed that any continuous function can be determined by a multi-layer neural network. In his 1965 paper concerning the structure of continuous functions of several variables, David Sprecher refined Kolmogorov's 1957 theorem dealing with the "thirteenth problem of Hilbert," which asks whether continuous functions of many variables can be represented by continuous functions of fewer variables. Papers by Sprecher [12] and Lorentz [13] contain results which extend Kolmogorov's original proof; the authors use representation of functions by linear superpositions. Kolmogorov had previously shown that there exist fixed continuous increasing functions $\Psi_{pq}(x)$ on $I = [0, 1]$ so that each continuous function f on I^n can be written as:

$$f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} g_q \left(\sum_{p=1}^n \Psi_{pq}(x_p) \right),$$

where g_p are properly chosen continuous functions of one variable. Hecht-Nielsen later made use of Sprecher's work in order to prove Kolmogorov's theorem in the context of neural networks (1987):

THEOREM 2.1. [10] *Given any continuous function $\phi : I^n \rightarrow \mathbb{R}^m$, $y = \phi(x)$, where I is the closed unit interval $[0,1]$ (and therefore I^n is the n -dimensional unit cube), ϕ can be implemented exactly by a three-layer neural network having n processing elements in the first (x -input) layer, $(2n+1)$ processing elements in the middle layer, and m processing elements in the top (y -output) layer. The processing elements on the bottom layer are fanout units that simply distribute the input x -vector components to the processing elements of the second layer.*

The processing elements of the second layer implement the following transfer function:

$$z_k = \sum_{j=1}^n \lambda^k \Psi(x_j + \varepsilon k) + k,$$

where the real constant λ and the continuous real monotonic increasing function Ψ are independent of ϕ (although they do depend on n) and the constant ε is a rational number $0 < \varepsilon < \delta$, where δ is an arbitrarily chosen positive constant. Further, it can be shown that Ψ can be chosen to satisfy a Lipschitz condition $|\Psi(x) - \Psi(y)| \leq c|x - y|^a$ for any $0 < a \leq 1$. The m top layer processing elements have the following transfer functions:

$$y_i = \sum_{k=1}^{2n+1} g_i(z_k),$$

where the functions g_i , $i = 1, 2, \dots, m$ are real and continuous (and depend on ϕ and ε).

The theorem can be summed up as follows: suppose we want to map any real n -dimensional vector to any other real vector of dimension m ; the only constraint we will place on the mapping is that the components of the input vector will all be scaled to lie in the closed unit interval; there is no such constraint on the output vector. Lippmann [14] summarizes: "The Kolmogorov theorem states that any continuous function of n variables can be computed using only linear summations and nonlinear but continuously increasing functions of only one variable. It effectively states that a three-layer perceptron with $n(2n+1)$ nodes using continuously increasing nonlinearities can compute any continuous function of n variables. However, the theorem does not indicate how weights or nonlinearities in the net should be selected or how sensitive the output function is to variations in the weights and internal functions." The above theorem therefore states that a multi-layer neural network exists which, instead of providing an approximation, can perform exact mapping. Furthermore, the network will have n neurons in its input layer, m neurons in its output layer, and $2n+1$ neurons in its middle layer. This result guarantees that a hierarchical or layered neural network can solve nonlinearly separable problems which simple single-layered networks cannot. In the next section, we shall outline and use the back-propagation training rule for a multi-layer feedforward network.

2.2. Neural Network Implementation of a Qualitative Approach

As mentioned earlier, neural networks have been utilized in problems dealing with classification of underwater sonar targets [15], recognition of radar targets [16], and target shape estimation [17]. In this context, however, our goal is the maneuver identification and trajectory prediction of a dynamic aircraft, in the presence of incomplete information. Specifically, we introduce a neural network designed to predict and identify tactical air combat maneuvers. Such a module, designed to function in a system which would guide medium and long range missiles toward unpredictable targets, would provide the ability to respond to rapid changes, and to predict future target location in real time.

