Direction of Arrival Estimation Using Artificial Neural Networks

Sanjay Jha and Tariq Durrani, *Fellow, IEEE*

Abstract—**The direction of arrival (DoA) problem is concerned with determining the directions of the sources irradiating an array of sensors in the presence of additive noise and other distortions. The maximum likelihood estimator is the optimal estimator of the direction of sources, but it requires the minimization of a complex, multimodal, multidimensional cost function. If an initial estimate of the direction of the sources is available then Gauss–Newton method can be used to achieve high resolution estimates. However, in the absence of initial estimates, computationally complex heuristic methods such as simulated annealing and genetic algorithms have been used to minimize the maximum likelihood cost function. A neural optimization procedure is presented that does not require an initial estimate of the direction of the sources, and which offers the potential of real-time solutions to the DoA problem by utilizing the fast relaxation properties of the Hopfield network. The minimum mean square error cost function of the DoA problem, which is also the maximum likelihood cost function for Gaussian noise and linear equispaced array data, is mapped onto the Liapunov energy function associated with the Hopfield network; the network is then used to minimize this cost function. However the Hopfield network implements a gradient descent procedure, and in common with all such procedures, it is liable to find the locally optimum solution rather than the desired globally optimum solution. Therefore, a novel modification based on iterated descent procedure is introduced into the Hopfield model dynamic equation to increase the probability of convergence to the global optimum. In the absence of technology for practical analog neural circuits, the proposed algorithms are implemented on an array of closely coupled transputers that perform the random asynchronous neural updates in parallel. The mapping onto the parallel architecture is achieved using a technique called "chaotic relaxation." Simulation results are presented to characterize the performance of the neural approach in terms of the variance of the estimates of source-directions, and in terms of the time required for the computation of the estimates.**

I. Introduction

TRADITIONALLY, the direction of arrival (DoA) problem has been solved using time series modeling methods such as AR/MA, maximum entropy (ME) or linear prediction (LP) [1]. A major limitation of these methods is their inability to resolve closely located sources. High resolution methods such as Pisarenko and MUSIC [2] offer asymptotically unbiased estimates of the direction of the irradiating sources, but they are computationally more expensive. These subspace-based methods, and their enhancements are highly sensitive to the structure of the covariance matrix (because of the high sensitivity of the eigenvectors to the perturbations in the matrix), and require additional computation to determine the number of sources radiating an array. This order-determination problem is a nontrivial task in noisy conditions [3]. The ESPRIT [4] algorithm offers advantages over the MUSIC algorithm by avoiding the orthogonality search, and by reducing the effect of sensor variability on the performance of the algorithm. However, this is achieved at the expense of increased number of sensors in the arrays.

There is a resurgence of interest in the use of neural networks to solve some of the problems arising in signal processing applications [5]. Hopfield and Tank [6] have applied a single layer network of fully interconnected neurons to solve combinatorial optimization problems such as the travelling salesperson problem (TSP). Following [12]–[15], the DoA cost function has been mapped onto the quadratic energy function of the Hopfield model network in a minimum-mean square sense. For Gaussian noise and linear equispaced array, this estimator can be shown to be equivalent to the maximum likelihood estimator.

The maximum likelihood estimator (MLE) for the DoA problem is the optimal estimator, but it requires the minimization of a highly nonlinear, multimodal, and multidimensional cost function. When a coarse initial estimate is available, the ML cost function can be minimized using the Gauss–Newton algorithm [16]. However, in the absence of a reliable estimate, computationally intensive heuristic algorithms such as simulated annealing [17] have been used to minimize the ML cost function.

Simulated annealing (SA) is a function minimization algorithm [18] that allows for the probability of increase in energy, thus enabling the optimization procedure to escape from local minima. The probability of acceptance of increase in energy is governed by the cooling schedule, which has to be very slow to guarantee convergence to the global minimum. A fast version of SA [19] has been proposed, but even this requires an excessively large computation effort [17], and precludes real-time solution to the DoA problem.

