N91-20814

A Space-Time Neural Network

James A. Villarreal and Robert O. Shelton

Software Technology Branch/PT4
Information Systems Directorate
Lyndon B. Johnson Space Center
Houston, Texas 77058

ABSTRACT

Neural network algorithms have impressively demonstrated the capability of modelling spatial information. On the other hand, the application of parallel distributed models to processing of temporal data has been severely restricted. This work introduces a novel technique which adds the dimension of time to the well known backpropagation neural network algorithm. The paper cites several reasons why the inclusion of automated spatial and temporal associations are crucial to effective systems modelling. An overview of other works which also model spatiotemporal dynamics is furnished. In addition, a detailed description of the processes necessary to implement the space-time network algorithm is provided. The reader is given several demonstrations which illustrate the capabilities and performance of this new architecture.

INTRODUCTION

Throughout history, the meaning of time has plagued the minds of mankind. The wise Greek philosophers, Socrates, Plato, and Aristotle pondered deeply with what the influence of time had on human knowledge. The English poet, Ben Johnson, wrote "O for an engine to keep back all clocks" giving voice to our ageless lament over the brevity of human life. The great scientist, Einstein, who developed the theory of relativity, believed that space and time cannot be considered separately, but that they depend upon one another.

A need for space-time knowledge capture representation is clearly evident. Human cognitive thought processes involve the use of both space and time. A childhood event is remembered by an occurrence (or space) and its associated place in time. We speak of an event which occurred a specific time ago. Linguistic meanings are expressed in a manner in which proper temporal order plays a crucial role in the conveyance of a concept. Take, for example, the phrases "house cat" and "cat house". Speech production, too, is very order dependent -- subtleties in intonations may change the whole meaning of a concept. The more advanced engineering systems have characteristics which vary over time. For instance, complex machines such as the Space Shuttle Main Engine are abound with sensors, each varying over the life of the machine's operation. A model which is capable of automatically associating spatial information with its appropriate position in time becomes increasingly significant.

Also, microscopic level investigations reveal a need to incorporate time or sequence discovery and adaptation into the modelling framework. It is clearly

evident that information exchange at the neuronal level occurs through a rich interchange of complex signals. Freeman [3] and Baird [1] have done extensive research on the olfactory bulb at anatomical, physiological, and behavioral levels. Their findings have shown that information in biological networks takes the form of space-time neural activity patterns. These dynamic space-time patterns encode past experience, attempt to predict future actions, and are unique to each biological network.

As seen in figure 1, the "classical" neuron has several dendrites which receive information from other neurons. The soma or cell body performs a wide range of functions; it processes information from the dendrites in a manner which is not entirely understood and also maintains the cell's health. The information processed by the neuron is distributed by its axon to other interconnected neurons by the propagation of a spike or action potential. Along each dendrite are thousands of protrusions where neurons exchange information through a process known as synapse. The synaptic cleft releases chemicals known as neurotransmitters. Even at this microscopic level, the relevance for time adaptive neural networks becomes clearly evident. Synaptic clefts take on roles such as neurotransmitter modulators, generators, and filters which cloud the neuron's inner workings and render these ever changing dynamical properties especially difficult to study.

Connectionist architectures have impressively demonstrated several models of capturing spatial knowledge. To accomplish this, the most popular solution has been to distribute a temporal sequence by forcing it into a spatial representation. This approach has worked well in some instances [11]. But there are problems with this approach and it has ultimately prove inadequate.

A REVIEW OF NEURAL NETWORKS

A network is comprised of numerous, independent, highly interconnected processing elements. For backpropagation networks, each element can be characterized as having some *input* connections from other processing elements and some *output* connections to other elements. The basic operation of an element is to compute its *activation value* based upon its inputs and send that value to its output elements. Figure 2 shows a schematic of a processing element. Note that this element has j input connections coming from j input processing elements. Each connection has an associated value called a *weight*. The output of this processing element is a non-linear transform of its summed, continuous-valued inputs by the sigmoid transformation in (1) and (2). Understanding the details of this transformation is not essential here; the interested reader will find an excellent description of such details provided by Rummelhart et. al.[8].

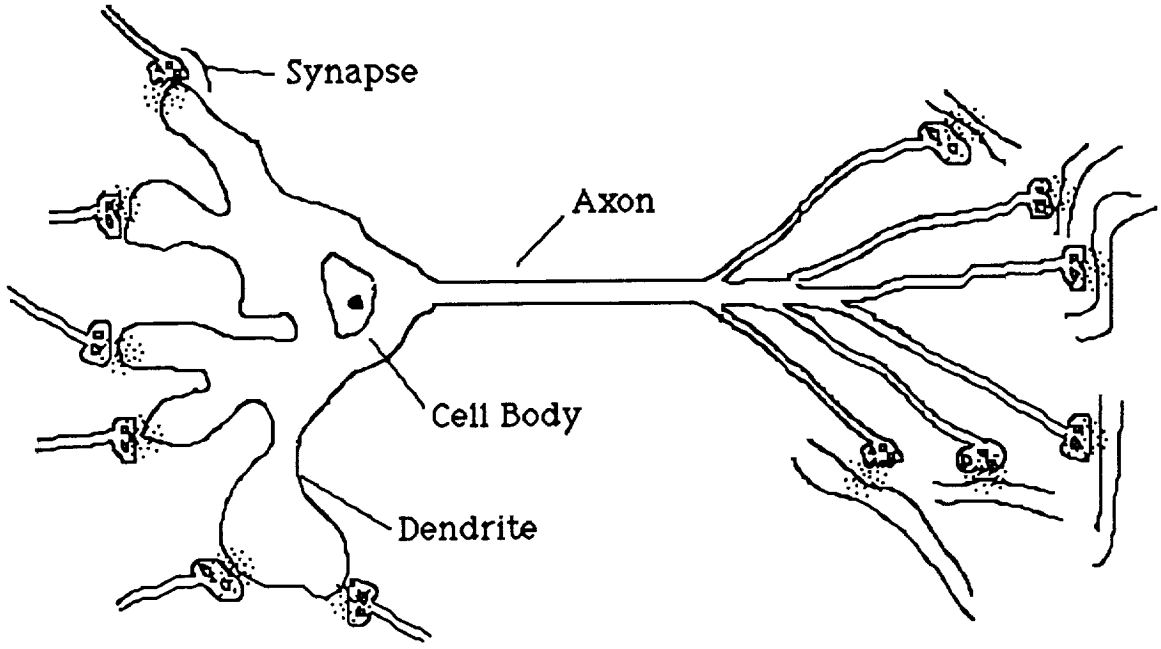

Figure 1 .... Schematized diagram of the classical neuron.