2.2.1. Related Work

Here we briefly review ongoing research aimed at the development of AI-based systems in military applications [18-37]. Some of the existing strategies for estimation of threat potential include the following:

MEASUREMENT OF TURNING-IN (BOEING ADVANCED SYSTEMS) [18]

- (1) determines if velocity vectors of hostile and own ship are aligning (P1)
- (2) determines if the flight paths of hostile and own ship are becoming laterally closer (P2)
- (3) determines if the range between hostile and own ship is close (P3)
- (4) Measurement of turning-in = $P1 \times P2 \times P3$ (updates for each time plot)

THREAT RISK (R.M. YANNONE, GE AEROSPACE, 1988)

- (1) Threat Risk = Intent \times Lethality \times Countermeasure Effectiveness
- (2) Threat Intent: measure of the inferred goals of each threat; determined from kinematics, mode changes, and multi-sensor report
- (3) Threat Lethality: ability to harm the ship
- (4) Countermeasure Effectiveness: assessment of whether threat lethality has been reduced

BLUE TIME ADVANTAGE (BOEING ADVANCED SYSTEMS, 1987)

- (1) Time advantage of blue plane over red plane in seconds
- (2) An approximation of the number of seconds from the time that the blue missile becomes autonomous against the red target, allowing the blue aircraft to disengage
- (3) The time that the blue aircraft must disengage to avoid being hit

ESTIMATION OF TARGET ACCELERATION VECTOR (KEARFOTT CORP.)

- (1) Ideas borrowed from failure detection theory
- (2) Kalman filtering

Pilot's Associate (directed by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Air Force Wright Aeronautical Laboratories). In fiscal years 1985-1990, McDonnell-Douglas (St. Louis, MO) and Lockheed-Georgia received a multimillion-dollar contract to develop an artificial-intelligence-based Pilot's Associate. The goals of the project were as follows:

- (1) monitoring of aircraft systems
- (2) assessment of external/internal situations
- (3) mission planning and replanning
- (4) optimum-strategy planning to counteract threats

The three-phase research/development plan can be summarized thus:

- (1) Phase Zero: exploration of potential benefits (1985)
- (2) Phase One: integration of systems/operations in non-real-time situations (1986-1988)
- (3) Phase Two: integration of missions in dynamic, real-time situations (1989-1990)

Examples of various PA projects are:

- (i) The F-16 Emergency Procedures Expert System, which runs on a LISP machine, monitors the simulated aircraft's alarm systems, evaluates dynamic emergency situations, and advises the pilot;
- (ii) Enhanced Terrain-Masking Penetration Program (low observables), which takes advantage of the masking effect of hills and buildings to avoid enemy radar tracking; and
- (iii) Cognitive Requirements for Cockpit Automation, which mimics the decision-making processes of an experienced pilot, and improves pilot/display interface.

Another ongoing project is the joint effort by Rockwell International (Los Angeles, CA) and Messerschmitt-Bolkow-Blohm (MBB) (West Germany). Its goals include developing and testing a manned simulator which provides the pilot with decision-aiding and advanced fire control systems; it is designed to operate in medium-range air combat situations.

The current system developed by Rockwell International is called the Flight and Fire Control System (FFCS); the updated version of this, the Tactical Decision-Aiding Expert System, is being developed here at Washington University. Examples of the FFCS' various features include:

- (1) heuristic-based solutions for air-to-air combat
- (2) expert system applications to air combat
- (3) parallel processing architectures
- (4) improved sensors (sensor fusion and management)
- (5) C^3I interface

- (6) weapon models
- (7) use of A^* search, simulated annealing, and Semantic Control methods

FFCS was successfully tested in the Spring of 1989 in West Germany; the patent is pending.

We now outline our method, which is not limited to expressing an expert's surface-level knowledge in the form (pattern, decision); rather, it is based on a qualitative method [38,39]. This enables us to parametrize the dynamics of the maneuvers (position, velocity, acceleration) as a triplet, which functions as the input vector to the neural network.

2.2.2. Methodology: Qualitative Representation

The problem at hand involves the continuous trajectories of motion evident in most air combat maneuvers; "continuous" in the sense that the quantities of interest, i.e., position and velocity, are continuously differentiable, and have smooth rather than abrupt changes between values.

A qualitative representation [39] divides the range of values a quantity can take into a set of regions of interest, e.g., +, -, 0. The behavior of the system can be viewed in terms of a qualitative state diagram—a process known as *envisionment*. Whenever the qualitative value of a state variable is changed, the system itself changes state; the values in the next state are determined by

- (1) identifying those qualities which cannot change value, and
- (2) propagating the effect of those quantities which are known to change.

Therefore, if $f(t)$ is some continuously differentiable function of t , repeated differentiation of $f(t)$ with respect to t will produce a series of values. The series given by the signs of the derivative values will be characteristic of the relationship between the variables involved.

The series of derivative signs will define a curve shape. The derivative signs can be represented by symbols, chosen from the set $\{+0-\}$ ($?$ = an unknown value). The curve shape can be defined as the triplet (position, velocity, acceleration) by truncating the above series after the second derivative. This is represented symbolically by (d^0, d^1, d^2) , where $d^n f$ = the n^{th} derivative of $f(t)$.