The overwhelming benefit of the neural method proposed here over the linear-algebra based methods is the fast relaxation of the network into the solution because of its massive parallelism and global connectivity. However, the Hopfield model implements a gradient descent algorithm, and in common with such algorithms, it is liable to find the local minimum of the energy function closest to the initial state rather than the desired global minimum. There have been

Manuscript received August 15, 1990; revised March 17, 1991. This work was supported by the SDIO under Contract no. N00014-88-J1198 and managed by ONR/IST.

The authors are with the Signal Processing Division, Department of Electronic and Electrical Engineering, The University of Strathclyde, The Royal College, 204 George Street, Glasgow, G1 1XW, Scotland, U.K.

IEEE Log Number 9102010.

attempts to combine the hill-climbing heuristics of the SA with the fast relaxation of the Hopfield model network. In [6], Hopfield suggested the use of analog neurons, in [20] Ackley *et al.* proposed the use of Boltzmann machine, and in [21] Levy and Adams proposed the stochastic network to increase the probability of finding the global minimum.

In this paper, we present a new procedure called *iterated descent* for increasing the probability of finding the global minima of the Hopfield network energy function. Simulation results are also presented to show the performance of the modified Hopfield model in solving the DoA problem in terms of variance of the estimates and the resolution of closely located sources.

The problem of hardware implementation of artificial neural networks has also been considered in this paper. There have been three approaches to neural implementation: the first is the use of a dedicated analog VLSI hardware [7], the second is the mapping of neural algorithm onto a general or dedicated parallel array hardware [8], and the third approach has been to use general purpose parallel machines [9] to increase the speed of simulations. While analog neural devices have been demonstrated, they remain at an experimental stage. In the absence of practical analog networks, dedicated parallel array architectures have been presented [8], but they do not implement the *random asynchronous* updating procedure required for the Hopfield model. In this paper, the random asynchronous neural updating procedure of the Hopfield model has been implemented onto an array of transputers. Results are presented for the execution times of the proposed DoA estimation algorithms.

The paper is organized as follows: Section II presents the neural algorithm and the optimum-finding modification to the Hopfield model net. The mapping of the DoA problem onto the neural network energy function is presented in Section III, and simulation results are presented in Section IV that characterize the performance of the neural bearing estimator. Section V looks at the computational requirements of the neural update procedure, and presents a dedicated parallel array architecture to perform the random-asynchronous neural update.

II. The Hopfield Networks Algorithm

A. Neural Network Energy Function

The Hopfield model neural network is a single layer of fully interconnected neurons (Fig. 1) that update their outputs upon sampling the outputs of other neurons in the network, via the *synaptic link*. The synaptic link between the ith and the jth neurons, in a network of P neurons, form a symmetric matrix T, such that [11]:

$$t_{ij} = t_{ji}; \qquad t_{ii} = 0 \quad \forall\, t_{ij} \in \Re \quad \text{for } 1 \le i, j \le P. \tag{1}$$

The network changes state using the following dynamic equation:

$$C_i \frac{du_i}{dt} = \sum_j^P t_{ij}\nu_j + I_i \tag{2}$$

where C_i is the input capacitance of the ith neuron, I_i is the external stimulus, and u_i is the internal state of the neuron.

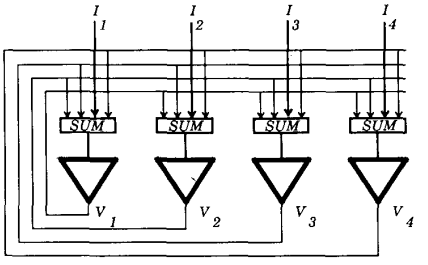


Fig. 1. The Hopfield model neural network comprises single layer of fully interconnected neurons that update their output upon sampling the output of other neurons in the network through the synaptic links. The synapses can be positive as well as negative. No self-feedback is allowed.