When groups of processing elements are arranged in sequential layers, each layer interconnected with the subsequent layer, the result is a wave of activations propagated from the input processing elements, which have no incoming connections, to the output processing elements. The layers of elements between the inputs and outputs take on intermediate values which perform a mapping from the input representation to the output representation. It is from these intermediate or *hidden* elements that the backpropagation network draws its generalization capability. By forming transformations through such intermediate layers, a backpropagation network can arbitrarily categorize the features of its inputs.

$$E_i = \Sigma w_{ij} p_j \tag{1}$$

$$p_i = P(E_i) = \frac{1}{1 + e^{-E_i}} \tag{2}$$

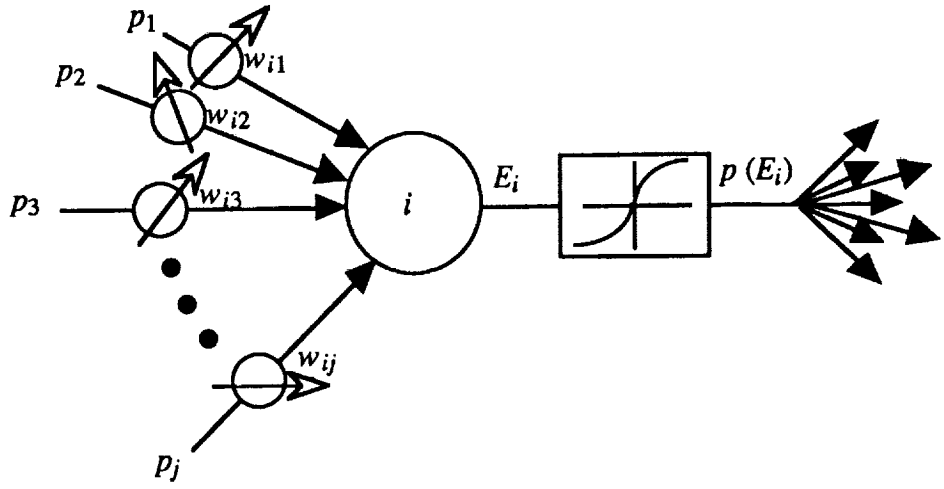

Figure 2 .... Processing element in a backpropagation network.

THE WEIGHTS OF A BACKPROPAGATION NETWORK

The heart of the backpropagation algorithm lies in how the values of its interconnections, or weights, are updated. Initially, the weights in the network are set to some small random number to represent no association between processing elements. Upon being given a set of patterns representing pairs of input/output associations, the network enters what is called a *training* phase. During training, the weights are adjusted according to the learning technique developed by Rumelhart et. al. The training phase is modelled after a behavioristic approach which operates through reinforcement by negative feedback. That is, the network is given an input from some input/output pattern for which it generates an output by propagation. Any discrepancies found when comparing the network's output to the desired output constitute mistakes which are then used to alter the network characteristics. According to Rumelhart's technique, every weight in the network is adjusted to minimize the total mean square errors between the response of the network, $p_{pi,}$ and the desired outputs, t_{pi} ,to a given input pattern. First, the error signal, δ_i, is determined for the output layer, N:

$$\delta_i^{(N)} = (t_i - p_i^{(N)}) P'(E_i^{(N)}) \qquad (3)$$

The indices *p* and *i* represent the pattern number and the index to a node respectively. The weights are adjusted according to:

$$\Delta w_{ij}^{(n+1)} = \alpha \Delta w_{ij}^{(n)} + \eta \delta_i^{(n+1)} P_j^{(n)} \qquad (4)$$

where $\Delta w_{ij}^{(n)}$ is the error *gradient* of the weight from the *j* th processing element in layer *n* to the *i* th unit in the subsequent layer *(n + 1)*. The parameter α, performs a damping effect through the multi-dimensional error space by relying on the most recent weight adjustment to determine the present

adjustment. The overall effect of this weight adjustment is to perform a gradient descent in the error space; however, note that true gradient descent implies infinitesimally small increments. Since such increments would be impractical, η is used to accelerate the learning process. In general, then, the errors are recursively back propagated through the higher layers according to:

$$\delta_i^{(n)} = \sum_j \delta_j^{(n+1)} w_{ji}^{(n)} P'(E_i^{(n)}) \tag{5}$$

where *P'(E)* is the first derivative of *P(E)*.

OTHER SPATIOTEMPORAL NEURAL NETWORK ARCHITECTURES

Advances in capturing spatial temporal knowledge with neural network architectures have been made by Jordan[4] and Elman[2]. Jordan approaches this problem by partitioning the input layer in a connectionist network into separate plan and state layers. In essence, Jordan's architecture acts as a backpropagation network, except for the specialized processing elements in the state layer, which receive their inputs from the output units, as well as from recurrent connections which allow the state layer elements to "remember" the network's most recent state. In other words, the state units behave as pseudo inputs to the network providing a past-state history. Here, a recurrent connection is one in which it is possible to follow a path from an element back onto itself as shown in figure 3. Recurrent networks of this type allow the element's next state to be not only dependent on external inputs, but also on the state of the network at its most previous time step. For further discussion of this network's operation refer to Jordan. In general, however, this network is trained to reproduce a predetermined set of sequence patterns from a static input pattern. One of the authors (Villarreal), used this network architecture extensively in developing a speech synthesizer. The inputs to the speech synthesis network represented a tri-phoneme combination and the output was partitioned to represent the various vocal tract components necessary to produce speech. I.e., the output layer in the speech synthesis neural network consisted of the coefficients to a time-varying digital filter, a gain element, and a pitch element which excited the filter, and a set of down-counting elements where each count represented a 100 millisecond speech segment. To train a single tri-phone set, the network was first reset by forcing the activation value of the elements in the state layer to zero, then a tri-phone pattern was presented to the network's input and held there during the learning process while the outputs changed to produce the appropriate output characteristics for that particular tri-phone combination. The outputs would represent the transition from one phoneme to another while a smooth transition in pitch, gain, and vocal tract characteristics would take place. The process was then repeated for other tri-phone combinations.