The curve shapes are constructed by considering all possible values for d^1 and d^2 . Nine general relationships can be described (Figure 3).










		d^2 value		
		+	0	-
d^1 value	+			
	0			
	-			

Figure 3. Possible velocity/acceleration combinations.

The rules governing possible sequences of qualitative vectors are formed in terms of vector pairs, using predecessor and successor vectors, each of which has three elements $(d^0 d^1 d^2)$.

Four rules define the vector sequences, which represent physically possible maneuvers:

- (1) elements can only change to the opposite sign by a transition through zero;
- (2) if any element d^n has the same value in the succeeding vector d_{succ}^n , its lower-order element must behave consistently (since each element, except the first, is the rate of change of its lower-order element, a low-order element cannot increase if its higher-order element defines a decrease, and vice versa; nor may it change in any direction if its higher-order element defines it as stationary);

- (3) "flat" parts of relationships (where the derivative is zero) are interconnected by the curves of the correct shape;
- (4) any vector with $d^n = 0$, d^{n+1} nonzero will represent an infinitesimal interval in t . (Such intervals serve a purely theoretical purpose; due to the fact that they are impossible to detect, they are not represented in the actual neural network implementation.)

One immediate advantage of these rules is a reduction in the space of possible "next states"; considering all possible combinations of signs would produce a large number of potential state transitions.

Applying (1), (2) and (3) in terms of feasible transitions between adjacent curve shapes will greatly reduce the range of possibilities, which we show by a directed graph (Figure 4).

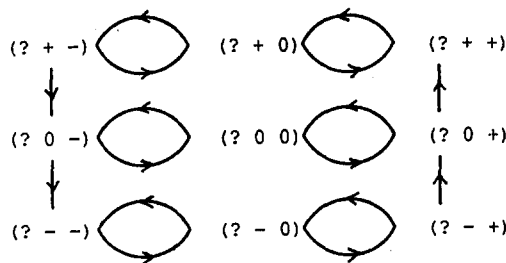


Figure 4. Feasible state-transition graph.

This graph enables us to compress a sequence of vectors without introducing ambiguity, permitting a compact expression of the relationship, which can be expanded (if needed) for more detail.

For example, the Defensive Rolling Scissors Maneuver can be represented by: $-1\ 1\ 1, -1\ 1\ 0, 0\ 1\ 0, 1\ 1\ -1, (0\ 0\ -1), -1\ -1\ -1, 0\ -1\ 0, 1\ -1\ 0, 0\ -1\ 1, -1\ -1\ 1, (-1\ 0\ 1), -1\ 1\ 1, 0\ 1\ 0, -1\ 1\ -1$; the Offensive Rolling Scissors Maneuver can be represented by: $-1\ 0\ 0, 0\ 0\ 0, 1\ 0\ 0, (1\ 0\ 1), 1\ 1\ 1, 0\ 1\ 1, -1\ 1\ 1, -1\ 1\ 0, 0\ 1\ -1, (1\ 0\ -1), 0\ -1\ -1, 1\ -1\ -1, 1\ -1\ 0, 0\ -1\ 1$. (Parenthesized sequences correspond to those of infinitesimal length described above in Step 4.) The other eleven maneuvers, not included here, can be similarly represented.

THEOREM 2.2. *The parametrizations of all the above maneuvers are unique.*

PROOF. Traversing the feasible state transition graph given above, one finds that there is only one traversal which represents each curve correctly.

This theorem provides an important property of the above qualitative formulation; this fact provides the above formulation some degree of generality, i.e., this method can be applied to represent various other continuous-trajectory dynamic phenomena (e.g., problems encountered in chemical process control).

In conclusion, the use of constraint techniques and transitional analysis appears to offer a powerful yet economic method of analyzing dynamic systems; all possible modes of a system can be identified while offering a complete parametrization of all possible tactical maneuvers. Additional information can be used to establish which of the several alternative behaviors will actually take place.

This qualitative approach breaks the maneuver trajectories presented in Figure 1 down into "eigencurves"-triplets describing relative, and therefore three-dimensional, changes in distance, velocity, and acceleration.

