World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:15, No:2, 2021

Implementation of an Associative Memory Using a Restricted Hopfield Network

Tet H. Yeap

Abstract—An analog restricted Hopfield Network is presented in this paper. It consists of two layers of nodes, visible and hidden nodes, connected by directional weighted paths forming a bipartite graph with no intralayer connection. An energy or Lyapunov function was derived to show that the proposed network will converge to stable states. By introducing hidden nodes, the proposed network can be trained to store patterns and has increased memory capacity. Training to be an associative memory, simulation results show that the associative memory performs better than a classical Hopfield network by being able to perform better memory recall when the input is noisy.

Keywords—Associative memory, Hopfield network, Lyapunov function, Restricted Hopfield network.

I. Introduction

IN 1982, based on studies of collective dynamical computation in neural networks, Hopfield [1], [2], [3] proposed an influential recurrent neural network that has many potential applications such as content addressable memory and optimization engine for the traveling-salesman problem. He formulated an Energy function for the network using the Lyapunov Direct Method showing that the network converges to a stable state if the network has symmetric weights and each network node does not have self-feedback.

Hopfield network comes in two forms: analog or discrete. However, in either form, the network can only be programmed to memorize patterns using Hebbian Rule and has a limited memory capacity of storing 0.15N patterns where N is the number nodes in the network. Many have tried to improve the memory capacity problem and trainability issue of the network [4], [5]. For example, instead of trying to memorize the patterns in one presentation cycle, Gardiner [6], [7], [8] improved the network performance to be an associative memory by presenting the training patterns repeatedly and using the perceptron convergence procedure to train each node to generate the correct state given the states of all the other nodes for a particular training vector.

A typical Hopfield network can only be programmed in one shot to memorize patterns using Hebbian Rule and has a limited memory capacity. An analog restricted Hopfield network (RHN) is proposed in this paper to solve the problem of memory capacity and the trainability issue of the Hopfield network. Similar to a Restricted Boltzmann Machine (RBM) [9], [10], [11], the architecture consists of two layers of nodes, visible and hidden nodes, connected by directional weighted paths. The network is a fully-connected bipartite graph and has no intralayer connection. The visible nodes are classified into either input or output nodes to manage the flow of information to/from the visible nodes.

An energy or Lyapunov function was derived to prove that the proposed network always converges to stable states. When an input vector is presented to the network, the proposed network iterates, sending signals back and forth between the two layers until all its nodes reach an equilibrium state based on the corresponding basin of attraction, generating the desired output vector.

The proposed network can be trained using the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm which was introduced by Spall in 1996 [12], [13]. The algorithm is simple to implement because the gradient of the error function can be estimated using only two final error values measurement.

Deploying hidden nodes and a modified network structure, simulation results show that the proposed network can be trained to store patterns and has increased memory capacity. Acting as an associative memory, simulation results show that the proposed network can be trained to store many images. The proposed network performs better memory recall than a classical Hopfield network when the input is noisy.

II. Hopfield Network

An analog Hopfield network consists of fully interconnected nodes modeled as amplifiers, in conjunction with feedback circuits comprises of wires, resistors, and capacitors, as shown in Fig. 1.

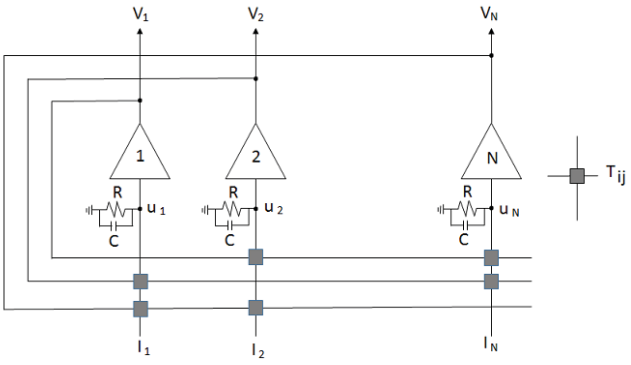

Fig. 1 An analog Hopfield network

The dynamics of the network can be described by the following differential equations:

T. H. Yeap is with the School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Ontario K1N 6N5, Canada (e-mail: tyeap@uottawa.ca).

$$\begin{aligned} \frac{du_i}{dt} &= \sum_{j=1}^{N} T_{ij} V_j - \frac{u_i}{\tau} + I_i \\ \tau &= RC \\ V_i &= g(u_i) \end{aligned} \tag{1}$$

where N is the number of nodes in the network, u_i is the input voltage of the amplifier, T_{ij} is the weight or conductance connecting the output of node j to the input of node i, V_j is the output of node j, RC is the time constant of the network, I_i is the input to node i, and $g()$ is the output function of a node.

The following energy function for the network was derived by Hopfield using the Lyapunov Direct Method if the network has symmetric weights and each network node does not have self-feedback. For the initial-value problem, I_i input is applied to node i at $t = 0$ and then allow the network to evolve. The integration of the above differential equations provides the evolution of the network states. With the existence of the energy function, the network will always converge to a stable state.

$$E = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} T_{ij} V_i V_j - \sum_{i=1}^{N} V_i \frac{u_i}{\tau} - \sum_{i=1}^{N} V_i I_i \tag{2}$$

Hopfield networks can only be programmed to memorize patterns using the Hebbian Rule. When the output function $g()$ is a sigmoid function, the network transforms the initial input vector iteratively and continuously into the output vector in the range $[0, 1]$. To program the network to memorize certain binary input vectors $(S(p), p = 1..P)$, the weight or conductance T_{ij} is determined by the following formula:

$$T_{ij} = \sum_{p=1}^{p} (2S_i(p) - 1)(2S_j(p) - 1) \tag{3}$$

When the output function $g()$ is a hyperbolic tangent function, the network transforms the initial input vector iteratively and continuously into output vector in the range $[-1, 1]$. To program the network to memorize certain binary input vectors $(S(p), p = 1..P)$, the weight or conductance T_{ij} is determined by the following formula:

$$T_{ij} = \sum_{p=1}^{p} S_i(p) S_j(p) \tag{4}$$

III. Proposed RHN (Restricted Hopfield Network)

To solve the problem of memory capacity and trainability issues of the Hopfield network, an analog restricted Hopfield network (RHN) is presented. The architecture, as shown in Fig. 2, consists of two layers of nodes, L visible and M hidden nodes, connected by directional weighted paths. The network is a connected bipartite graph and has no intralayer connection.

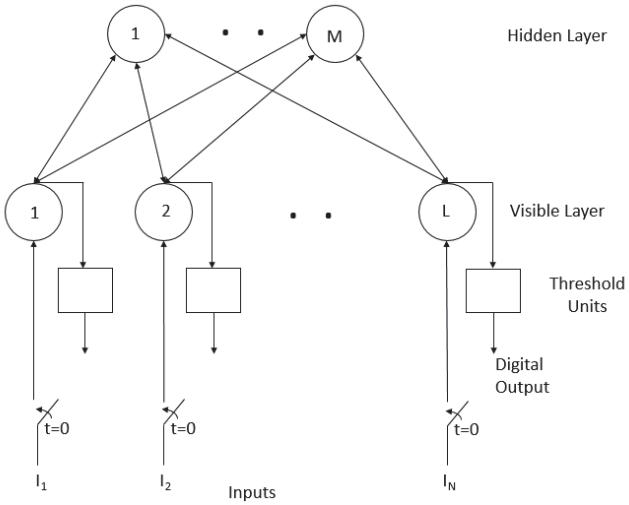

Fig. 2 Proposed Restricted Hopfield Network with hidden and visible nodes

The dynamics of the network can be described by the following differential equations:

In the forward path:

$$\begin{aligned} \frac{du_i^H}{dt} &= \sum_{j=1}^{L} w_{ij}^H V_j^V + \theta_i^H \\ V_i^H &= g(u_i^H) \end{aligned} \tag{5}$$

where u_i^H is the sum of all inputs to the hidden nodes, w_{ij}^H is the weight connecting the output of visible node j to the input of hidden node i, V_j^V is the output of visible node j, θ_i^H is the threshold of hidden node i, V_i^H is the output of hidden node i, $g()$ is the output function of hidden node I, and I_j is the initial input to the visible node j.