The output state ν_i of the neuron is given by the following nonlinear transformation (Fig. 2):

$$\nu_i = g_i(u_i, \lambda) = \frac{1}{2}\left(1 + \tanh\frac{u_i}{2\lambda}\right); \quad \forall\, \nu_i \in \{0, 1\}. \tag{3}$$

where $g_i(u_i, \lambda)$ is the sigmoid transfer function of the ith neuron, and $1/\lambda$ is the gain of the neuron. The network dynamic equation ((2)) defines a complex system, but it is possible to find an energy function satisfying the Liapunov's stability criterion [11]:

$$E = -\frac{1}{2}\sum_i^P \sum_j^P t_{ij}\nu_i\nu_j - \sum_i^P I_i\nu_i. \tag{4}$$

The rate of change of energy E with respect to time is given by

$$\frac{dE}{dt} = \sum_i^P \frac{dE}{d\nu_i}\frac{d\nu_i}{dt} = \sum_i^P \frac{dE}{d\nu_i}\frac{d\nu_i}{du_i}\frac{du_i}{dt}. \tag{5}$$

But from (4):

$$\frac{dE}{d\nu_i} = -\left(\sum_j^P t_{ij}\nu_j + I_i\right) = -C_i\frac{du_i}{dt}. \tag{6}$$

Therefore

$$\frac{dE}{dt} = -\sum_i^P C_i\left(\frac{du_i}{dt}\right)^2\frac{d\nu_i}{du_i}. \tag{7}$$

If the gradient of the neuronal transfer function $(d\nu_i)/(du_i)$ is always positive, that is to say, if $g_i(u_i, \lambda)$ is a monotonically increasing function (as is the case for the transfer function chosen in (3)), then the network always evolves to reduce the network energy. Under the conditions presented therefore, the Hopfield model implements a gradient descent algorithm, and given the complex multimodal cost function of the DoA problem, the network is liable to find a local rather than the global minimum. To overcome this problem and increase the probability of finding the global minimum, some modifications have been proposed. These are considered in the next subsections.

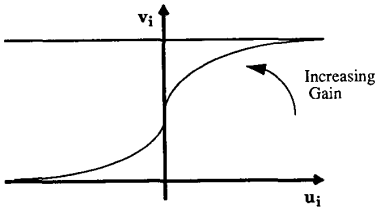


Fig. 2. The sigmoid neuronal transfer function. As the gain increases, the transfer function approaches the Signum function.

B. Gain Annealing

The decision space of a *P-neuron* network is represented by a *P-dimensional* hypercube $D = [0,1]^P$. Each corner of this hypercube represents a possible digital output state of the network; one of these corners represents the solution state, and one or more of the other corners represents the local-minima of the energy function. The network starts from some initial state within D and evolves towards one of the corners that corresponds to a minimum. For a digital neuronal transfer function, the network can only change state along the edges of the hypercube. However for an analog network with a sigmoid transfer function, the network can traverse within the interior of the decision-space hypercube to reach the desired corner. A low-grain analog network therefore embeds a digital optimization problem in an analog domain and offers a better solution because of the greater degree of freedom in the optimization process. The energy function of (3) however applies only to a digital network, and the use of an analog transfer function represents a perturbation to this function, of the form

$$\overline{E} = -\frac{1}{2}\sum_{i}^{P}\sum_{j}^{P} t_{ij}\nu_i\nu_j - \sum_{i}^{P} I_i\nu_i + \sum_{i}^{P}\xi(\nu_i,\lambda) \quad (8)$$

where $\overline{E}$ is the perturbed form of the energy function E, and $\xi(\nu_i,\lambda)$ is the perturbation in the energy function due to the analog transfer function of the ith neuron, and is given by [22]:

$$\tau_i \frac{d}{d\nu_i}\xi(\nu_i,\lambda) = g_i^{-1}(\mu_i,\lambda) = u_i \quad (9)$$

where τ_i is the time-constant of the loading on the ith neuron, and it incorporates the capacitive loading C_i of (2). The gradient $\left((d\overline{E})/(d\nu_i)\right)$ is now given by

$$\frac{d\overline{E}}{d\nu_i} = \frac{u_i}{\tau_i} - \sum_{j}^{P} t_{ij}\nu_j - I_i \quad (10)$$

And using Pontryagin's adjoint equations:

$$\frac{du_i}{dt} = -\frac{u_i}{\tau_i} + \sum_{j}^{P} t_{ij}\nu_j + I_i. \quad (11)$$

