K. M. Passino, M. A. Sartori and P. J. Antsaklis, "Neural Computing for Numeric to Symbolic Conversion in Control Systems," I EEE C ontrol S ystems M agazine , pp. 44-52, April 1989.

Neural Computing for Numeric-to-Symbolic Conversion in Control Systems

Kevin M. Passino, Michael A. Sartori, and Panos J. Antsaklis

ABSTRACT: Neural computing offers massively parallel computational facilities for the classification of patterns. In this paper, a certain type of neural network, called the *multilayer perceptron*, is used to classify numeric data and assign appropriate symbols to various classes. This numeric-to-symbolic conversion results in a type of "information extraction," which is similar to what is called "data reduction" in pattern recognition. After introducing the idea of using the neural network as a numeric-to-symbolic converter, its use in autonomous control is discussed and several applications are studied. The perceptron is used as a numeric-to-symbolic converter for a discrete-event system controller supervising a continuous variable dynamic system. It is also shown how the perceptron can implement fault trees, which provide useful information (alarms) in a biological system and information for failure diagnosis and control purposes in an aircraft example.

Introduction

An important new branch of control theory is concerned with the study of discrete-event systems, which include manufacturing systems and traffic systems. Another relatively new area is the field of autonomous control theory, where "intelligent" controllers are used to control complex systems. Such systems include certain space systems and autonomous land and underwater vehicles. In both cases, as is explained below, symbolic (automata-theoretic) formalisms, as opposed to conventional differential/difference equation approaches, are often used in controller development. Frequently, however, portions (or all) of the plant output are numeric data rather than symbols. There is then the need to convert the numeric data to symbolic information, i.e., to perform numeric-to-symbolic conversion. Such a conversion is, for instance, inherently performed by fault trees in chemical process control to detect failures from plant data. Another example of such conversion is to reject or accept items in a manufacturing system based on their compliance with a tolerance. In this paper, a certain type of neural network, called the *multilayer perceptron*, is used to perform the numeric-to-symbolic conversion task. This paper is an expanded version of the work presented in [1].

The goal of this paper is to introduce the idea of using the multilayer perceptron as a numeric-to-symbolic converter to provide measurement information for specific control purposes; there is no contribution to the theory of neural networks or pattern recognition. The work involves the application of a particular neural network to problems often encountered in some of the more recent developments in control theory. In particular, the results of this paper show one alternative for implementing the interface between numeric and symbolic information, a problem of significant importance. The results here also suggest that the use of numeric-to-symbolic (and symbolic-to-numeric) converters is a possible starting point in system design (e.g., for "hybrid" systems). Other applications of neural networks to control have been in adaptive systems, since neural networks have an inherent learning capability. (See, for instance, the special section on neural networks for systems and control in [2].) Such applications are not studied here since the learning capabilities of the perceptron used are not utilized.

Neural computing involves the use of neural networks to perform computations in a massively parallel fashion. A *neural network* is a model that represents the structure and function of the interconnected network of neurons in, for instance, a human brain. Researchers hope to emulate the highly efficient "computing" that humans perform with ease. Here, a certain neural network, called the multilayer perceptron, is used to classify numeric data and assign appropriate symbols to the various classes. It is a feedforward model that is restricted so that it does not self-adjust its weights or learn. The classification it performs is a form of information extraction from given data. It is performed with massively parallel computing; hence, it is inherently faster than conventional sequential processing.

The conversion of numeric data to symbols is a form of information extraction. Intuitively speaking, *information extraction* is the transformation of detailed information into abstract, generalized information; there is always a certain loss of information in the extraction process. Here, the detailed information is an n-dimensional vector of real numbers, the numeric data. The abstract information is a set of symbols representing regions in the n-dimensional space. The information extraction process is performed here using the multilayer perceptron. The type of information obtained is, for example, "signal 1 is greater than 0 and less than 5, and signal 2 is greater than -3 and less than 10." The information lost in our extraction process is the exact value of the real numbers. (The "symbols" we refer to may not satisfy various mathematical properties such as addition and multiplication; however, they are normally chosen to have a particular physical interpretation.) It is understood that there are other types of information extraction, e.g., when the *behavior* of the system generating the data is characterized. Characterization of behavior could involve finding that "signal 1 is the derivative of signal 2."

The process of information extraction via the neural network studied here is closely related to what has been called *pattern recognition*. In fact, the multilayer perceptron solves what are called "deterministic classification problems" [3]. Similar problems have also been studied using the threshold logic circuits (networks) described in [4], except that in these circuits the theory only allows a finite number of discrete inputs, u. The pattern recognition process is viewed as a three-step "data reduction" process [3], [5]:

(1) Sensors make *measurements* of certain variables in the environment.

Kevin M. Passino, Michael A. Sartori, and Panos J. Antsaklis are with the Department of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, IN 46556.

0272-1708/89/0400-0044 $01.00 © 1989 IEEE

K. M. Passino, M. A. Sartori and P. J. Antsaklis, "Neural Computing for Numeric to Symbolic Conversion in Control Systems," I EEE C ontrol S ystems M agazine , pp. 44-52, April 1989.

(2) *Feature extraction* is then used to obtain a more global, high-level description of the measured data.

(3) *Classification algorithms* are used to name the pattern.

The viewpoint can be taken that this complete process is performed by the multilayer perceptron. The patterns sensed are points in the n-dimensional space of real numbers. Feature extraction entails determining which region each element of the n-dimensional vector lies in. Classification involves taking logical combinations of these regions and attaching labels to them. Rather than viewing the numeric-to-symbolic conversion process as a traditional pattern recognition problem, we prefer to view the multilayer perceptron as the device that performs only step 1 (in the preceding process) of a more abstract recognition process in which the goal is to take some action based on the pattern detected. For step 1 of the more abstract process, the perceptron senses data in the environment and provides symbols as its measurements. These symbols are then used by a controller to make decisions on what to input into the plant. Clearly, the controller may need abstract, "symbolic" feature extraction and classification to complete the high-level pattern recognition so that it can make its decisions, but these problems are not studied here. A similar view of pattern recognition is taken in [6]. Note that the method used here does not employ the more complex pattern recognition techniques (e.g., statistical or syntactical) to recognize wider classes of patterns. For example, as indicated earlier, it will not characterize the dynamical behavior of the environment over a period of time. For more information on pattern recognition techniques, see, for instance, [3], [7]. An alternative view from the perspective of artificial intelligence that supports the preceding ideas on the abstract classification process is given in [8].