In the backward path:

$$\begin{aligned} \frac{du_i^V}{dt} &= \sum_{j=1}^{M} w_{ij}^V V_j^H + \theta_i^V \\ V_i^V &= g(u_i^v) \end{aligned} \tag{6}$$

where u_i^V is the sum of all inputs to the output nodes, w_{ij}^V is the weight connecting the output of hidden node j to the input of output node i, V_j^H is the output of hidden node j, θ_i^V is the threshold of output node i, V_i^V is the output of output node i, and $g()$ is the output function of output node i.

Initial conditions of the network are:

$$\begin{aligned} V_j^V(0) &= I_j \\ V_i^H(0) &= 0 \end{aligned} \tag{7}$$

The input to the network can be either digital or analog taking value between 0 and 1. Using a sigmoid function as output function $g()$ for all the nodes, the output of all the nodes takes the value between 0 and 1. Note that the proposed network will also work with hyperbolic tangent output function. The proposed RHN always generates analog outputs between 0 and 1 using sigmoid output function or between -1 and 1 using the hyperbolic tangent output function.

Threshold units can be deployed to obtain digital outputs, as shown in Fig. 2.

Using the Lyapunov Direct Method, it can be shown that the following equation is the energy or a Lyapunov function of the proposed network in Fig. 2 if and only if all weights are symmetric. Due to lack of space, the derivation will not be shown.

$$E = -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{L} w_{ij}^{H} V_i^H V_j^V - \frac{1}{2}\sum_{j=1}^{L}\sum_{i=1}^{M} w_{ji}^{V} V_j^H V_i^V - \sum_{i=1}^{M} V_i^H \theta_i^H - \sum_{i=1}^{L} V_i^V \theta_i^V \quad (8)$$

The proposed RHN is a dynamic system. Therefore, it has attractors toward which a system tends to evolve, for a wide variety of initial conditions of the system. There exists a basin of attraction for each attractor such that any initial condition in that region will eventually be iterated into the corresponding attractor. When an input vector is presented to the input nodes and the input vector is in a certain basin of attraction, the proposed network sends signals back and forth between the hidden and visible layers until all its nodes reach an equilibrium state or the corresponding attractor based on minimizing the energy function above, generating the desired output vector.

IV. Training of Proposed RHN

Conventional Hebbian rule as mentioned earlier ((3) and (4)) is not suitable for training the proposed network because the outputs of the hidden nodes are not known and the equations cannot handle analog quantities.

In many applications, the ideal or desired outcome for the network is known. Using this information, we can evaluate how well the network performs. The evaluation gives us a clue in terms of how to optimize the weights of the network. Therefore, the backward-error-propagation (BEP) through time method can be used to train the proposed RHN by propagating the error quantity through time from a stable state to an initial state. However, it is difficult to apply the method directly, because it is complicated to compute the gradient corresponding to all of the weights in every time step.

The simultaneous perturbation stochastic approximation (SPSA) algorithm uses a gradient approximation that requires only 2N objective function measurements over all N iterations regardless of the dimension of the optimization problem [9], [10]. Therefore, the SPSA algorithm, as shown in the following formulae, is suited for a high-dimensional optimization problem of minimizing an objective function dependent on multiple adjustable weights.