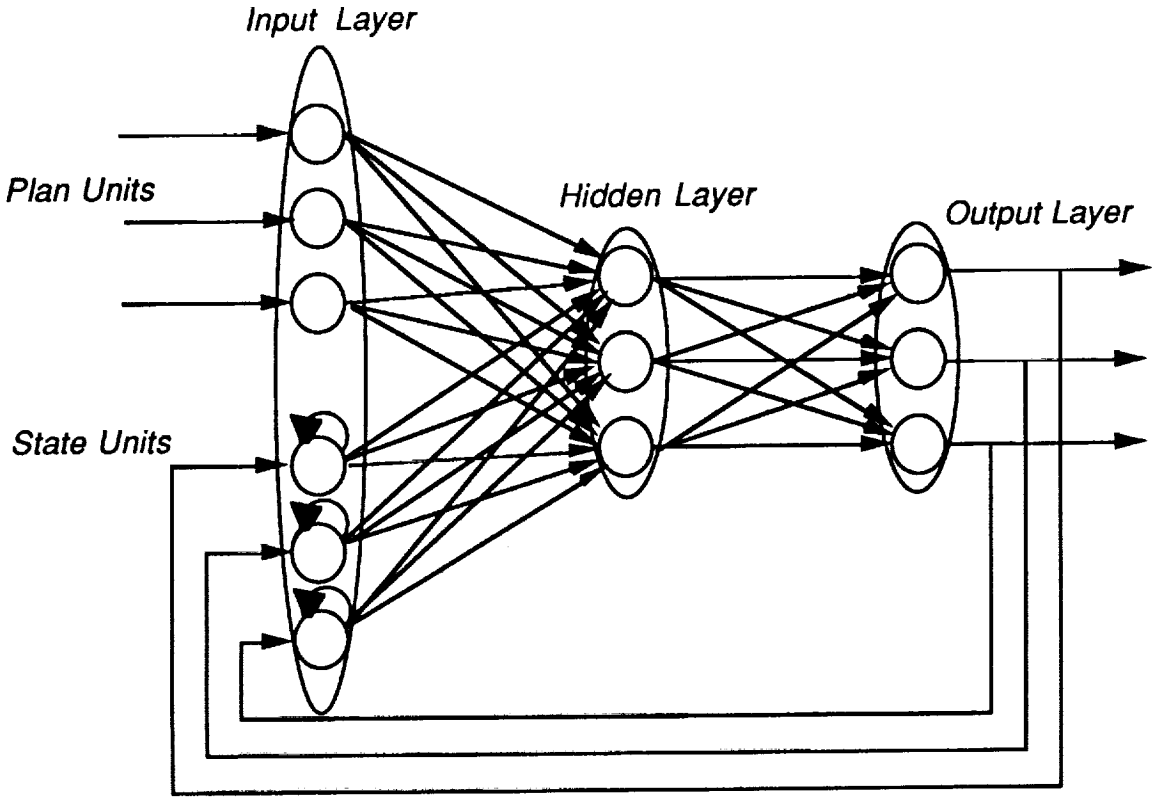

Figure 3 .... The connection scheme for Jordan's network architecture which learns to associate a static input with an output sequence.

Elman modifies Jordan's approach by constructing a separate layer, called the *Context Layer,* which is equal in size to the number of units in the hidden layer (see figure 4). However, the context units receive their input along a one-to-one connection from the hidden units, instead of from the output units as described by Jordan. The process works as follows. Suppose there is a sequential pattern to be processed. Initially, the activation values in the context units are reset to a value midway between the upper and lower bounds of a processing element's activation value, indicating ambiguous or don't care states. A pattern is presented to the network's input, forward propagating the pattern toward the output. At this point, the hidden layer activation levels are transferred one-for-one to elements in the context layer. If desired, error backpropagation learning can now take place by adjusting the weights between output and hidden, hidden and input, and hidden and context layers. The recurrent connections from the hidden to context layers are not allowed to change. At the next time step, the network's previous state is encoded by the activation levels in the context units. Thus, the context layer provides the network with a continuous memory.

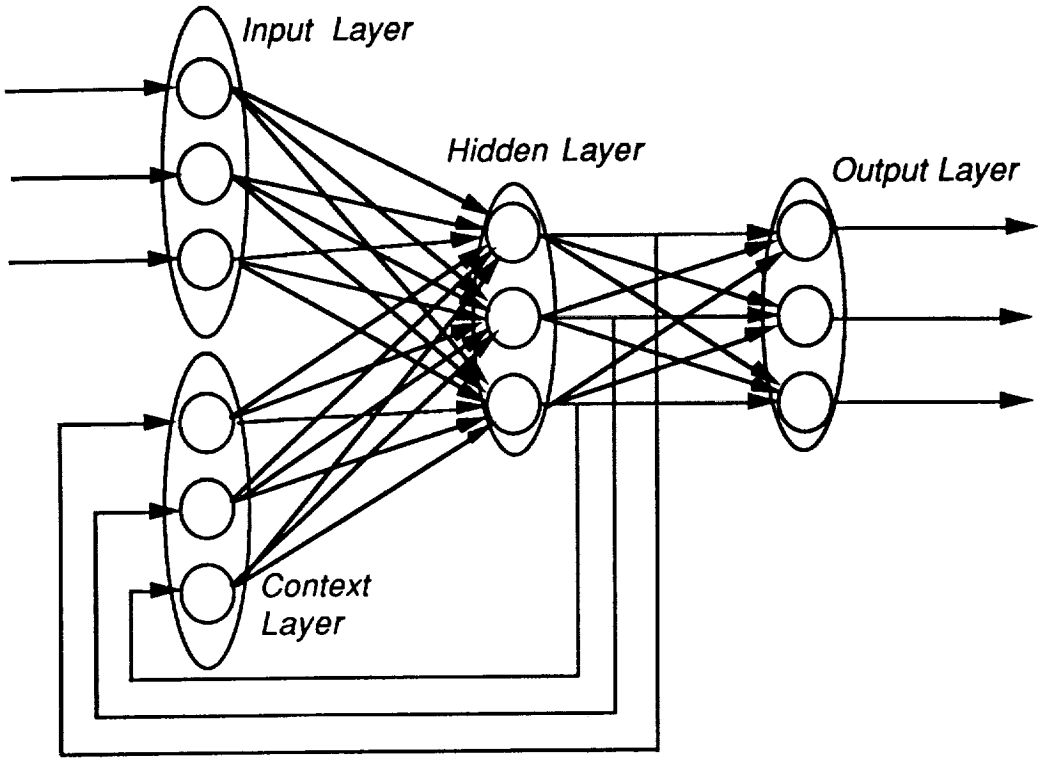

Figure 4 .... With the Elman network, a history of the network's most previous state is stored by transferring the activations in the hidden layer to the pseudo input, context layer. Longer term memories are attainable by adding recurrent connections to the context units.