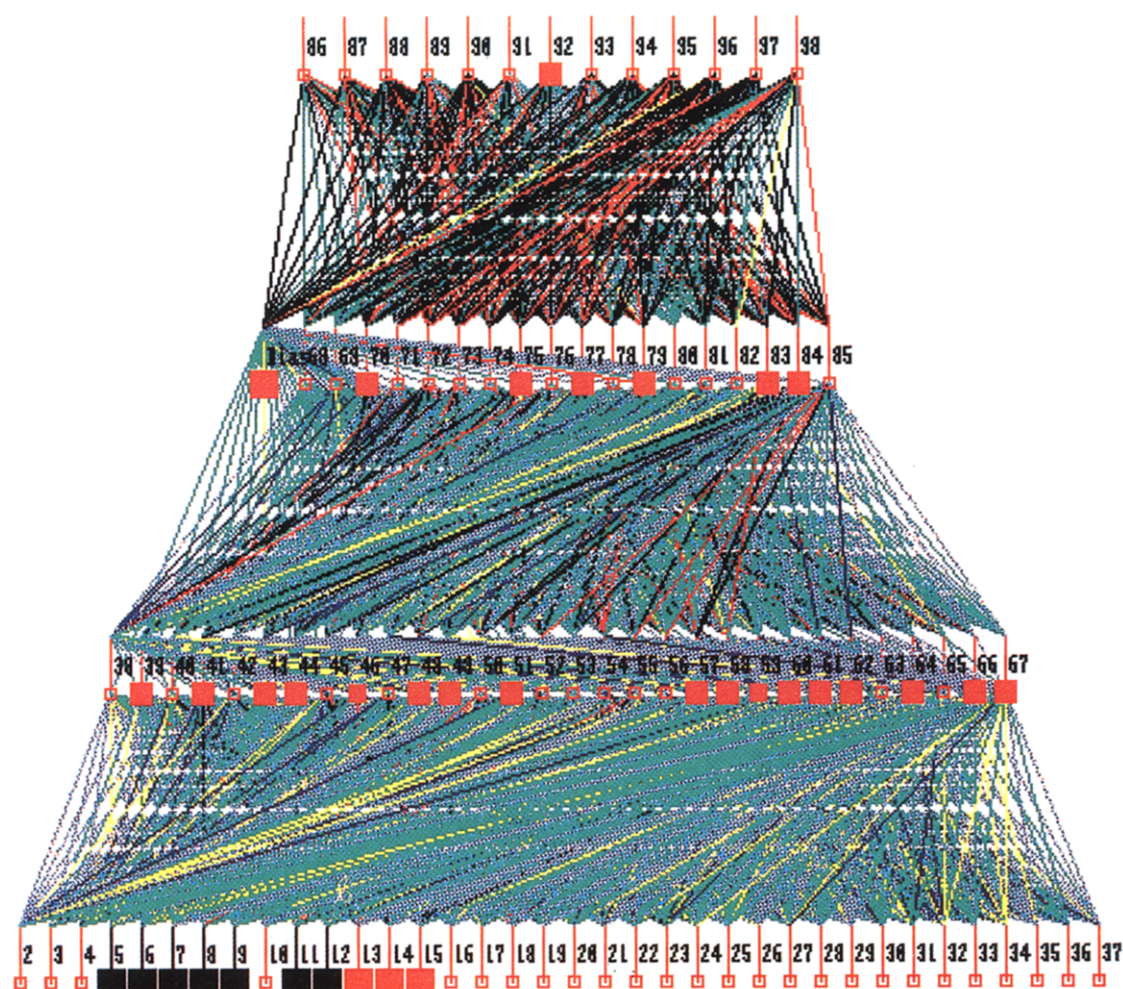
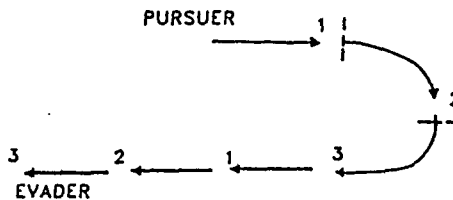


Figure 5. Three-layer neural network.