$$\Delta w_{ij}(k) = \frac{J(W(k) + c(k)\Delta(k)) - J(W(k) - c(k)\Delta(k))}{2c(k)\Delta_{ij}(k)} \quad (9)$$

$$w_{ij}(k+1) = w_{ij}(k) - a(k)\Delta w_{ij}(k) \quad (10)$$

At each iteration, a simultaneous perturbation delta vector with mutually independent zero-mean random variables is generated; each element $\Delta_{ij}(k)$ in $\Delta(k)$ matrix is generated with a probability of 0.5 of being either +1 or -1. Two weight matrices $W+$ and $W-$ are calculated by adding and subtracting the $\Delta(k)$ matrix scaled by gain sequence $c(k)$ to/from the current weight matrix $W(k)$ to compute their respective contributions $J(W+)$ and $J(W-)$ to the objective function. Dependent on the outcome of the evaluation and scaled by gain sequences $a(k)$ and $c(k)$, the current weight matrix W is updated accordingly. The gain sequences $a(k)$ and $c(k)$, decrease as the number of iterations k increases, will converge to 0 as k approaches ∞.

The objective function J used for the optimization of the proposed RHN is:

$$J = \sum_{i=1}^{P}\sum_{j=1}^{B} (\widehat{D}_{ij} - V_{ij}^{o}(k))^2 \quad (11)$$

where $\widehat{D}_{ij}$ is the j^{th} element of the desired output vector i, $V_{ij}^{o}(k)$ is the output value of the j^{th} output node when training pattern i is presented, B is the number of output nodes, and P is the number of training patterns.

V. Simulation Results

A. Comparison of Hopfield Network and RHN

Let us consider storing 3 vectors ([1 1 0 0], [0 1 1 0], [0 1 0 1]) in a Hopfield network. Hopfield network can be programmed to memorize these three vector patterns using Hebbian Rule and the weight matrix is:

$$\begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

The weight matrix will generate the correct output vector when the corresponding input vector with noise is provided. Let us consider storing the fourth vector [1 1 1 1] in the network. The weight matrix to store vector [1 1 1 1] is:

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Adding this weight matrix to the previous weight matrix of the Hopfield network programmed to store 3 vectors ([1 1 0 0], [0 1 1 0], [0 1 0 1]) yields a weight matrix with zero elements, erasing all the previous programming, as shown in the following. Therefore, the Hopfield network cannot be programmed to remember all four vectors ([1 1 0 0], [0 1 1 0], [0 1 0 1], [1 1 1 1]) using Hebbian rule.

$$\begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

However, by introducing 5 hidden nodes in the proposed RHN, all four-vectors can be easily stored and recalled correctly. This shows that it is possible to increase the memory capacity of the Hopfield network by using more hidden nodes. In fact, by increasing the number of hidden nodes to 20, the proposed network can remember all 16 binary vectors.

B. Associative Memory

An RHN with 35 visible nodes and 10 hidden nodes is created. The network is then trained to memorize the A, U, T, S characters, each with 5×7 binary pixels. To test whether the network can re-create the images of these characters in the presence of noise, the input images are distorted by changing some of the pixels measured in Hamming distance.

Shown in Figs. 3 and 4 are distorted images of A, U, T, S on the left of the figures and the re-created images of these characters on the right. The figures show that the network is able to perform perfect re-creation of these images even when distorted character images with hamming distance of 5 is presented.

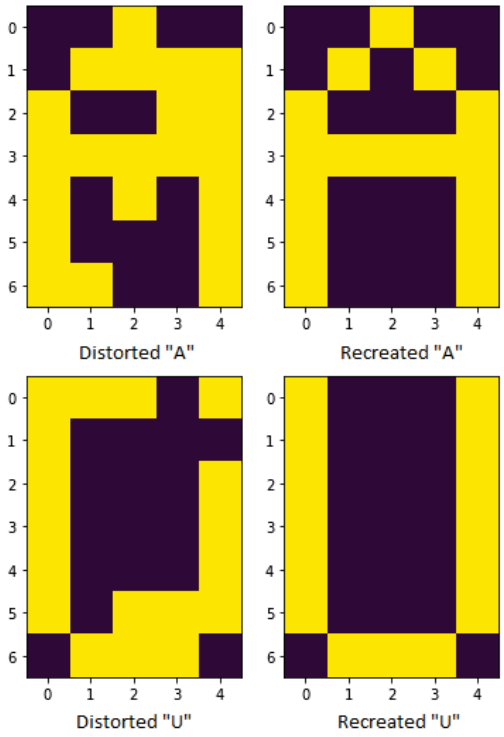

Fig. 3 Distorted images of A and U on the left and the recreated images on the right