Numeric-to-symbolic conversion can be used in several different ways in control systems. It can be used in the control of certain discrete-event systems, in autonomous control applications, and to implement fault trees. The three applications summarized here are described in more detail in the next section. The first applications considered are certain *discrete-event systems*. Note that there are cases when discrete-event controllers can be used for the control of continuous variable systems. In these instances, there is a need for a numeric-to-symbolic converter to provide the symbols to the controller. In the case of *autonomous control*, there is a "symbolic/numeric interface," where high-level, possibly "intelligent," controllers are used to control continuous systems [9], [10]. The multilayer perceptron can be used to convert numeric information into more abstract symbolic information for use in, for instance, a planning system. Finally, the information can be extracted by the perceptron so that it is useful in higher-level control decisions. For instance, the information provided by a *fault tree* on the failure status of a plant (what alarms should be on or off) can be used to generate appropriate control actions so that the effects of the failures can be eliminated. It is shown here that the multilayer perceptron can be used to implement a fault tree to produce alarms.

In the sections that follow, the theory of the multilayer perceptron is outlined. Next, three examples are presented. First, an example of the use of the perceptron as a numeric-to-symbolic converter for a surge tank is given. Next, it is shown how the perceptron can be used for the implementation of fault trees for a biological system and an aircraft. The paper closes with some concluding remarks and future directions.

Information Extraction in Control Systems: The Numeric-to-Symbolic Converter

In this section, we discuss several ways in which information extraction is used in control theory. In practice, information is always extracted—in the modeling of processes, for example. The extraction we discuss here is the transformation of numeric data to symbolic information, which is done dynamically in an on-line control system. There are three main applications to control theory discussed here.

The first is *discrete-event systems*. There are dynamic systems that either cannot be described by differential/difference equations, or it is not desirable to do so because subsequent analysis and design is very complicated, awkward, or simply inadequate. An alternative approach is to begin with a different model. In [11], the authors give an example of how to use a symbolic formalism—the nondeterministic finite-state automaton—to describe a plant (a surge tank) that is normally described with a nonlinear differential equation. Using this description, a finite-state controller was given and certain closed-loop properties were verified. In these cases, for controller implementation, a numeric-to-symbolic converter is needed to convert the sensor data to symbols for use in the discrete-event controller. The perceptron developed here is used to implement this converter. Another example of a combined study of discrete-event and continuous systems is given in [12].

The discrete-event controllers considered in this paper will be referred to as *symbolic controllers*, since they are developed using a symbolic formalism (automata-theoretic) and have symbols as inputs and outputs rather than numeric data. (Of course, numbers are symbols too, but the symbols we refer to here may not possess mathematical properties such as addition or multiplication.) The situation where we have a symbolic controller controlling a continuous variable dynamic system is depicted in Fig. 1. The symbolic controller uses Y_i (symbols) as inputs and a reference input, say, R_i (symbols), and generates the control inputs to the plant U_i (symbols). The numeric/symbolic converter transforms the measurement data $y(t)$ into symbols Y_i. The symbolic/numeric converter transforms the symbols U_i into numeric plant inputs $u(t)$.

The numeric/symbolic converter can be thought of as analogous to the analog-to-digital (A/D) converter in digital systems. In the one-dimensional case, the real number line (numeric data) is partitioned into regions associated with, say, binary numbers (symbols), and the values of the input are converted into the appropriate binary numbers. Indeed, there have been neural network implementations of a 4-bit A/D converter [13], [14]. The work here generalizes that application to the multi-input case using a different neural network. In addition to performing a more general conversion process than A/D conversion, the perceptron can operate in an asynchronous mode as well as in a synchronous one. Note that the perceptron can be expected to perform numeric-to-sym-

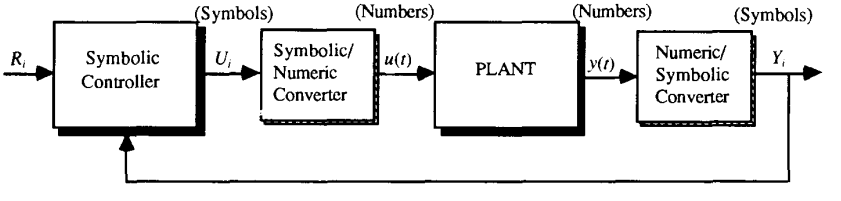


Fig. 1. Symbolic/numeric conversion.

K. M. Passino, M. A. Sartori and P. J. Antsaklis, "Neural Computing for Numeric to Symbolic Conversion in Control Systems," I EEE C ontrol S ystems M agazine , pp. 44-52, April 1989.

bolic conversion quickly. This is very important if it is to be placed in the control loop, because delays tend to cause instability. The symbolic controller's control actions on the continuous plant can be translated into numeric inputs by a generalized method analogous to digital-to-analog (D/A) conversion. It may convert a symbol to a simple numeric value or to a sine wave, square wave, or some other more complex signal. This problem is not addressed here.

Autonomous controller architectures can be viewed as hierarchical with three main levels (see, for example, [9], [10], and the references therein). These are the management and organization level, the coordination level, and the execution level. The *execution level* contains the hardware and conventional control algorithms performing low-level feedback control. The *management and organization level* has the interface to the operator and the intelligent learning and planning systems for supervising the actions taken at the lower levels. The *coordination level* provides a link between the management and organization level and the execution level. Autonomous controllers are used in the control of very complex systems that are normally hybrid. A *hybrid* system is one that contains dynamics that can be described by, for instance, differential or difference equations, *and* dynamics that are convenient to describe with some symbolic formalism. The controllers developed for such systems are also hybrid, with various numeric and symbolic components at each of the three levels. The numeric-type algorithms normally are used at the lower level of the architecture, the execution level, whereas the symbolic-type algorithms are operating at the highest level, the management and organization level. There is then a symbolic/numeric interface in the autonomous controller much like the one in certain discrete-event systems discussed earlier. Consequently, the perceptron developed here will be useful in autonomous control applications.

The results can also be used in the field of intelligent control called *expert control*. The "safety nets" in [15] use information extraction of ranges of variables for making high-level control decisions in adaptive control. The results of this paper provide a method to produce the information for such control decisions. In [15], the authors propose to use an *expert controller*, one that uses an expert system to emulate the actions a human operator would perform to maintain an adaptive controller's operation. The human in the control loop is being emulated. It is then natural to use a perceptron to emulate the human's pattern classification abilities.

In general, the multilayer perceptron can be used to classify data so that control decisions can be made. The symbols attached to various regions of the n-dimensional space of real numbers may have a particular physical interpretation. The perceptron can provide failure information such as "signal 1 is in range 4 AND signal 3 is in range 5 OR signal 7 is in range 11." This information may be quite useful in making control decisions. For instance, the values of signals 1, 3, and 7 may indicate that a certain failure condition is occurring and that a specific control action ought to be taken so that the failure will not degrade system performance. Failure information of the type described earlier is typically generated by *fault trees* [16], [17]. The *low-level failures* in fault trees are characterized by regions of certain variables, and fault-tree *high-level failures* are characterized by logical combinations of the low-level failures. Here, the multilayer perceptron determines in which regions certain variables appear and performs the appropriate ANDing and ORing to indicate what high-level failure has occurred. In the following section, an outline of multilayer perceptron theory is given.