DESCRIPTION OF THE SPACE-TIME NEURAL NETWORK

Another dimension can be added to the processing element shown in figure 2 by replacing the synaptic weights between two processing elements with an adaptable-adjustable filter. Instead of a single synaptic weight which with the standard backpropagation neural network represented the association between two individual processing elements, there are now several weights representing not only association, but also temporal dependencies. In this case, the synaptic weights are the coefficients to adaptable digital filters. The biological implication of this representation can be seen when one considers that synapses undergo a refractory period -- responding less readily to stimulation after a response. Before proceeding with a description of the space-time network, it is important to introduce digital filter theory and some nomenclature.

DIGITAL FILTER THEORY REVIEW

Linear difference equations are the basis for the theory of digital filters. The general difference equation can be expressed as:

$$y(n) = \sum_{k=0}^{N} b_k x(n-k) + \sum_{m=1}^{M} a_m y(n-m) \quad (6)$$

Where the x and y sequences are the input and output of the filter and a_m's and b_k's are the coefficients of the filter. Sometimes referred to as an s-transform, the well known continuous domain Laplace transform is an extremely powerful tool in control system design because of its capability to model any combination of direct current (DC) or static levels, exponential, or sinusoidal signals and to express those functions algebraically. The s-plane is divided into a damping component (σ) and a harmonic component ($j\omega$) and can mathematically be expressed as

$$s = e^{-(\sigma + j\omega t)} \quad (7)$$

This formulation has several interesting characteristics which should be noted:

- The general Laplace transfer function can be thought of as a rubber sheet on the s-plane. A desirable transfer function is molded by strategically placing a transfer function's roots of the numerator and the denominator in their appropriate positions. In this case, polynomial roots of the numerator are referred to as zeros and "pin" the rubber sheet to the s-plane's ground. On the other hand, polynomial roots of the denominator are referred to as poles and their locations push the rubber sheet upwards -- much like the poles which hold up the tarpaulin in a circus tent. Therefore, zeros null out certain undesirable frequencies and poles can either generate harmonic frequencies (if close enough to the $j\omega$ axis) or allow certain frequencies to pass through the filter.
- Setting the damping coefficient , σ, to zero is effectively similar to taking a cross sectional cut along the $j\omega$ axis. This is the well known Fourier transform.
- A pole on the $j\omega$ axis, signifying no damping, will produce a pure sinusoidal signal. However, a pole which travels onto the left half plane of the s-plane exponentially increases, eventually sending the system into an unstable state.

The discretized form of the Laplace transform has been developed further and is referred to as the z-transform. The notation z^{-1} is used to denote a delay equal to one sampling period. In the s-domain, a delay of T seconds corresponds to e^{-sT} . Therefore, the two variables s and z are related by:

$$z^{-1} = e^{-sT} \quad (8)$$

where T is the sampling period. The mapping between the variables can be further illustrated by referring to figure 5. First notice that the left half plane of the s-plane maps to the area inside a unit circle on the z-plane. In abiding with the Nyquist criterion, sampling at least twice the signal bandwidth, f_S, note that as one traverses from $-f_S/2$ to $+f_S/2$ on the s-plane is equivalent to going from π radians toward 0 radians and back to π radians in a counterclockwise direction

on the z-plane. Furthermore, note that lines in the s-plane map to spirals in the z-plane.

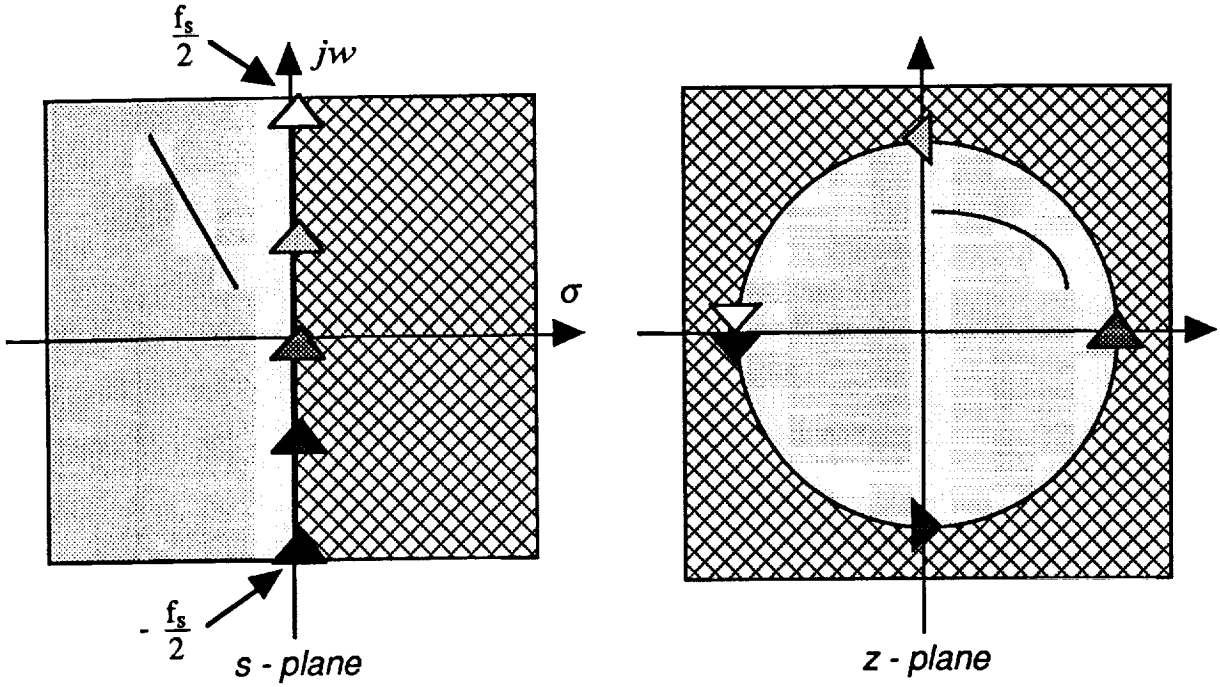

Figure 5 .... Pictorial relationship between the continuous domain s-plane and discrete domain z-plane.