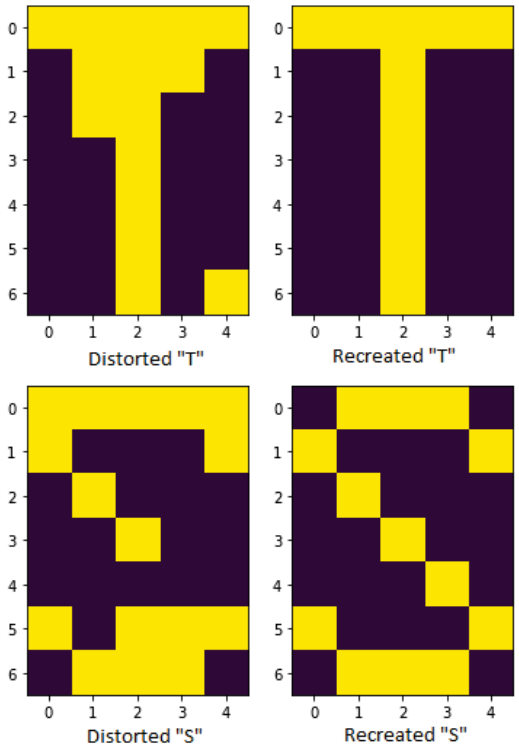

Fig. 4 Distorted images of T and S on the left and the recreated images on the right

A Hopfield Network with 35 nodes was also created and programmed to store the same images of characters A, U, T, S. To compare whether both networks can re-create the images of these characters in the presence of noise, the input images are distorted by randomly changing some of the binary pixels measured in Hamming distance. Using 1,000 images per character, Fig. 5 shows the percentage error of both networks for not re-creating the images perfectly. The RHN can perform perfect re-creation of the training images with an average error rate of 7% when the Hamming distance is 8. A classical Hopfield network can only achieve 16% error rate with the same Hamming distance.

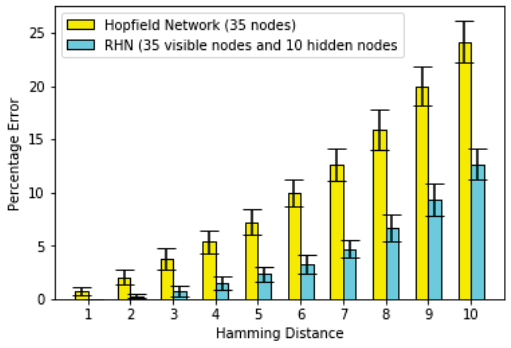

Fig. 5 Performance comparison of Hopfield Network and RHN both functioning as an associative memory to store images of A, U, T, S characters

A RHN with 35 visible nodes and 50 hidden nodes was

trained to store 10 7×5 binary pixels of numerical digits. Using a total of 10,000 distorted images, simulation results show that the RHN can re-create these images with an average error rate of 14% when the Hamming distance is 5, as shown in Fig. 6.

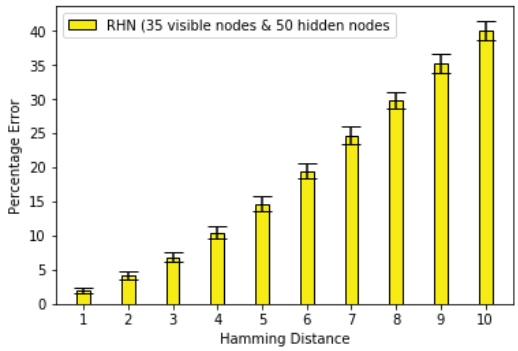

Fig. 6 Performance comparison of Restricted Hopfield Network as associative memory for 0 9 numbers

A RHN with 35 visible nodes and 50 hidden nodes was trained to store 26 7×5 binary pixels of A to Z alphabets. Using a total of 26,000 distorted images, simulation results show that the RHN can re-create these images with an average error rate of 23% when the Hamming distance is 5, as shown in Fig. 7.