2.2.3. Implementation: Neural Networks

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \delta_i x_{ij},$$

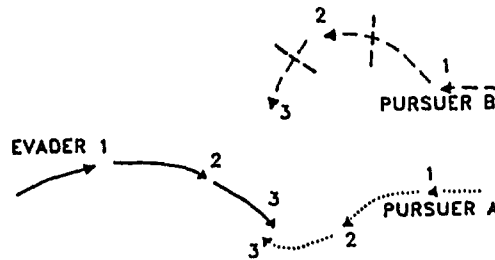
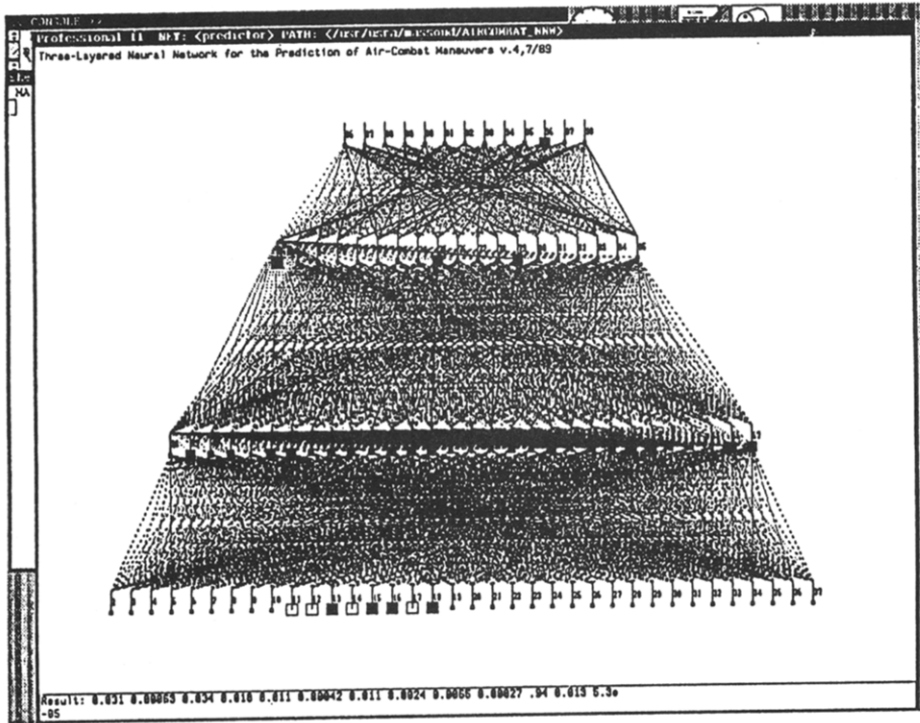


Figure 7. Offensive bracket (Over-The-Top) maneuver (38% of total information is given by the segment bounded within two broken lines; accuracy = 94%).

where $\omega_{ij}(t)$ is the weight from a hidden node (or an input node) i to a node j at time t ; x_i is either the output of node i or an input; η is the gain term (learning rate) between 0 and 1, which represents the speed of convergence; and δ_j is an error term for node j . If node j is an output node with an actual output y_j , then

$$\delta_j = y_j(1 - y_j)(d_j - y_j),$$

where d_j is the desired output of node j . If node j is a hidden node, then

$$\delta_j = x_j(1 - x_j) \sum_k (\delta_k \omega_{jk}),$$

where k is over all nodes in the layers above node j . Convergence may become faster if a momentum coefficient α is added, and weight changes are given by:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \delta_j x_i + \alpha [\omega_{ij}(t) - \omega_{ij}(t-1)]$$

We now define the following:

$x_j(q)$ = current output state of the j^{th} neuron on layer q ,

$I_j(q)$ = weighted sum of inputs to the j^{th} neuron on layer q ($\sum_i \omega_{ij}(q)x_i(q-1)$); therefore

$x_j(q) = f[I_j(q)]$; f is a sigmoidal nonlinearity.

Define the local error parameter passed through the layers as

$$e_j(q) = -\frac{\partial E}{\partial I_j(q)};$$

then, for sigmoidal nonlinearity, $f'(x) = f(x)[1 - f(x)]$, $e_j(q)$ becomes:

$$e_j(q) = f'(I_j(q)) \sum_k [e_k(q+1) \omega_{kj}(q+1)].$$

Now, using a gradient descent rule to reduce global error, we obtain:

$$\Delta \omega_{ij}(q) = -\eta \frac{\partial E}{\partial \omega_{ij}(q)}.$$

We can now calculate $\frac{\partial E}{\partial \omega_{ij}(q)}$ from local error values:

$$\frac{\partial E}{\partial \omega_{ij}(q)} = \frac{\partial E}{\partial I_j(q)} \frac{\partial I_j(q)}{\partial \omega_{ij}(q)} = -e_j(q) x_i(q-1).$$

Therefore, $\Delta \omega_{ij}(q) = \eta e_j(q) x_i(q-1)$; define the global error function E as:

$$E = \frac{1}{2} \sum_{t=1}^n (d_t - y_t)^2.$$

The local error at each node of the output layer becomes:

$$e_k(\text{output}) = -\frac{\partial E}{\partial I_k(\text{output})} = -\frac{\partial E}{\partial y_k} = d_k - y_k$$

The basic idea is to change the weights in the direction of decreasing energy; in this case the energy is a measure of errors at the output of the neural network.