By evaluating the z-transform on both sides of the linear difference equation, we can show that

$$F(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 - \sum_{m=1}^{M} a_m z^{-m}} \qquad (9)$$

Digital filters are classified into recursive and nonrecursive types. The nonrecursive type have no feedback or recurrent paths and as such all the a_m terms are zero. Furthermore, digital filters are also classified in terms of their impulse responses. Because nonrecursive filters produce a finite number of responses from a single impulse, such filters are referred to as Finite Impulse Response (FIR) filters. On the other hand, the recursive filters produce an infinite number of responses from an impulse and are therefore referred to as Infinite Impulse Response (IIR) filters. For example, if a unit impulse is clocked through the filter shown in figure 6(a), the sequence

$$b_0, b_1, b_2, ...b_M, 0, 0, 0, 0, 0, ...0, 0, 0$$

will be output. Notice that the filter produces only the coefficients to the filter followed by zeroes. However, if a unit impulse is presented to the filter shown in

figure 6(b), because of the recursive structure, the response is infinite in duration.

FIR and IIR filters each possess unique characteristics which may make one more desirable over another depending on the application. To summarize, the most notable of these characteristics include:

- FIR filters, because of their finite duration are not realizable in the analog domain. IIR filters, on the other hand, have directly corresponding components in the analog world such as resistors, capacitors, and inductive circuits.
- IIR filters cannot be designed to have exact linear phase, whereas FIR filters have this property.
- Because of their recursive elements, IIR filters are orders of magnitude more efficient in realizing sharp cutoff filters than FIR filters.
- Because of their nonrecursiveness, FIR filters are guaranteed stable. This property makes FIR filters much easier to design than FIR filters.

These different properties between FIR and IIR filters must be carefully weighed in selecting the appropriate filter for a particular application.

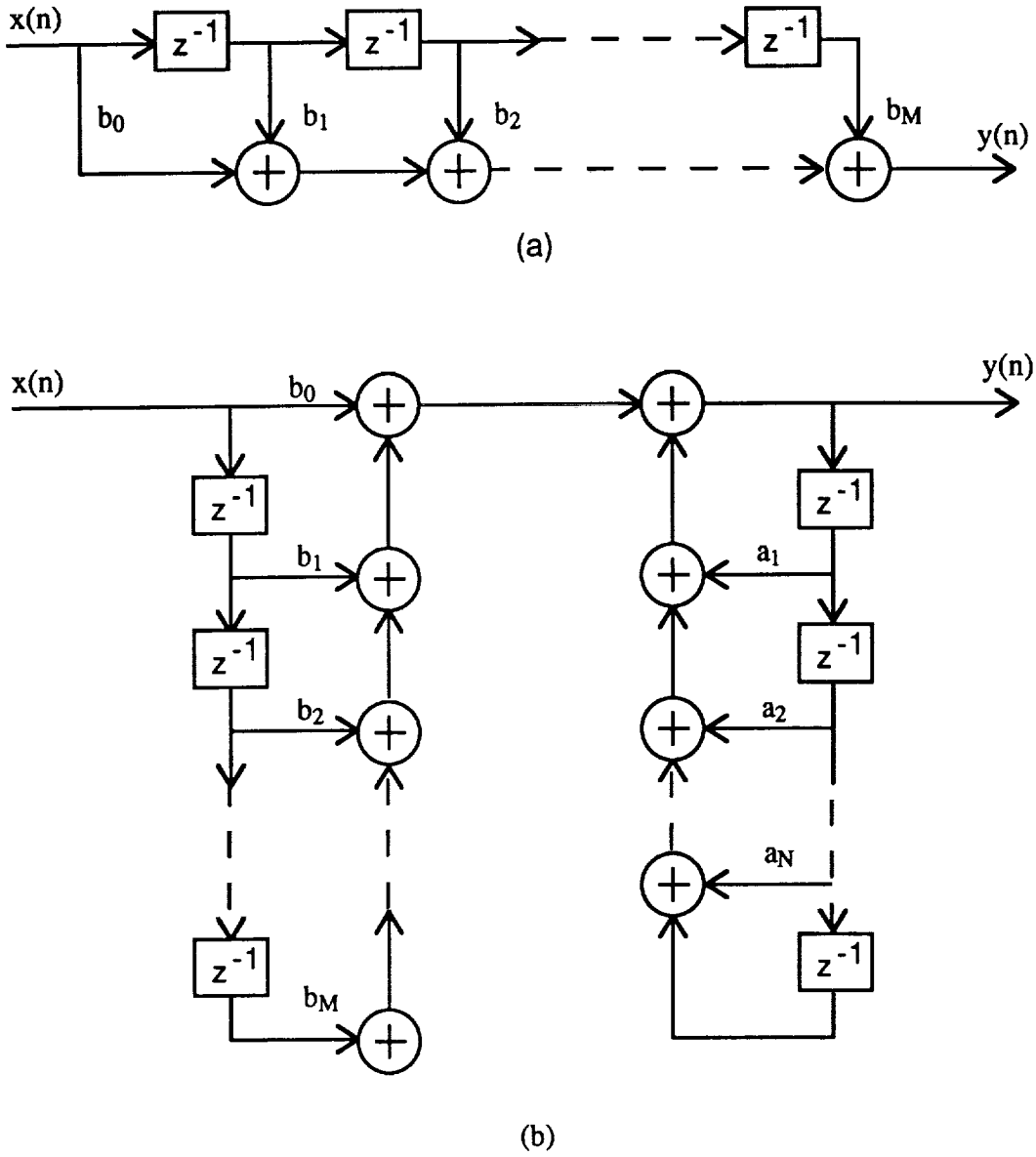

Figure 6 .... (a) Digital network for FIR system; (b) Digital network for IIR system