Multilayer Perceptron for Numeric-to-Symbolic Conversion

The *multilayer perceptron* is a feedforward neural network used for neural computing (see, for example, [18]). The multilayer perceptron considered here contains a single *hidden layer* between the *input* and the *output layers*, as illustrated in Fig. 2 for the case of two nodes at each level. This cascading concept is well understood in control systems. For example, with two hidden layers, the output layer is measured directly. The first hidden layer feeds the output layer; the second hidden layer feeds the first hidden layer. The input of the perceptron is applied to the second hidden layer. The input to the multilayer perceptron considered here is the vector u with components u_i, and it contains M continuous real-valued elements. The vector x' (with components x_i'), which contains M_1 binary elements, is the output of the input layer and the input to the hidden layer. The vector x'', which contains M_2 binary elements, is the output of the hidden layer and the input to the output layer. The vector y (with components y_i), which contains N binary elements, is the output of the perceptron.

In Fig. 2, the *nodes* are denoted with circles and the *biases* with arrows that point downward. The biases on the input layer are denoted by b_i', on the hidden layer by b_i'', and on the output layer by b_i. The *weights* are denoted by all of the other arrows (which will be labeled with the weights) that are between u and the input nodal layer, between x' and the hidden nodal layer, and between x'' and the output layer. The element w_{ij} of the $M \times M_1$ matrix W denotes the weight on the arc from u_i to the node with x_j' as its output. The $M_1 \times M_2$ matrix W' denotes the weights on the arcs from each x_i' to the node with x_j'' as its output. The $M_2 \times N$ matrix W'' denotes the weights on the arcs from each x_i'' to the node with y_j as its output. The weights on the arcs connecting the output nodal layer to the outputs y_i are unity. For convenience, if the weight of any arc is zero, the arc will be omitted from the graphical representation of the perceptron.

Each node produces at its output a summation of its weighted inputs and its bias, which is passed through a threshold nonlinearity. The result is a binary output for each layer. Two typical threshold nonlinearities are illustrated in Fig. 3.

The input to the ith threshold nonlinearity of the input layer, denoted by $\bar{x}_i(t)$, is the weighted sum of the inputs added to the bias.

$$\bar{x}_i(t) = \sum_{j=1}^{M} (w_{ji}u_j(t)) + b_i' \qquad (1)$$

If $f_{k'}$ denotes a threshold nonlinearity of type k for the ith node, then the output of the input nodal layer is given by

$$x_i'(t) = f_{k'}(\bar{x}_i(t)) \qquad (2)$$

The outputs of the hidden layer and the output layer are given by similar relations. With these equations, the input-output relationship for the multilayer perceptron is easily specified. These equations were used for the development of the multilayer perceptron simulations in the examples.

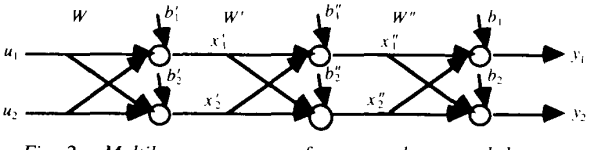


Fig. 2. Multilayer perceptron for two nodes at each layer.

K. M. Passino, M. A. Sartori and P. J. Antsaklis, "Neural Computing for Numeric to Symbolic Conversion in Control Systems," I EEE C ontrol S ystems M agazine , pp. 44-52, April 1989.

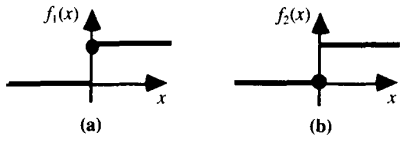


Fig. 3. Threshold nonlinearities for the multilayer perceptron.

The multilayer perceptron is commonly used for pattern classification. In the classification problem, one desires to identify certain "decision" regions that the input vector u lies in. Examples of dividing a two-dimensional input space into decision regions are illustrated in Fig. 4. An input pattern u is presented to the perceptron. After processing, the output vector y represents certain decision regions. In particular, if u is a member of one decision region, an appropriate element of the output vector y becomes a 1 (HI). If the input pattern u is not a member of a particular decision region, an appropriate element of the output vector y becomes a 0 (LO).

The weights and biases of the multilayer perceptron depend on the decision regions to be identified. The technique used to determine the weights and biases (and number of nodes) for the multilayer perceptron will be referred to herein as the "Harvey method." It is based on the results presented in [18] and is outlined as follows:

(1) Determine:
   (a) decision regions (hyperplanes), and
   (b) symbols that characterize various regions.

(2) To define the input nodal layer, for each hyperplane:
   (a) Add a node to the input layer, and
   (b) Define the weights and bias from each perceptron input to the node using the hyperplane definition.

(3) To define the hidden layer, use one node for each logical conjunction of input nodal layer outputs needed to define the output symbols.

(4) To define the output layer, use one node for each logical disjunction of hidden nodal layer outputs needed to define the symbols output from the perceptron.

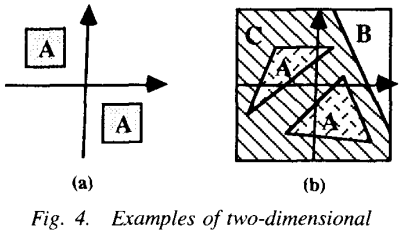


Fig. 4. Examples of two-dimensional decision regions.

Notice that not only are the weights and biases determined, but also the actual number of nodes in each nodal layer. As with the weights and biases, the number of nodes also depends on the decision regions to be identified. The three layers of the multilayer perceptron, as shown in Fig. 2, are named (from left to right) the input/plane layer, the AND layer, and the OR/output layer. The input/plane layer consists of nodes representing the hyperplanes dividing the input space. For example, in Fig. 4(a), region A needs eight one-input input/plane nodes since the region is constrained by eight different lines. The AND layer consists of one AND node for each decision region defined by more than one hyperplane. For example, in Fig. 4(a), region A needs two four-input AND nodes, each to denote that u is below one line, to the left of another line, above one line, and to the right of still another line. In other words, each four-input AND node will indicate that u lies in one of the two boxes that define region A. The OR/output layer consists of nodes used to associate disjoint decision regions. For example, in Fig. 4(a), region A needs one two-input OR/output node to indicate that u lies in one of the two boxes. Using the preceding method, the weights depend on which side of the plane the decision region lies, and the biases depend on the locations of the decision regions. As an alternative, the weights and biases could be determined by passing sample decision region data through the multilayer perceptron and updating the weights and biases based on a training (learning) algorithm (see [18]).