Therefore, the new dynamic equation of the network is defined by (11). However, to ensure that the stable point of the network still lie in the corners of the decision space, the following condition has to be satisfied:

$$\lim_{\lambda\to 0}\sum_{i}^{P}\xi(\nu_i,\lambda) = 0. \quad (12)$$

To satisfy this condition, while retaining the benefits of an analog network, the following *gain annealing* scheme may be adopted:

$$\lambda_t = \lambda_0 \times \alpha^t \quad (13)$$

where α is the annealing factor between 0 and 1. Note that

$$\lim_{\lambda\to 0}\frac{1}{2}\left(1+\tanh\frac{u_i}{2\lambda}\right) = \operatorname{sgn}(u_i) \quad (14)$$

where sgn(.) is the Signum function.

C. The Boltzmann Machine

Gain annealing as proposed in the previous section, increases the probability of finding *better* minima [6], but it does not allow for an increase in the network energy. The Boltzmann machine (BM) is a probabilistic network that allows for an increase in the energy of the network. The structure of the BM net is the same as the digital Hopfield network, but it uses the following dynamic equation to determine the new state of the neuron, based on the transition probability given by

$$\rho_i(0) = \frac{1}{1+\exp\left(\frac{-\Delta E_i}{KT}\right)} \quad \text{and} \quad \rho_i(1) = 1-\rho_i(0) \quad (15)$$

where $\rho_i(0)$ and $\rho_i(1)$ are the probability that the new state of the neuron will be zero or one respectively, and $-\Delta E_i$ is the change in the energy of the network due to the change in the state of the ith neuron to zero; K is the Boltzmann constant and T is the "temperature" of the optimization procedure (by analogy with SA). If the control parameter T is reduced sufficiently slowly, the probability density of the states of the network is guaranteed to converge to the global minimum state of the energy function [20]. The cooling schedule that guarantees convergence is given by

$$T(t) = \frac{c}{\log t} \quad (16)$$

where $T(t)$ is the temperature at the tth iteration and c is a constant. For the BM network:

$$\Delta E_i = \left[\sum_{j}^{P} t_{ij}\nu_j + I_i\right]\Delta\nu_i \quad (17)$$

and, for the transition $\nu_i = 1$ to $\nu_i = 0, \Delta\nu_i = -1$, therefore:

$$\Delta E_i = -\sum_{j}^{P} t_{ij}\nu_j - I_i = -u_i. \quad (18)$$

Hence, the transition probability can be expressed as

$$\rho_i(0) = \frac{1}{1+\exp\left(\frac{u_i}{KT}\right)} \quad \text{and} \quad \rho_i(1) = 1-\rho_i(0). \quad (19)$$

At high temperatures, the decision to update to 0 or 1 is largely independent of the internal state u_i, but at lower temperatures, the network behaves like a digital Hopfield model network.

D. The Stochastic Network

In the BM, the optimization is carried out in the digital domain, and therefore it does not benefit from the greater degree of freedom in the optimization process offered by the gain annealing method. The stochastic network (SN) [21] combines gain annealing with a hill-climbing procedure based on the diffusion process. Under the SN optimization process, the network dynamic equation is a combination of the drift term proportional to the $-\Delta E(\nu_i)$, and a white noise process whose intensity is proportional to the control parameter T. It ensures that the optimization process possesses the key property of the SA procedure: if the temperature is reduced sufficiently slowly, the global minimum is guaranteed [21]. The dynamic equation of the SN is

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} + \sum_{j}^{P} t_{ij}\nu_j + I_i + \sqrt{2T}\,\eta_i(t) \tag{20}$$

where $\eta_i(t)$ is the noise process such that

$$\boldsymbol{E}[\eta_i(t)(\eta_i(s))] = q(u_i)\delta(t-s) \tag{21}$$

$\delta(.)$ is the Dirac-Delta function, and $q(u_i)$ is variance and given by

$$q(u_i) = \frac{d}{d\nu_i}\, g_i^{-1}(\nu_i, \lambda). \tag{22}$$

For the sigmoid function considered in (3):

$$[q(u_i)]^{\frac{1}{2}} = 2\sqrt{\lambda}\,\cosh\frac{u_i}{2\lambda}. \tag{23}$$

Therefore, for the given sigmoid function, the SN dynamic equation is given by

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} + \sum_{j}^{P} t_{ij}\nu_j + I_i + 2\sqrt{2T\lambda}\,\cosh\frac{u_i}{2\lambda}\eta_i(t) \tag{24}$$

where $\eta_i(t)$is now a white-noise process with unit variance.