The meaning of the output layer elements shown in Figures 5-7 is as follows:

<u>Output PE</u>	<u>Countermeasure for</u>
#86	Defensive Turn-in
#87	Defensive Lead-Turn
#88	Defensive Flat Scissors
#89	Defensive Rolling Scissors
#90	Defensive Hook-and-Drag
#91	Offensive Turn-in
#92	Offensive Lead-Turn
#93	Offensive Flat Scissors
#94	Offensive Rolling Scissors
#95	Offensive Bracket (underside)
#96	Offensive Bracket (over-the-top)
#97	Offensive Hook-and-Drag (flying Line-of-sight)
#98	Offensive Hook-and-Drag (go out and turn in)

Figures 6 and 7 illustrate the neural network implementation on a SUN-4/260 workstation, using Neural Works Professional IITM. The lowest level on the display corresponds to the input layer, and the uppermost level to the output layer. The window at the bottom of the display shows the actual input and result quantities. Given perfect information, the network selects the best countermeasure with accuracy of 96.7-98% (given in Table 1); however, our interest is limited

Table 1. Accuracy of countermaneuver predictions.

GIVEN INFORMATION	DEFEN. Turn-In	DEFEN. Lead-Turn	DEFEN. Flat Scis.	DEFEN. Roll. Scis.	DEFEN. Hook & Drag	OFFEN. Turn-In	OFFEN. Lead-Turn	OFFEN. Flat Scis.	OFFEN. Roll. Scis.	OFFEN. Brack. Part I*	OFFEN. Brack. Part II*	OFFEN. H&D Part I*	OFFEN. H&D Part II*
100%	97%	98%	97%	97%	97%	98%	93%	97%	97%	98%	97%	96.7%	97%
70-75%		93%			94%								
60-69%		37%*	87%				90%						
51-59%						95%			78%		67%		81%
40-50%	91%		91%	94%				91%		79%	79%	92%	
30-39%	79%					90%					94%		
25-29%						55%							

The "blank" entries correspond to those instances which occur either too soon or too late to predict a countermaneuver. Since our objective is to gain an advantage over the hostile aircraft by "early" maneuver prediction, we input the parts of the maneuvers which correspond to earlier segments only, i.e., those segments bound by two broken lines in Figures 6 and 7.

* = All other countermaneuver alternatives are negligible.

Table 2. Summary of robustness and fault-tolerance results.

GIVEN INFORMATION	DEFEN. Turn-In	DEFEN. Lead-Turn	DEFEN. Flat Scis.	DEFEN. Roll. Scis.	DEFEN. Hook & Drag	OFFEN. Turn-In	OFFEN. Lead-Turn	OFFEN. Flat Scis.	OFFEN. Roll. Scis.	OFFEN. Brack. Part I ¹	OFFEN. Brack. Part II*	OFFEN. H&D Part I*	OFFEN. H&D Part II*
CASE 1	92%	90%	60%	90%	36%	90%	91%	90%	90%	90%	42%	84%	90%
CASE 2	90%	90%	60%	90%	83%	?	?	?	90%	90%	60%	83%	90%
CASE 3	65%	60%	60%	90%	90%	83%	?	?	90%	68%	90%	90%	90%
70-72%		93%											
60-69%		35%*	67%				83%						
50-59%	43%*							90%					92%
41-49%			95%					?		59%	90%	91%	
33-40%	?						67%				92%		

Case 1: Eight processing elements are disabled in the first hidden layer; complete (100%) maneuver information is given;

Case 2: Same as in Case 1; in addition, two processing elements are disabled in the hidden layer;

Case 3: Four processing elements in the second hidden layer are disabled; all processing elements in the first hidden layer are enabled. "?" means failure to correctly identify the best countermaneuver. All entries below Case 3 are for available partial information (i.e., segments bounded by two broken lines in Figures 6 and 7). Four processing elements are disabled in the second hidden layer; all processing elements in the first hidden layer are enabled. The "blank" entries correspond to earlier segments only.