There are numerous other methods used to automatically adjust the weights and biases; these are considered to be learning algorithms. For instance, the Back-Propagation Training Algorithm, which is presented in [19], [20], can be used for perceptrons similar to the one shown in Fig. 2. This algorithm may be particularly useful if the input space must be divided into decision regions that are separated by curved borders. In [21], three other training procedures are introduced and compared to the Back-Propagation Training Algorithm. In [22], an alternative to the Back-Propagation Training Algorithm, called the Selective Update Back-Propagation Algorithm, is introduced and shown to work in cases where the Back-Propagation Training Algorithm will not. Through the presentation of input data, the multilayer perceptron's output, and the desired output, these algorithms train the weights and the biases until they stabilize. Although the Back-Propagation Training Algorithm seems to work for most cases, convergence of the algorithm has not yet been proven for the fully interconnected multilayer perceptron [19], [20]. Convergence for the lightly interconnected multilayer perceptron is considered in [23].

Examples of Numeric-to-Symbolic Conversion in Control Systems

In this section, three examples of how numeric-to-symbolic conversion can be performed via the multilayer perceptron are given. A discrete-event system is studied in the first example, and fault trees are produced for the last two.

Surge Tank

The first example, from chemical process control, involves a liquid-holding surge tank similar to the one studied in [11]. The tank is viewed as a *discrete-event system*, and its liquid level is controlled by a symbolic controller. The controller is a simple map from the plant outputs (liquid levels) to the plant inputs (valve open or closed). The tank is depicted in Fig. 5. The empty valve is unpredictably opened by some user, and the controller turns the fill valve on and off to keep the tank from becoming empty or full. The surge tank can be described with the first-order nonlinear differential equation [24],

$$dh(t)/dt = \{F_i(t) - kc(t)[h(t)]^{1/2}\}/\rho A \tag{3}$$

where $\rho = 99.8$, the fluid density of the liquid (water), $A = 1$, the cross-sectional area of the tank, $h(t)$ is the height of the liquid in the tank, and $k = 1000$, a physical constant. The value of $c(t)$ represents the variable outflow pipe cross-sectional area. The unpredictable user is modeled as $c(t) = 0.1c_0(t)$, where $c_0(t)$ is a random variable uniformly distributed on 0 to 1. The value of $F_i(t) = 100\sqrt{3}F(t)$, where $F(t)$ is either 1 (indicating that the fill valve is on) or 0

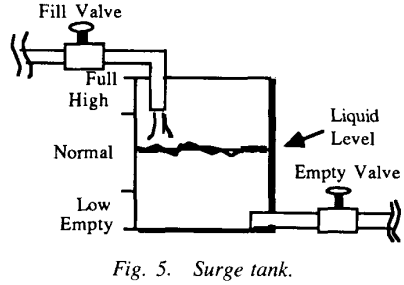


Fig. 5. Surge tank.

K. M. Passino, M. A. Sartori and P. J. Antsaklis, "Neural Computing for Numeric to Symbolic Conversion in Control Systems," I EEE C ontrol S ystems M agazine , pp. 44-52, April 1989.

(indicating that the fill valve is off). Notice that, with this choice, if the input valve is open, the liquid level will stay the same or rise.

The height of the liquid in the tank is discretized into five regions, which are represented with the symbols y_i, $i = 1, 2, 3, 4, 5$. The case when $F(t) = 1$ will be represented with the symbol u_1; $F(t) = 0$ with the symbol u_0. The symbolic/numeric converter is simply a device that outputs 1 when it has as an input u_1, and 0 when its input is u_0. The controller that is the same as in [11] is given in Table 1.

Next, the numeric/symbolic converter must be specified. Let $h_1 = 0$ (empty) and h_4 represent the case when the tank is completely full. Also, let h_2 and h_3 represent two particular levels in the tank that distinguish, respectively, a low liquid level from a normal one and a normal level from a high one. Table 2 gives the necessary information to specify the classification problem.

The multilayer perceptron designed using the Harvey method is illustrated in Fig. 6. The arrows indicate amplifiers. The shaded and unshaded circles indicate a summing of the inputs to the circle (arrows pointing toward the circle) and that the sum is passed through one of the two nonlinearities shown in Fig. 3. The shaded circles refer to Fig. 3(a) (f_1), and the unshaded circles refer to Fig. 3(b) (f_2). Using Harvey's method to specify the multilayer perceptron entails appropriately connecting nodes and arcs between $h(t)$ and y_i. For instance, we choose the first output $y_1 = f_1(h_1 - h(t))$ so that $y_1 = 1$ when $h(t) \leq h_1$; consequently, $y_1 = 1$ (HI) when $h(t) = h_1 = 0$. The output $y_1 = 0$ (LO) when $h(t) > h_1 = 0$. The output $y_2 = f_1(-1.5 + f_2(h(t) - h_1) + f_2(h_2 - h(t)))$ so that $y_2 = 1$ (HI) if $h_1 < h(t) < h_2$, otherwise $y_2 = 0$ (LO). Harvey's method was used in a similar manner for y_3, y_4, and y_5.

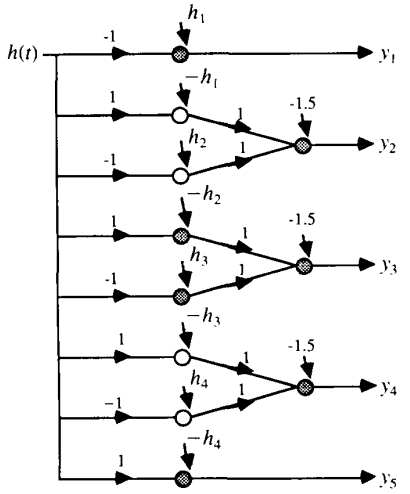


Fig. 6. Numeric/symbolic converter for the surge tank.

The object of the controller studied in [11] was to stabilize the height of the liquid in the surge tank around a particular level. In particular, it was desired that the steady-state behavior of the plant entail only a sequence of y_3 and y_4 alternating. When the plant, the numeric/symbolic converter, the controller, and the symbolic/numeric converter were connected together as in Fig. 1 and simulated, the output of the plant behaved as desired. The plant was given two initial conditions, completely empty and at level y_3. Each time, the controller first opened the fill valve and then closed the fill valve when the level rose to y_4. The input valve was opened again when the lower level fell to y_3 and closed again when the level rose to y_4. The system continued in this fashion, with the controller attempting to keep the level just above y_3. Consequently, the multilayer perceptron successfully implemented the numeric/symbolic converter for the surge tank and its symbolic controller.

Table 1
Surge Tank Controller

	Controller				
Controller Input	y_1	y_2	y_3	y_4	y_5
Controller Output	u_1	u_1	u_1	u_0	u_0

Biological System

The second example involves a biological system where the multilayer perceptron is used as a *fault tree* to produce alarm information. To model a form of biological growth, one of Volterra's population equations is used. A simple model representing the spreading of a disease in a given population is given by the following equations [25], where $x_1(t)$ is the density of the infected individuals, $x_2(t)$ is the density of the uninfected individuals, and $a > 0$, $b > 0$.

$$dx_1(t)/dt = -ax_1(t) + bx_1(t)x_2(t) \quad (4)$$

$$dx_2(t)/dt = -bx_1(t)x_2(t) \quad (5)$$

Equations (4) and (5) are valid only for $x_1(t) \geq 0$ and $x_2(t) \geq 0$. Equation (5) intuitively means that the uninfected individuals become infected at a rate proportional to $x_1(t)x_2(t)$. This term is a measure of the interaction between the two groups. The term $-ax_1(t)$ in Eq. (4) represents the rate at which individuals die from the disease or survive and become forever immune. The term $bx_1(t)x_2(t)$ in Eq. (4) represents the rate at which previously uninfected individuals become infected. The initial conditions $x_1(0) \geq 0$ and $x_2(0) \geq 0$ must also be specified.