DESCRIPTION OF THE SPACE-TIME NEURAL NETWORK - CONTINUED

Having introduced digital filter theory, it is now possible to continue with the description of the space-time neural network. What follows is a detailed procedure for constructing and training the space-time neural network. As mentioned earlier, the space-time neural network replaces the weights in the standard backpropagation algorithm with adaptable digital filters. The procedure involves the presentation of a temporal ordered set of pairs of input and output vectors. A network consisting of at least two layers of adaptable digital filters buffered by summing junctions which accumulate the contributions from the subsequent layer is required. A pictorial representation of the space-

time processing element is illustrated in figure 7. In this case, a value, say $x_j(n)$, is clocked in to its associated filter, say $H_{ji}(n)$, producing a response $y_j(n)$ according to the filter representation

$$y_j(n) = \sum_{m=1}^{M} a_{mj} y_j(n - m) + \sum_{k=0}^{N} b_{kj} x_j(n - k) \tag{10}$$

All remaining inputs are also clocked in and accumulated by the summing junction i:

$$S_i(n) = \sum_{\text{all } j} y_j(n) \tag{11}$$

The contributions from the signals fanning in to the summing junction are then non-linearly transformed by the sigmoid transfer function

$$p_i(S_i(n)) = \frac{1}{1 + e^{-S_i(n)}} \tag{12}$$

This output is then made available to all filter elements connected downstream.

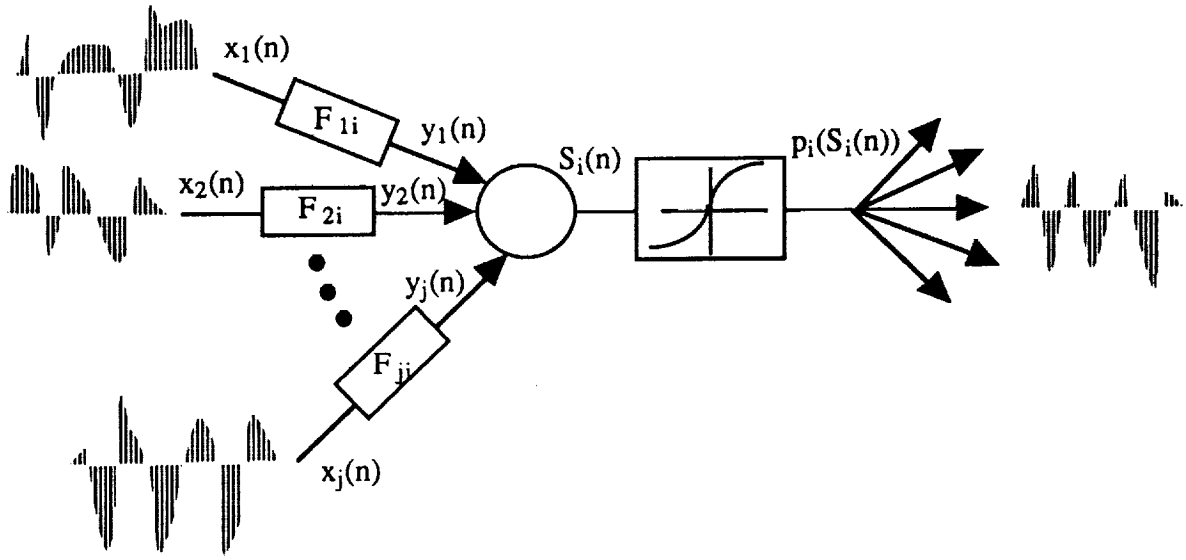

Figure 7 .... A pictorial representation of the Space-Time processing element.

It was earlier discussed that the space-time network is comprised of at least two layers of filter elements fully interconnected and buffered by sigmoid transfer nodes at the intermediate and output. A sigmoid transfer function is not used at the input. Forward propagation involves presenting a separate sequence dependent vector to each input, propagating those signals throughout the intermediate layers as was described earlier until reaching the output processing elements. In adjusting the weighting structure to minimize the error for static networks, such as the standard backpropagation, the solution is straightforward. However, adjusting the weighting structure in a recurrent network is more complex because not only must present contributions be accounted for but contributions from past history must also be considered.

Therefore, the problem is that of specifying the appropriate error signal at each time and thereby the appropriate weight adjustment of each coefficient governing past histories to influence the present set of responses.

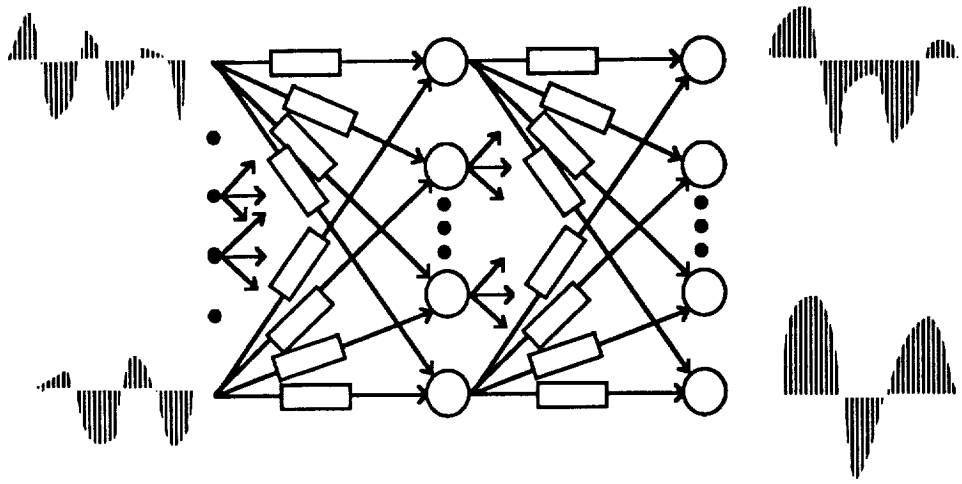

Figure 8 .... A representation of a fully connected network utilizing Space-Time processing elements. This depicts a set of input waveform sequences mapped into an entirely different output waveform sequence.