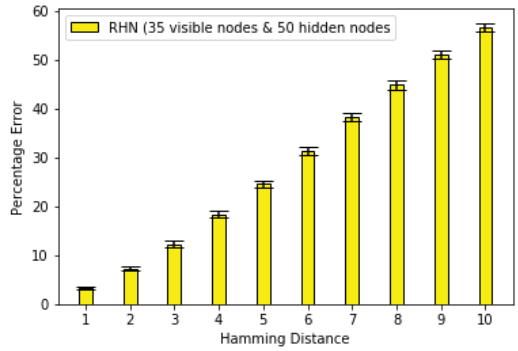

Fig. 7 Performance comparison of Restricted Hopfield Network as associative memory for A Z characters

VI. Conclusions

An analog restricted Hopfield Network is presented in this paper. A Lyapunov function was provided to show that the proposed network will converge to stable states. By introducing hidden nodes, the proposed network can be trained to store patterns and has increased memory capacity. Acting as an associative memory, simulation results show that the RHN can be trained to store many images. The proposed network performs better memory recall than a classical Hopfield network when the input is noisy.

References

[1] Hopfield, J.J., Neural networks and physical systems with emergent collective computational abilities, Proc. of the National Academy of Sciences, USA, Vol. 79, 1982, 2554-2558.

[2] Hopfield, J.J. and Tank D.W., Neural computation of decisions in optimization problems, Biological Cybernetics, Vol. 52, 1985, 141-152.

[3] Hopfield, J.J. and Tank D.W., Computing with neural circuits: model, Science, New Series, Vol. 233, No. 4764 (Aug. 8, 1986), 625-633.

[4] McEliece, R.J., Posner, E.C., Rodemich, E.R., and Venkatesh, S.S., The capacity of Hopfield associative memory, IEEE Transaction on Information Theory, Vol. IT-33, No. 4, July 1987, 461-482.

[5] Storkey, A.J. and Valabregue, R., The basins of attraction of a new Hopfield learning rule, Neural Networks, Vol. 12, 1999, 869-876.

[6] Gardner, E., Maximum storage capacity in neural networks, Europhys. Lett., Vol. 4, 1987, 481-485.

[7] Gardner, E., The space of interactions in neural network models, J . Phys. A : Math Gen., Vol. 21, 1988, 257-270.

[8] Gardner, E. and Derrida, B., Optimal storage properties of neural network models, J . Phys. A : Math Gen., Vol. 21, 1988, 271-84.

[9] Salakhutdinov, R. and Hinton, G., Restricted Boltzmann Machines, Proc. of the 12th International Confe-rence on Artificial Intelligence and Statistics (AISTATS), 2009, Clearwater Beach, Florida, USA.

[10] Sutskever, I., Hinton, G., and Taylor, G., The recurrent temporal restricted Boltzmann machine, Proc. of Neural Information Processing Systems, 2008.

[11] Salakhutdinov, R. R., Mnih, A., and Hinton, G. E., Restricted Boltzmann machines for collaborative filtering, Proceedings of the International Conference on Machine Learning, Vol. 24, 791-798., ACM.

[12] Spall, J.C., Multivariate stochastic spproximation using simultaneous perturbation gradient approximation, IEEE Transaction on Automatic Control, Vol. 37, No. 3, March 1992, 333-341.

[13] Spall, J.C., An overview of the simultaneous perturbation method for efficient optimization, Johns Hopkins Apl Technical Digest, Vol. 19, No. 4, 1998, 482-492.