* = All other countermaneuver alternatives are negligible.

to prediction of countermaneuvers, given only partial and incomplete input. The segments of the maneuvers which are bounded within two broken lines (sketched in the lower half of Figures 6 and 7) indicate these "broken pieces." The network consistently performs well, even in cases of extremely sparse input values, in which case an accuracy of around 50% is obtained; even here, the ratio of best to second-best choice is still two to one. In cases where input values are more numerous (30-75% of total data given), accuracy is 67-94%, and a second choice is negligible. The network is very robust; if input error or failure of a few processing elements in the hidden layers occurs, accuracy is not affected (results are given in Table 2). This feature, combined with the ability to "massively" process information, renders this network, as opposed to the aforementioned techniques, extremely useful in real-time situations.

2.2.4. Summary/Conclusion

In conclusion, we have broken our central dynamical problem down into several smaller sub-problems ("eigencurves"), which describe the states of a continuous-trajectory dynamic system. We then used a qualitative method to determine which state transitions were possible.

We found that the resulting sequences of vectors uniquely expressed the time evolution of interacting dynamic objects. This method has been used to describe the forms of relationships

between accelerations and velocities (not the values themselves). All possible modes of a system can be identified while offering a complete parametrization of all possible tactical maneuvers. Additional information can be used to establish which of the several alternative behaviors will actually take place.

These sequences serve as the symbolic input to the neural network we have provided. We found that a single hidden layer could not satisfactorily distinguish (with at least 55–85% accuracy) simple one-on-one maneuvers, such as the Turn-In, from more complex two-on-one maneuvers; for this reason, two hidden layers were incorporated. For each layer, many different architectures and learning rules were tested; the network described here gives the best results (55–95% accuracy for partial information). A few of the results are summarized in Tables 1 and 2. Thus, we found that the neural network implementation provided a high-speed, fault-tolerant, and robust computational cell for the identification of tactical maneuvers and suggestions for a best countermaneuver.

We are currently incorporating the above module into an event-based automated reasoning system which

- (1) generates qualitative vectors by detecting changes in distances, velocities, and accelerations of hostile aircraft, and
- (2) provides inputs to the above neural network.

The neural network, as illustrated in this paper, will suggest an advantageous countermaneuver to overcome enemy actions. The presence of multiple hostile aircraft may cause a conflict as to which countermaneuver is preferable; therefore it is necessary to build an optimization (conflict resolution) model which will choose, from among alternatives, the countermaneuver which minimizes enemy threat/lethality. Finally, we mention that the entire event-based system discussed here will be a component in a general purpose Tactical Decision-Aiding Expert System which also learns differential games and their corresponding control laws. Such a system would increase the effectiveness of an autonomous missile as well as the on-board capabilities of a fighter pilot engaged in tactical maneuvers, examples of which we have recently seen in the Gulf of Sidra and the Persian Gulf.

REFERENCES

1. W.S. McCulloch and W. Pitts, A logical calculus of ideas immanent in nervous activity, *Bulletin of Mathematics and Biophysics* 5, 115–137 (1943).
2. W.S. McCulloch and W. Pitts, Neural networks, In Singh, Jagjit, *Great Ideas in Information Theory, Language and Cybernetics*, pp. 145–172, Dover Publications, New York, (1966).
3. F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, Washington, DC, (1962).
4. M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, (1969).
5. S. Grossberg, Nonlinear neural networks: Principles, mechanisms and architectures, *Neural Networks* 1, 17–61 (1988).
6. M. Cohen and S. Grossberg, Absolute stability of global pattern formation and parallel memory storage by competitive neural networks, *IEEE Trans. on Systems, Man, and Cybernetics* 3 (Sept./Oct.), 815–826 (1983).
7. G.E. Hinton and T.J. Sejnowski, Learning and relearning in Boltzmann machines, In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pp. 282–317, MIT Press, Cambridge, MA, (1986).
8. T. Kohonen, An introduction to neural computing, *Neural Networks* 1, 3–16 (1988).
9. T. Kohonen, *Self-Organization and Associative Memory*, Springer Series in Information Sciences, Berlin: Springer-Verlag, (1989).
10. R. Hecht-Nielsen, Kolmogorov's mapping neural network existence theorem, Presented at the *Proceedings of the IEEE International Conference on Neural Networks, San Diego*, pp. 1–4, (June 1987).
11. R. Hecht-Nielsen, Performance limits of optical, electro-optical, and electronic neurocomputers, *SPIE, Optical and Hybrid Computing* 634, 277–306 (1986).
12. David A. Sprecher, On the structure of continuous functions of several variables, *Trans. Amer. Math. Society* 115, 340–355 (March 1965).
13. G.G. Lorentz, The thirteenth problem of Hilbert, *Proceedings of Symposia in Pure Mathematics*, Vol. 28, pp. 419–430, American Mathematical Society, (1976).
14. R.P. Lippmann, An introduction to computing with neural nets, *IEEE ASSP Magazine*, 4–22 (April 1987).