The procedure for adjusting the weighting structure for the space time network follows. First compute the errors at the output layer for each processing element, *i*,

$$\delta_i = (D_i(k) - A_i(k))\, P'(E_i(k)) \tag{13}$$

where:

$D_i(k)$	is the *kth* desired response from a given sequence for neuron i at the output layer
$A_i(k)$	is the network's output response at neuron i for the *kth* input sequence pattern
$P'(E_i(k))$	is the first derivative of the sigmoid function for the *ith* output's activation value or $P(E_i(k))(1 - P(E_i(k))$

Now to compute the updates for the coefficients each filter element between the hidden and output layer neurons, a reversal procedure is implemented. Whereas in the forward propagation, input values were clocked into the filter elements, here, backpropagation instead involves the injection of errors into the filter elements according to:

$$\Delta b_{ijk}(n + 1) = \alpha[\eta \Delta b_{ijk}(n) + (1 - \eta)\delta_i X_{ijk}] \tag{14}$$

where:

$\Delta b_{ijk}(n+1)$	is the update for a zero coefficient, b_k, lying between processing elements i and j
α	is the learning rate
$\Delta b_{ijk}(n)$	is the most recent update for the *kth* zero element between processing elements i and j
η	damps most recent updates
δ_i	is described by (13)
X_{ijk}	contain a history of the activation values for the non-recursive filter elements between neurons i and j, k time steps ago

The recursive components in each filter element are treated the same way and are updated according to:

$$\Delta a_{ijk}(n+1) = \alpha[\eta \Delta a_{ijk}(n) + (1 - \eta)\delta_i Y_{ijk}] \tag{15}$$

where:

$\Delta a_{ijk}(n+1)$	is the update for a zero coefficient, b_k, lying between processing elements i and j
α	is the learning rate
$\Delta a_{ijk}(n)$	is the most recent update for the *kth* zero element between processing elements i and j
η	damps most recent updates
δ_i	is described by (13)
Y_{ijk}	contain a history of the activation values for the non-recursive filter elements between neurons i and j, k time steps ago

For implementation purposes, the present algorithm only considers the accumulation of errors which span the length of the number of zeroes, nz_{ho}, between the hidden and output neurons.

$$\delta_{ik} = \sum_{k=0}^{nz_{ho}} P'(E_{ik}) \sum_j \delta_{jk} X_{jik} \tag{16}$$

where:

i	is the index of the hidden neuron
j	ranges over the neuron indices for the output layer
δ_{hi}	is the accumulated error for the ith neuron in the hidden layer

$P'(A_{ik})$ is the first derivative of the sigmoid function for the *kth* history of activation levels for the ith neuron in the hidden layer

$\sum_j \delta_{jk} X_{jik}$ sums the results of injecting the previously computed errors found in equation (13) through the FIR portion of the filter element, X_{jik}, found between the ith neuron in the hidden layer and the jth neuron in the output layer.

Simulations

The first simulation is a variation of the classic XOR problem. The XOR is of interest because it cannot be computed by a simple two-layer network. Ordinarily, the XOR problem is presented as a two bit input combination of (00, 01, 10, 11) producing the output (0, 1, 1, 0).

This problem can be converted into the temporal domain in the following way. The first bit in a sequence XOR'd with the second bit will produce the second bit in an output sequence, the second bit XOR'd with the third will produce the third in an output sequence, and so on.

Input	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	.....
Output	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	.....

In the simulation, the training data consisted of 100 randomly generated inputs and the outputs constructed in the manner described above. A network was constructed which had 1 input, 6 hidden elements, 1 output unit, 5 zero coefficients and 0 pole coefficients in the input to hidden layer, and 5 zero coefficients and 0 pole coefficients in the hidden to output layer. The task of the network was to determine the appropriate output based on the input stream.

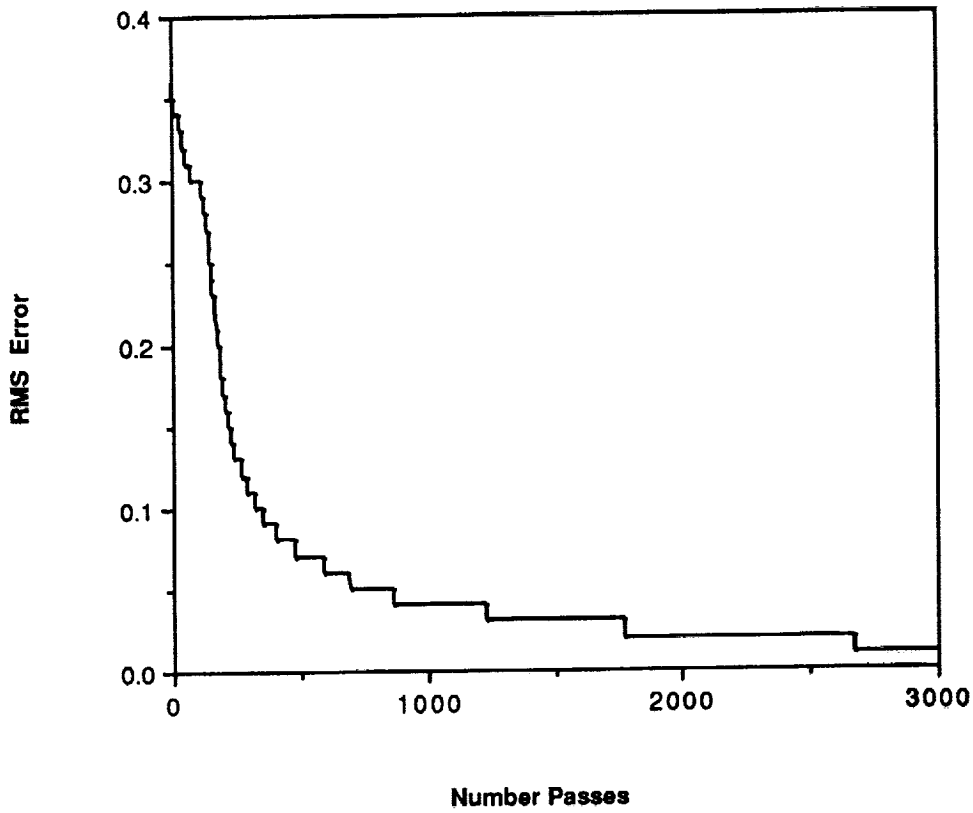

Figure 9 .... Error curve for the temporal XOR problem trained on a 1 input, 5 hidden, 1 output, 5 zeros and 0 poles in input to hidden layer, and 5 zeros and 0 poles in the hidden to output layer.