The perceptron is to be designed to produce alarms if certain conditions occur in the diseased population; it is to implement a simple fault tree. The perceptron uses $x_1(t)$ and $x_2(t)$ as inputs to produce the following alarms at its outputs:

(i) "*Warning*: the density of infected individuals is unsafe"; this occurs if $x_1(t) > \alpha_1$, where α_1 is some positive real number.

(ii) "*Caution*: the density of the infected individuals is unsafe *and* the number of infected individuals is greater than the number of uninfected individuals"; this occurs if $x_1(t) > \alpha_1$ and $x_1(t) \geq x_2(t) + \alpha_2$ but $x_1(t) < x_2(t) + \alpha_3$, where α_2 and α_3 are positive real numbers.

(iii) "*Critical*: the density of the infected individuals is unsafe *and* the number of

Table 2
Surge Tank Numeric/Symbolic Converter Specifications

	Numeric/Symbolic Converter				
Numeric/Symbolic Converter Output	y_1	y_2	y_3	y_4	y_5
Plant Output Regions	$h(t) = h_1$	$h_1 < h(t) < h_2$	$h_2 \leq h(t) \leq h_3$	$h_3 < h(t) < h_4$	$h(t) = h_4$

K. M. Passino, M. A. Sartori and P. J. Antsaklis, "Neural Computing for Numeric to Symbolic Conversion in Control Systems," I EEE C ontrol S ystems M agazine , pp. 44-52, April 1989.

infected individuals is much greater than the number of uninfected individuals''; this occurs if $x_1(t) > \alpha_1$ and $x_1(t) \geq x_2(t) + \alpha_3$.

The three alarms represent certain faults characterized by the decision regions shown in Fig. 7.

The first fault mode occurs when the values of $x_1(t)$ and $x_2(t)$ lie to the right of the line $x_1(t) = \alpha_1$. The second occurs when $x_1(t)$ and $x_2(t)$ lie to the right of $x_1(t) = \alpha_1$ and $x_2(t) = x_1(t) - \alpha_2$ [or on $x_2(t) = x_1(t) - \alpha_2$] and to the left of $x_2(t) = x_1(t) - \alpha_3$. The third occurs when $x_1(t)$ and $x_2(t)$ lie to the right of $x_1(t) = \alpha_1$ and $x_2(t) = x_1(t) - \alpha_3$ [or on $x_2(t) = x_1(t) - \alpha_3$].

Using these decision regions, the Harvey method was used to produce the multilayer perceptron shown in Fig. 8. Each output becomes high when its corresponding fault occurs. The parameters α_1, α_2, and α_3 can be chosen according to the specific problem at hand.

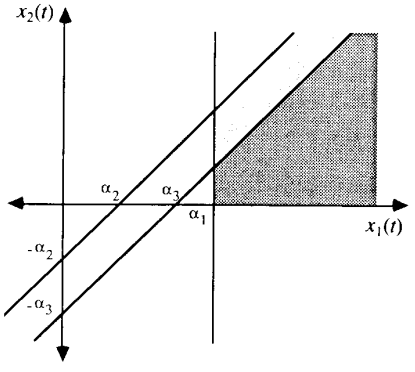


Fig. 7. Decision regions for the biological system.

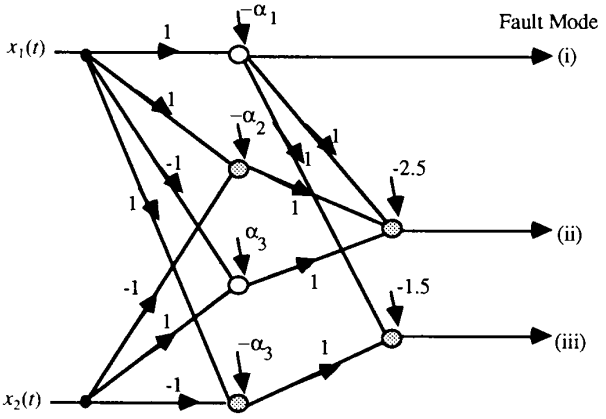


Fig. 8. Numeric/symbolic converter for the biological system.

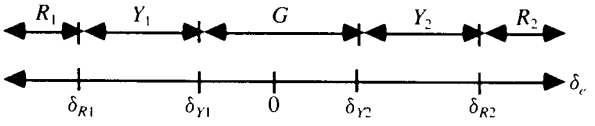


Fig. 9. Aircraft decision regions.

Aircraft

The third example involves the implementation of a *fault tree* for an aircraft. The multilayer perceptron is used to indicate the failure modes that depend on the aircraft's inputs and outputs. The aircraft's input vector u has two components: elevator (deg) δ_e and thrust (deg) δ_t. The output vector y has three components: pitch rate (deg/sec) q, pitch angle (deg) θ, and load factor (g) η_z. Four aircraft failure modes are considered here. Each input and output is discretized into five regions, with four boundaries associated with the real number line. For example, the elevator δ_e is discretized as shown in Fig. 9. The G (for green) region denotes an area of safe operation, the Y_1 and Y_2 (for yellow) regions denote areas of warning, and the R_1 and R_2 (for red) regions denote areas of unsafe operation. Table 3 defines the five regions for the elevator δ_e. The four other aircraft variables (δ_t, q, θ, η_z) are discretized in a similar manner using similar notation.

Using the defined regions for the parameters, four failure modes for the aircraft are identified as follows:

(1) Load factor is in region R_2 ($\eta_z \in R_2$).

(2) Load factor is in region Y_2 ($\eta_z \in Y_2$).

(3) Load factor is in region Y_2 and elevator is in region Y_1 ($\eta_z \in Y_2$ and $\delta_e \in Y_1$).

(4) (Pitch rate is in Y_1 and pitch angle is in Y_1) or (pitch rate is in Y_2 and pitch angle is in Y_2) [($q \in Y_1$ and $\theta \in Y_1$) or ($q \in Y_2$ and $\theta \in Y_2$)].