The cooling schedule for T, which guarantees convergence is the same as that for BM (16). The gain λ has to be decreased more slowly than T, and it has to be kept bounded away from zero. As T decreases, the noise intensity in (24) decreases, and for low values of T the SN updating equation approaches the gain annealing updating (11).

E. Iterated Descent

Iterated descent (ID) is a novel algorithm that combines the benefits of analog Hopfield network with the ability of the BM and SN to dislodge the network from the local minimum. If we consider the optimization pair ($\boldsymbol{f}$, $\boldsymbol{S}$), where $\boldsymbol{f}$ is the energy or cost function, and $\boldsymbol{S}$ is the finite space over which $\boldsymbol{f}$ is valid, then

$$\boldsymbol{f} : \boldsymbol{S} \rightarrow \Re \tag{25}$$

defines the relationship between them. The minimization task is to find the state $s^{\text{opt}}\{s^{\text{opt}} \in \boldsymbol{S}\}$ such that the energy value of that state satisfies the following condition:

$$\boldsymbol{f}(s^{\text{opt}}) \leq \boldsymbol{f}(s) \quad \text{for } \forall\ s \in \boldsymbol{S} \tag{26}$$

The Hopfield model neural network on the other hand only finds the local minima of the cost function. That is to say, for a subset of the state-space $\boldsymbol{L_i}, \{\boldsymbol{L_i}, \subset \boldsymbol{S}\}$ such that

$$\boldsymbol{S} = \boldsymbol{L}_1 \cup \boldsymbol{L}_2 \cup \ldots \boldsymbol{L}_n \tag{27}$$

the Hopfield network finds the local minima $s_i^{\text{opt}}, \{s_i^{\text{opt}} \in \boldsymbol{L}_i\}$ such that

$$\boldsymbol{f}(s_i^{\text{opt}}) \leq \boldsymbol{f}(s_i), \quad \text{for } \forall\ s_i \in \boldsymbol{L}_i \tag{28}$$

The subset $\boldsymbol{L}_i$ is said to be the *basin of attraction* of s_i^{opt}. The ID algorithm utilizes the ability of the Hopfield model to find the attractor of the basin it has been placed in. Therefore, the optimization problem stated in (26) can be reduced to placing the network in the basin $\boldsymbol{L}_{\text{opt}}$ such that $\{s^{\text{opt}} \in \boldsymbol{L}_{\text{opt}}\}$. So, whereas BM and SN try to find the optimum state, the ID algorithm merely finds the basin of attraction that contains the optimum state. Since, in any cost function, the number of basins of attraction are fewer than the number of states in the statespace, the ID algorithm solves a simpler problem. The ID algorithm proceeds as follows.

1) Find a local minimum using the gain annealing dynamic equation of Section II-A
2) Displace the network in the state-space and give it an energy-kick. If the network has found the local minima $s_i^{\text{opt}}, \{s_i^{\text{opt}} \in \boldsymbol{L}_i\}$, then displace it into another subspace $\boldsymbol{L}_j$. Let the network then find the local optimum of $s_j^{\text{opt}}, \{s_j^{\text{opt}} \in \boldsymbol{L}_j\}$.
3) Reduce the state-space displacement and the energy-kick to the network for successive iterations using a cooling schedule that guarantees convergence to the global minima *in probability*. In the limit, the ID procedure is the same as the stochastic network procedure, and therefore its convergence is similarly guaranteed [21]. In practice however, the ID procedure converges more quickly, as shown by the results in Section IV.