For the second simulation, a network with 2 input units, 8 hidden units, 8 output units, 5 zeros - 0 poles between input to hidden, and 5 zeros - 0 poles between hidden to output was constructed. One of the authors (Shelton) constructed a problem, called the Time Dependent Associative Memory Test, which would test the network's ability to remember the number of events since the last trigger pattern was presented. The data consisted of 1000 input output pairs where the input bits were randomly constructed and the output appropriately constructed. As an example, consider the first 7 sets of data in the list. Note that a "1" bit sequentially gets added to the output for the input patterns 0 0, 1 0, 1 0, 0 0, 1 0, and 0 1 until the 1 1 pattern is presented which resets the output back to the 1 0 0 0 0 0 0 0 state.

Input	Output
0 0	1 1 0 0 0 0 0 0
1 0	1 1 1 0 0 0 0 0
1 0	1 1 1 1 0 0 0 0
0 0	1 1 1 1 1 0 0 0
1 0	1 1 1 1 1 1 0 0
0 1	1 1 1 1 1 1 1 0
1 1	1 0 0 0 0 0 0 0
1 0	1 1 0 0 0 0 0 0
1 0	1 1 1 0 0 0 0 0
1 1	1 0 0 0 0 0 0 0
1 0	1 1 0 0 0 0 0 0
0 1	1 1 1 0 0 0 0 0
1 1	1 0 0 0 0 0 0 0

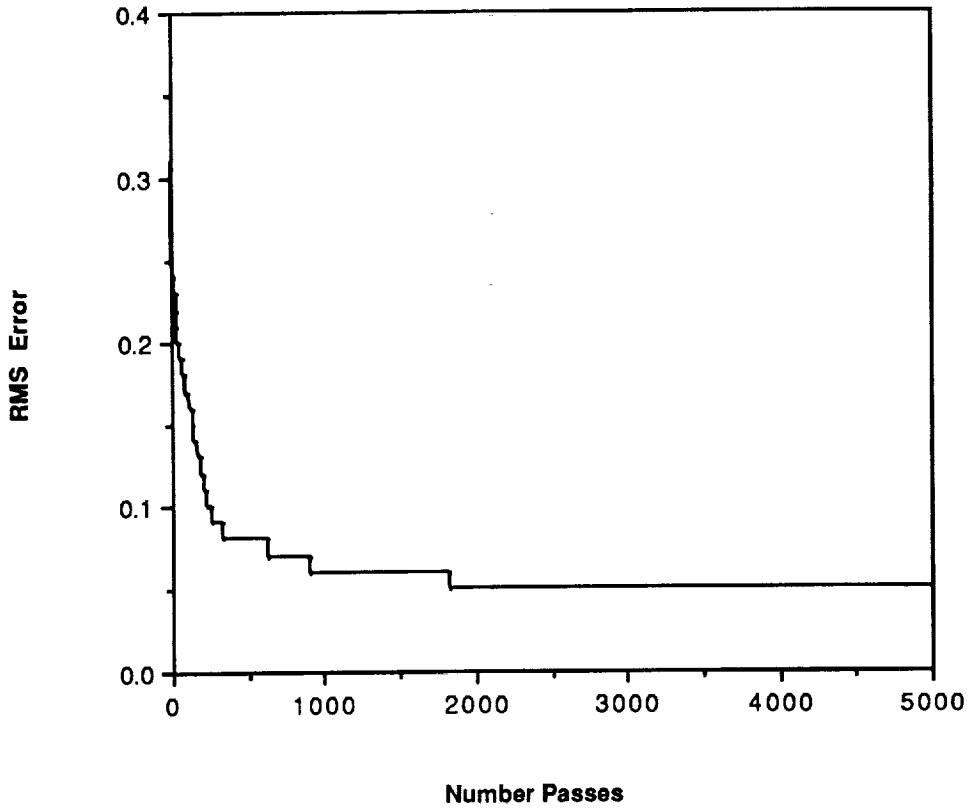

Figure 10 .... Error curve for a 2 input, 8 hidden, 8 output, 5 zero - 0 pole between input to hidden, and 5 zero - 0 pole between hidden to output network operating on the

References

[1] Baird, B., Nonlinear Dynamics of Pattern Formation and Pattern Recognition in the Rabbit Olfactory Bulb, Elsevier Science Publishers B. V., North-Holland Physics Publishing Division, 0167-2789, 1986.

[2] Elman, J. L., Finding Structure in Time, CRL Technical Report 8801, Center for Research in Language, University of California, San Diego, California, 1988.

[3] Freeman, W. J., Why Neural Networks don't yet fly: Inquiry into the Neurodynamics of biological intelligence, IEEE International Conference on Neural Networks, San Diego, California, 1988.

[4] Jordan, M. I., Serial Order: A Parallel Distributed Processing Approach, ICS Report 8604, Institute for Cognitive Science, University of California, San Diego, California, 1986.

[5] Ogata, K., Modern Control Engineering, Prentice-Hall Inc., Englewood Cliffs, NJ, 1970.

[6] Oppenheim, A. V. and Schafer, R. W., *Digital signal processing,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1973.

[7] Rabiner, L. R. and Schafer, R. W., Digital Processing of Speech Signals, Prentice Hall Inc., Englewood Cliffs, NJ, 1978.

[8] Rumelhart, D. E., Hinton, G. E., & Williams, R. J., Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1) (pp. 318-362). Cambridge, Mass.: MIT Press, 1986.

[9] Steiglitz, K., Computer aided design of recursive digital filters, IEEE Transactions on Audio and Electroacoustics 18 (No. 2), pp 123-129, 1970.

[10] Villarreal, J. A. and McIntire, G., Neural Network Based Speech Synthesizer: A Preliminary Report, *Third Conference on Artificial Intelligence for Space Applications,* NASA CP-2492, pp 389-401,1987.

[11] Villarreal, J. A. and Baffes, P., Sunspot Prediction Using Neural Networks, *SOAR'89 - Third Annual Workshop on Automation and Robotics,* 1987.

[12] Yassaie, H., Digital filtering with the IMS A100, IMS A100 Application Note 1, INMOS, 72APP00100, Bristol, UK, 1986.