For the four failure modes, decision regions can be defined and a multilayer perceptron can be designed using the Harvey method. The perceptron's inputs are the aircraft inputs and outputs, and the perceptron's outputs are the four fault modes. For the first mode, the output is HI if the load factor η_z is in region R_2. Notice that $\eta_z \in R_2$ is equivalent to $\eta_z \geq \eta_{R2}$ or $\eta_z - \eta_{R2} \geq 0$. Using the last equation, the multilayer perceptron for the first fault mode is shown at the top of Fig. 10 (output 1). For the second fault mode, the output should be HI if the load factor η_z is in region Y_2. Let $\Leftrightarrow$ denote equivalence. Notice that the condition $\eta_{Y2} \leq \eta_z < \eta_{R2}$ is equivalent to

$$\eta_z \geq \eta_{Y2} \Leftrightarrow \eta_z - \eta_{Y2} \geq 0$$

and

$$\eta_z < \eta_{R2} \Leftrightarrow -\eta_z + \eta_{R2} > 0 \qquad (6)$$

Using the Eq. (6), the multilayer perceptron for the second fault mode is shown in the middle of Fig. 10 (output 2). For the third fault mode, the output should be HI if the load factor η_z is in region Y_2 and the elevator δ_e is in region Y_1. Notice that the first equation for this fault mode is the same as Eq. (6). The second equation (which must be ANDed with the preceding equation) is $\delta_{R1} < \delta_e \leq \delta_{Y1}$, which is equivalent to

$$\delta_e > \delta_{R1} \Leftrightarrow \delta_e - \delta_{R1} > 0$$

and

$$\delta_e \leq \delta_{Y1} \Leftrightarrow -\delta_e + \delta_{Y1} \geq 0 \qquad (7)$$

Using Eqs. (6) and (7), the multilayer perceptron for the third fault mode is shown in Fig. 10 (output 3).

For the fourth fault mode, the output should be high if either the pitch rate is in region Y_1 and the pitch angle is in region Y_1, or the pitch rate is in region Y_2 and the pitch angle is in region Y_2. The decision regions for the fourth fault mode are shown in Fig. 11. The shaded regions represent the regions that θ and q must occupy for the fourth failure to occur. To develop the perceptron, notice that the fourth failure mode can be characterized by

$$q - q_{R1} > 0 \text{ and } -q + q_{Y1} \geq 0$$

and

$$\theta - \theta_{R1} > 0 \text{ and } -\theta + \theta_{Y1} \geq 0$$

or

$$q - q_{Y2} \geq 0 \text{ and } -q + q_{R2} > 0$$

K. M. Passino, M. A. Sartori and P. J. Antsaklis, "Neural Computing for Numeric to Symbolic Conversion in Control Systems," I EEE C ontrol S ystems M agazine , pp. 44-52, April 1989.

Table 3
Elevator Decision Regions

	Elevator δ_e Regions				
Elevator Region	R_1	Y_1	G	Y_2	R_2
Elevator Range	$\delta_e \le \delta_{R1}$	$\delta_{R1} < \delta_e \le \delta_{Y1}$	$\delta_{Y1} < \delta_e < \delta_{Y2}$	$\delta_{Y2} \le \delta_e < \delta_{R2}$	$\delta_{R2} \le \delta_e$

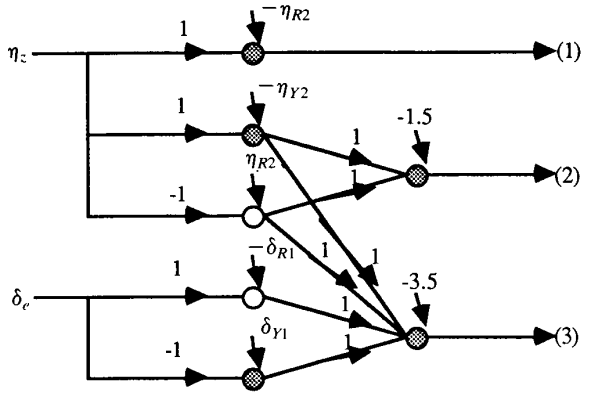


Fig. 10. Numeric/symbolic converter for aircraft fault modes 1-3.

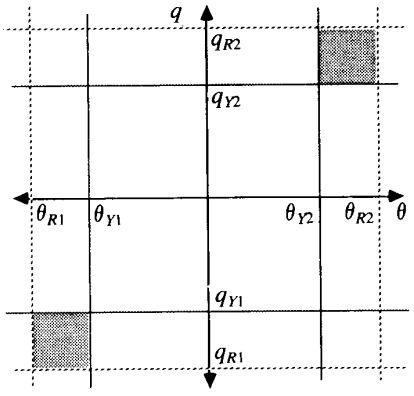


Fig. 11. Decision regions for aircraft fault mode 4.

and

$$\theta - \theta_{Y2} \ge 0 \text{ and } -\theta + \theta_{R2} > 0 \quad (8)$$

Using Eq. (8), the multilayer perceptron for the fourth fault mode is shown in Fig. 12. The aircraft and the multilayer perceptron that implemented the fault tree were simulated. The multilayer perceptron appropriately identified all the fault modes.

Concluding Remarks and Research Directions

In this paper, we have shown how to use a multilayer perceptron to solve a numeric-to-symbolic conversion task, which is often encountered in discrete-event and autonomous systems. The proposed method is used for deterministic classification problems. An important research direction is to determine the applicability of more elaborate pattern recognition techniques to the symbolic-to-numeric conversion problem. Then, for instance, the recognition of system behavior could be achieved.

It appears that the idea of using numeric/symbolic converters as a starting point in certain discrete-event or autonomous system designs is novel. The idea is introduced and illustrated via examples. Furthermore, the symbolic-to-numeric conversion task was discussed only briefly. The development of a complete theory of numeric/symbolic conversion is quite important, and it seems that it will impact research in hybrid systems (e.g., combined discrete-event and continuous variable systems).

Clearly, another important problem is how to determine the level of quantization for the decision regions (i.e., the number of hyperplanes needed). This problem is analogous to the choice of the quantization levels and sampling period in conventional digital systems; hence, it will impact many aspects of the system design. In general, the solution of this problem will be quite difficult because it depends on the plant, the controller, and the design objectives.

As noted, the technique proposed herein does not utilize the learning capability of the perceptron used. The problems encountered did not dictate the need for learning. In other problems, however, its use may be required; in that case, an algorithm that learns the boundaries of the regions is needed (for example, where there are curved decision regions). Several approaches to solve this learning problem were outlined. These

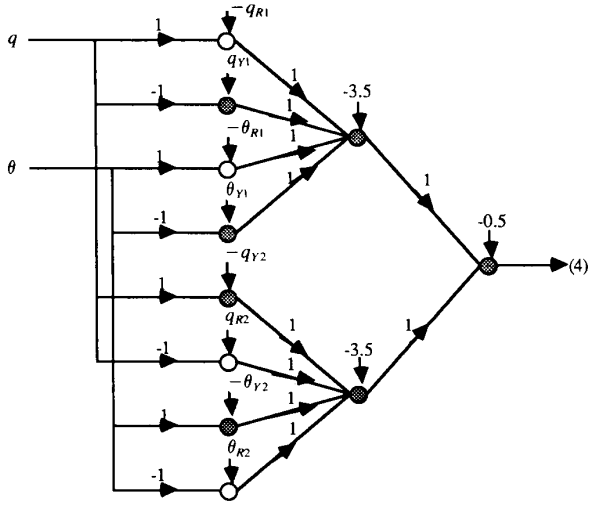


Fig. 12. Numeric/symbolic converter for aircraft fault mode 4.

methods seem quite promising for the numeric/symbolic conversion task but require more investigation. Alternatively, the results in [26], [27] may contribute some ideas on the knowledge of region boundaries.