The state displacement and the energy-kick to the network is provided by adding noise to the dynamic update equation after the network has settled, using the SN dynamic equation (24). A proposed cooling schedule is [19]:

$$T(t) = \frac{c}{t}. \tag{29}$$

III. The Bearing Estimation Problem

A. The Data Model

The data vector $\boldsymbol{y}_t$ incident upon a linear equispaced array of n sensors described by

$$\boldsymbol{y}_t = \sum_{i}^{m} \alpha_i e^{j\omega_i t}\boldsymbol{d}_i + \boldsymbol{w}_i(t) \tag{30}$$

where m is the number of sources irradiating the array, α_i, and ω_i are the amplitude and frequency of the ith source

respectively, and $\boldsymbol{d}_i$ is the m element direction vector corresponding to each source, of the form

$$\boldsymbol{d}_i = \left(1, e^{-j\tau_i}, e^{-j2\tau_i}, \cdots, e^{-j(n-1)\tau_i}\right)^t \quad \text{for } 1 \le i \le m. \tag{31}$$

The sources are assumed to be in the far-field, in the plane of the sensor array, and stationary (i.e. there is no relative movement between the sources and the sensors). The position of the sensors have been assumed to be known accurately. The noise processes, $\boldsymbol{w}_i(t)$, are assumed to be uncorrelated complex Gaussian random variables with zero mean. The τ_i in (31) is the differential propagation delay between sensors for the ith source. The bearing of the ith source, θ_i, is related to the τ_i by

$$\tau_i = \frac{2\pi d}{\lambda_i} \sin\theta_i \tag{32}$$

where d is the intersensor separation of the equispaced array, and λ_i is the wavelength of the radiation from the ith source.

B. Mapping the DoA Problem onto the Neural Network

The DoA problem is mapped onto the Hopfield model neural network by discretizing the bearing, amplitude and the frequency spaces of interest to the required granularity, and using each neuron in the network to represent a unique combination of the three properties [12]. If each of the three spaces have been discretized to p levels, than, a network of p^3 neurons is required to represent each unique hypothesis vector, $\boldsymbol{s}_{q,r,s}$ of the form:

$$\boldsymbol{s}_{q,r,s} = \alpha_q e^{j\omega_r}\left(1, e^{-j\tau_s}, e^{-j2\tau_s}, \cdots, e^{-j(n-1)\tau_s}\right)^t \quad \text{for } 1 \le q, r, s \le p. \tag{33}$$

For the ease of notation:

$$\boldsymbol{s}_{q,r,s} \Rightarrow \boldsymbol{s}_i \quad \text{for } 1 \le q, r, s \le p; \quad \text{and} \quad 1 \le i \le p^3. \tag{34}$$

These hypothesis are used to minimize the error $\mathbf{Q}$ in the *least square* sense to best explain the data incident at the sensors, $\boldsymbol{y}$:

$$\boldsymbol{Q} = \|\boldsymbol{y} - \boldsymbol{S\nu}\|_2^2 \tag{35}$$

where $\boldsymbol{S}$ is a $p^3 \times n$ matrix whose column vectors are $\boldsymbol{s}_i\{i = 1 \ldots p^3\}$, and $\boldsymbol{v}$ is the output state vector of the network comprising $v_i\{i = 1 \ldots p^3\}$. The element of $\boldsymbol{v}$ take values between 0 and 1, and once the network has settled, they take values close to zero and one. The hypothesis $\boldsymbol{s}_i$ corresponding to $v_i = 1$ indicates the composition of the input data. Expanding (35) leads to

$$\boldsymbol{Q} = \boldsymbol{y}^h\boldsymbol{y} + \boldsymbol{\nu}^t\boldsymbol{S}^h\boldsymbol{S\nu} - \boldsymbol{y}^h\boldsymbol{S\nu} - \boldsymbol{\nu}^t\boldsymbol{S}^h\boldsymbol{y} \tag{36}$$

where the h operator represents a conjugate transpose of both matrices and vectors, and the t operator represents vector transpose. In (35), the first right-hand side term is independent of $\boldsymbol{v}$, and

$$\boldsymbol{\nu}^t\boldsymbol{S}^h\boldsymbol{y} + \boldsymbol{y}^h\boldsymbol{S}\boldsymbol{S\nu} = 2\ \mathrm{Re}\left[\boldsymbol{y}^h\boldsymbol{S\nu}\right]. \tag{37}$$