15. P.R. Gorman and T.J. Sejnowski, Analysis of hidden units in a layered network trained to classify sonar target, *Neural Networks* 1, 75-89 (1988).
16. N.H. Farhat, S. Miyahara and K.S. Lee, *Optical Analog of Two-Dimensional Neural Networks and their Applications in Recognition of Radar Target*, pp. 146-152, American Institute of Physics, (1986).
17. N.H. Farhat and B. Bai, Echo inversion and target shape estimation by neuromorphic processing, *Neural Networks* 2, 117-125 (1989).
18. C.D. Meier and R.O. Stenerson, Recognition networks for tactical air combat maneuvers, Presented at the *Proceedings of the 4th Annual AAAI Conference, Dayton, OH*, (Oct. 1988).
19. S.M. Amin, Intelligent prediction methodologies in the navigation of autonomous vehicles, Ph.D. Thesis, Washington University, St. Louis, MO, (May 1990).
20. M.E. Bennett, Real-time continuous AI systems, *IEEE Proceedings* 134, Pt.D. (4), 272-277 (July 1987).
21. J.D. Corrigan, K.J. Keller and S.A. Meyer, Promise of decision aiding, *Aerospace America*, 30-31 (July 1989).
22. H.E. Fiala, Aerospace applications of AI-A survey, Presented at the *Fourth AAAI Conf.*, pp. 265-274, (Oct. 1988).
23. D.C. Fraser, Aircraft control systems—A projection to the year 2000, *IEEE Control Systems Magazine*, 11-13 (February 1985).
24. D.M. Hoamer, A pilot's view of intelligent systems, *Aerospace America*, 32-33 (July 1989).
25. K.J. Kaiser *et al.*, Integration of intelligent avionics systems for crew decision aiding, Presented at the *Fourth AAAI Conference*, pp. 230-241, (Oct. 1988).
26. T. Kurien and M.E. Liggins, Report-To-Target assignment in multisensor multitarget tracking, Presented at the *27th Conference on Decision and Control, Austin, TX*, pp. 2484-2488, (1988).
27. C.T. Leondes and J.M. Mendel, Artificial intelligence control, *Survey of Cybernetics*, (Edited by J. Rose), Iliffe Books, pp. 209-228, (1969).
28. Y. Lirov, Artificial intelligence methods in decision and control systems, Ph.D. Thesis, Washington University, St. Louis, MO, (August 1987).
29. J. Llinas, A survey of techniques for CIS data fusion, Presented at the *IEEE 2nd International Conference on CSI*, pp. 77-84, (April 1987).
30. R.K. Miller and T.C. Walker, *AI Applications in Engineering*, Fairmont Press, California, (1988).
31. R.R. Mitchell, Expert systems and air-combat simulation, *AI Expert*, 38-43 (Sept. 1989).
32. E.Y. Rodin and S.M. Amin, Intelligent navigation for an autonomous mobile robot, *Proceedings of the Third IEEE Int'l. Symposium on Intelligent Control*, (Aug. 1988).
33. E.Y. Rodin and S.M. Amin, Prediction and identification of tactical air combat manoeuvres: Neural network implementation of a qualitative approach, Presented at the *Fifth AAAI Conference, Dayton, OH*, (Oct. 1989).
34. E.Y. Rodin, Y. Lirov, B.G. McElhaney and L.W. Wilber, Artificial intelligent modelling of control systems, *Simulation* 50 (1), 12-24 (January 1988).
35. R. Shaw, *Fighter Combat Tactics and Maneuvering*, U.S. Naval Institute, Annapolis, MD, (1985).
36. D.M. Smith, Expert systems' role broadens, *Aerospace America*, 26-28 (July 1989).
37. M.L. Wright, *et al.*, An expert system for real-time control, *IEEE Software*, pp. 16-24, (March 1986).
38. J. deKleer, Qualitative and quantitative reasoning in classical mechanics, *Artificial Intelligence* (Edited by P.H. Winston and R.H. Brown), pp. 13-30, MIT Press, Cambridge, MA, (1982).
39. A.J. Morgan, Predicting the behaviour of dynamic systems with qualitative vectors, *Advances in Artificial Intelligence* (Edited by J. Hallam and C. Mellish), pp. 81-95, Wiley and Sons, (1988).
40. J.L. McClelland and D.E. Rumelhart, *Parallel Distributed Processing*, Vol 1, MIT Press, Cambridge, MA, (1988).