Acknowledgments

This work was partially supported by the Jet Propulsion Laboratory under Contract 957856 and by the U.S. Army Research Office under Contract DAAL 03-88-K-0144. The authors would also like to thank the anonymous reviewers for their comments and suggestions.

References

[1] K. M. Passino, M. A. Sartori, and P. J. Antsaklis, "Neural Computing for Information Extraction in Control Systems," *Proc. 26th Annual Allerton Conf. on Communication, Control, and Computing*, Univ. of Illinois at Urbana-Champaign, Sept. 1988.

[2] Special Section on Neural Networks for Systems and Control, *IEEE Contr. Syst. Mag.*, vol. 8, no. 2, Apr. 1988.

[3] K. S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*, New York: Academic Press, 1968.

[4] Z. Kohavi, *Switching and Finite Automata Theory*, New York: McGraw-Hill, 1978.

[5] R. Forsyth and R. Rada, *Machine Learning*, New York: Wiley, 1986.

[6] R. Wilson, "Is Vision a Pattern Recognition Problem?," in *Pattern Recognition, Lecture Notes in Computer Science 301*, J. Kittler, ed., Springer-Verlag, 1988.

[7] S. C. Shapiro, ed., *Encyclopedia of Artificial Intelligence, Volume 2*, New York: Wiley, 1987.

[8] B. Chandrasekaran and A. Goel, "From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 3, pp. 415-424, May/June 1988.

[9] P. J. Antsaklis, K. M. Passino, and S. J. Wang, "Autonomous Control Systems: Architecture and Fundamental Issues," *Proc. Amer. Contr. Conf.*, Atlanta, GA, pp. 602-607, June 1988.

[10] P. J. Antsaklis, K. M. Passino, and S. J. Wang, "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues," to appear in *Journal of Intelligent and Robotic Systems*.

[11] J. F. Knight and K. M. Passino, "Decidability for Temporal Logic in Control Theory," *Proc. 25th Annual Allerton Conf. on Communication, Control, and Computing*, Univ. of Illinois at Urbana-Champaign, vol. I, pp. 335-344, Sept. 1987.

[12] P. Peleties and R. DeCarlo, "Modeling of Interacting Continuous Time and Discrete Event Systems: An Example," *Proc. 26th Allerton Conf. on Communication, Control, and Computing*, Univ. of Illinois at Urbana-Champaign, Sept. 1988.

[13] D. T. Tank and J. J. Hopfield, "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," *IEEE Trans. Circ. Syst.*, vol. CAS-33, no. 5, pp. 533-541, May 1986.

[14] D. L. Gray, *Synthesis Procedures for Neural Networks*, Master's Thesis, Dept. of Electrical and Computer Eng., Univ. of Notre Dame, Notre Dame, IN, July 1988.

[15] K. J. Astrom et al., "Expert Control," *Automatica*, vol. 22, no. 3, pp. 277-286, 1986.

[16] Battelle Columbus Div., *Guidelines for Hazard Evaluation Procedures*, American Institute of Chemical Engineers, NY, 1985.

[17] R. E. Barlow et al., *Reliability and Fault Tree Analysis*, SIAM, Philadelphia, PA, 1975.

[18] R. L. Harvey, "Multi-Layer Perceptron Theory," Dept. of Electrical and Computer Eng. Seminar, Univ. of Notre Dame, Notre Dame, IN, Mar. 1988; also in "An Introduction to Multi-layer Perceptrons," Proj. Rept. NN-1, MIT Lincoln Lab, Nov. 1988.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, D. E. Rumelhart and J. L. McClelland, eds., Cambridge, MA: MIT Press, 1986.

[20] R. P. Lippmann, "An Introduction to Computing with Neural Networks," *IEEE ASSP Mag.*, pp. 4-22, Apr. 1987.

[21] W. Y. Huang and R. P. Lippmann, "Neural Net and Traditional Classifiers," presented at the IEEE Conference on Neural Information Processing Systems—Natural and Synthetic, Denver, CO, Nov. 1987.

[22] S. C. Huang, "Supervised Learning with a Selective Update Strategy for Artificial Neural Networks," Master's Thesis, Dept. of Electrical and Computer Engineering, Univ. of Notre Dame, Notre Dame, IN, Aug. 1988.

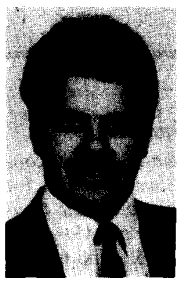
[23] Y. Shrivatava and S. Dasgupta, "Convergence Issues in Perceptron-Based Adaptive Neural Network Models," *Proc. 25th Annual Allerton Conf. on Communication, Control, and Computing*, Univ. of Illinois at Urbana-Champaign, vol. II, pp. 1133-1141, Sept. 1987.

[24] A. Johnson, *Process Dynamics Estimation and Control*, London: Peregrinus, 1985.

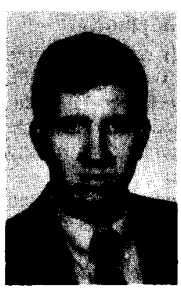
[25] R. K. Miller and A. N. Michel, *Ordinary Differential Equations*, New York: Academic Press, 1982.

[26] D. A. Waterman, "Generalization Learning Techniques for Automating the Learning of Heuristics," *Artif. Intellig.*, vol. 1, pp. 121-170, 1970.

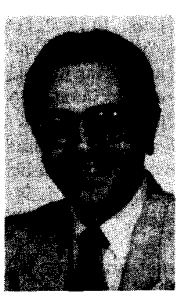
[27] M. E. Soklic, "Adaptive Model for Decision Making," *Pattern Recog.*, vol. 15, no. 6, pp. 485-498, 1982.



Kevin M. Passino was born in Fort Wayne, Indiana, in 1961. He received the B.S. degree from Tri-State University, Angola, Indiana, in 1983, and the M.S. degree from the University of Notre Dame in 1984. Both degrees are in electrical engineering. He has worked at Magnavox Electronic Systems in Fort Wayne and at McDonnell Aircraft, St. Louis, Missouri, on research in flight control. He is currently a Ph.D. candidate in electrical and computer engineering at the University of Notre Dame. His research interests include discrete-event systems, temporal logic, neural networks, and autonomous systems.



Michael A. Sartori was born in St. Louis, Missouri, in 1965. He received the B.S. degree in 1987 and shall receive the M.S. degree in 1989 in electrical engineering from the University of Notre Dame. He worked for McDonnell Douglas Electronics during the summers of 1986 and 1987. He is currently in the Ph.D. program in electrical and computer engineering at the University of Notre Dame. His research interests include autonomous systems, neural networks, discrete-event systems, and digital image processing.



Panos J. Antsaklis received the diploma in mechanical and electrical engineering from the National Technical University of Athens, Greece, in 1972, and the M.S. and Ph.D. degrees in electrical engineering from Brown University, Providence, Rhode Island, in 1974 and 1977, respectively. After holding faculty positions at Brown University, Rice University, and Imperial College, University of London, he joined the University of Notre Dame, where he is currently an Associate Professor in the Department of Electrical and Computer Engineering. In

summer 1986, he was a NASA Faculty Fellow at the Jet Propulsion Laboratory, Pasadena, California. He was a Senior Visiting Scientist at the Laboratory for Information and Decision Systems of the Massachusetts Institute of Technology during his sabbatical leave in 1987. His research interests are in multivariable system and control theory, and in discrete-event systems, autonomous systems, and neural networks. Dr. Antsaklis has served as Associate Editor of the *Transactions on Automatic Control*, and he is currently Chairman of the Technical Committee on Theory of the IEEE Control Systems Society.

News

Call for Volunteers

The IEEE Control Systems Society is considering candidates for Associate Editors for the *IEEE Transactions on Automatic Control* and the *IEEE Control Systems Magazine*. We are also looking for people who wish to take an active part in Technical Committees and Working Groups and other Society committees. If you are interested in serving the Control Systems Society in that capacity or some other capacity, please send a cover letter along with your resume to the Society President-Elect: Prof. William S. Levine, Department of Electrical Engineering, University of Maryland, College Park, MD 20742; phone: (301) 454-6841.

John E. Ward Memorial

John E. Ward, recently retired from the Massachusetts Institute of Technology (MIT), died unexpectedly on December 11, 1988. Throughout a distinguished career, he made many contributions to automatic control and to the control profession.

John Ward authored numerous reports and papers, contributed to several books and the *Encyclopedia Britannica,* and was coholder of a number of patents in the areas of control and display. He helped organize the Professional Group on Automatic Control of the Institute of Radio Engineers in 1954, was its Chairman from 1958 to 1960, and served as President of the American Automatic Control Council from 1963 to 1965. He was a Fellow of the IEEE (1968), was named a Distinguished Member of the IEEE Control Systems Society in 1983, and was a recipient of the IEEE Centennial Medal in 1984.

He was born in Toledo, Ohio, in 1920. He attended MIT, where he received the B.S. and M.S. degrees in electrical engineering in 1943 and 1947, and he has been associated with MIT since that time. At the end of the war, he joined the MIT Servomechanisms Laboratory; he became Laboratory Executive Officer in 1955, Assistant Director in 1959, and served as Deputy Director from 1967 through 1973.

John Ward wrote the history article entitled "Predecessors of the IEEE Control Systems Society," which appeared in the February 1987 issue of the *IEEE Control Systems Magazine,* and his biography appears in the October 1987 issue of the *Magazine,* in honor of his retirement from MIT. We will miss him as a colleague and as a friend.

Suri Receives Award

Rajan Suri has received an award of $50,000 from Ford Motor Company "in recognition of outstanding contributions made to the field of perturbation analysis of discrete-event systems." Professor Suri developed the first videocassette under the auspices of the Control Systems Society for the IEEE Home Video Tutorials, "Analysis and Modeling of Modern Manufacturing Systems." He is also one of a team of people from his university who received the 1988 LEAD (Leadership and Excellence in Application and Development) Award from the Society of Manufacturing Engineers. In July 1988, Suri was promoted to Full Professor of Industrial Engineering at the University of Wisconsin-Madison.

1990 ACC

Plans are proceeding for the 1990 American Control Conference (ACC), which will be held several weeks earlier than usual, May 23-25, 1990, at the Sheraton Harbor Island Hotel in San Diego, California. The General Chairman is Dagfinn Gangsaas from Boeing Advanced Systems and the Program Chairman is Eliezer Gai from Draper Laboratory. A call for papers will appear in the August 1989 issue of the *Magazine.* For more information, contact: Dagfinn Gangsaas, 1990 ACC Chair, Boeing Advanced Systems, MS 33-12, P.O. Box 3707, Seattle, WA 98124-2207, phone: (206) 241-4348.

Software in Control Education

The International Federation of Automatic Control has recently established a new Working Group on the theme "Teachware for Control." One of the major goals of this group is to make a basic set of programs in control engineering available to every student. A first prototype of such a software package has been prepared with the help of many institutions and individuals at the IDA Center of ETH Zürich. It consists of some examples of training programs, basic simulation tools, and the public-domain version of MATLAB (C. Moler), and is available for Macintosh computers and IBM compatibles. A similar set consisting of XLISP (D. Betz), PROLOG (ABB), and small expert system shells is also available for introductory teaching of artificial intelligence techniques. For further information, contact: Prof. W. Schaufelberger, Projekt-Zentrum IDA, ETH-Zentrum, CH-8092 Zürich, Switzerland.

Control Software for Netlib

As one of his activities as Chairman of the IEEE Control Systems Society Technical Committee on Computer-Aided Control Systems Design (CACSD), Doug Birdwell is setting up a sublibrary of the Netlib facility at the Argonne National Laboratory. If you have software that you believe can benefit a reasonably significant segment of the control systems community and would be willing to place the software in the public domain and distribute it through Netlib, Professor Birdwell will recommend the software's inclusion in Netlib with the understanding that the intent is strictly to provide a collection and distribution service for Society members.

Authors of software will benefit by receiving recognition for their software through its use. In addition, the expanded use of the software is highly likely to highlight limitations of the implementation or the theory on which the software is based, providing the author valuable feedback and an incentive to improve the software. The Society membership will benefit in general from the fast and free availability of software relevant to control systems problems.

The Society Technical Committee on CACSD is offering to maintain the Society sublibrary of Netlib as a service; however, there are a few disclaimers. First, we will not serve as a reviewer of any of the software; whatever representations the authors make about the software are included in the library, but no testing is done. Therefore, we make no warranties as to the utility of the software. Second, we will accept only software that is placed in the public domain by its authors and has no restrictions or limitations on its use or consequent distribution. Only the author of the software, or the editor of a collection of software, may submit the software for inclusion in Netlib.

If you are interested in contributing software, please contact: Prof. J. Doug Birdwell, Dept. of ECE, Ferris Hall, University of Tennessee, Knoxville, TN 37996-2100. His e-mail address can be any one of the following. (All will end up at the first address, so don't send multiple messages.)
birdwell@cascade.engr.utk.edu
birdwell@utkvx.bitnet
jdb@msr.epm.ornl.gov