Therefore, (36) can be rewritten as

$$\boldsymbol{Q} = \boldsymbol{\nu}^t\boldsymbol{S}^h\boldsymbol{S\nu} - 2\ \mathrm{Re}\left[\boldsymbol{y}^h\boldsymbol{S\nu}\right]. \tag{38}$$

Following Hopfield and Tank [6], a penalty term is introduced to this cost function to force the neurons to digital values:

$$\boldsymbol{Q} = \boldsymbol{\nu}^t\boldsymbol{S}^h S\nu - 2\ \mathrm{Re}\left[\boldsymbol{y}^h S\boldsymbol{\nu}\right] - \sum_{i=1}^{p^3}\left(\boldsymbol{s}_i^h\boldsymbol{s}_i\right)\boldsymbol{\nu}_i(\boldsymbol{\nu}_i - 1). \tag{39}$$

Note that the last term contributes to the error only when v_i is not equal to either zero or one. In the above equation $\boldsymbol{S}^h\boldsymbol{S}$ is a Hemittian symmetric matrix, therefore

$$\boldsymbol{\nu}^t S^h S\boldsymbol{\nu} = \mathrm{Re}\left[\boldsymbol{\nu}^t S^h S\nu\right] = \mathrm{Re}\left[\sum_i^{p^3}\sum_j^{p^3}(\boldsymbol{s}_i\boldsymbol{s}_j)\boldsymbol{\nu}_i\boldsymbol{\nu}_j\right] \tag{40}$$

By substituting in (39), we have

$$\boldsymbol{Q} = \mathrm{Re}\left[\sum_{\substack{i\\ i\neq j}}^{p^3}\sum_j^{p^3}(\boldsymbol{s}_i\boldsymbol{s}_j)\nu_i\nu_j - \sum_i^{p^3}\left(2\boldsymbol{y}^{\mathbf{h}}\boldsymbol{s}_i + \boldsymbol{s}_i^h\boldsymbol{s}_i\right)\nu_i\right]. \tag{41}$$

Comparing (41) with the Liapunov energy function of the Hopfield model ((4)) leads to

$$t_{ij} = \begin{cases} -\mathrm{Re}\left[\boldsymbol{s}_i^h\boldsymbol{s}_j\right] & \text{for } i \neq j \\ 0 & \text{for } i = j \end{cases} \tag{42}$$

$$I_i = \mathrm{Re}\left\{\boldsymbol{y}^h\boldsymbol{s}_j - \frac{1}{2}\boldsymbol{s}_i^h\boldsymbol{s}_j\right\}. \tag{43}$$

In the development above, p^3 neurons have been used, and even for modest values of p, the number of neurons required is large.

There are practical circumstances where only the direction of arrival is of interest: that is to say, a narrow-band point-source of unit amplitude is assumed. For this scenario, only the phase space needs to be discretized to p levels, and the $\boldsymbol{Q}$ is redefined as

$$\boldsymbol{Q} = \|\boldsymbol{y} - [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_p]\boldsymbol{\nu}\|_2^2 \tag{44}$$

where $\boldsymbol{v}$ is a *p-element* vector, and $\boldsymbol{x}_i$ is defined as

$$\boldsymbol{x}_i = \boldsymbol{P}_{s_i}\boldsymbol{y} \tag{45}$$

In the above equation, $\boldsymbol{P}_{s_i}$ is the projection matrix that projects $\boldsymbol{y}$ onto the new hypothesis vector $\boldsymbol{s}_i$:

$$\boldsymbol{s}_i = \left(1, e^{-j\tau_i}, e^{-j2\tau_i}, \cdots, e^{-j(n-1)\tau_i}\right)^t \quad \text{for } 1 \le i \le p. \tag{46}$$

The projection matrix is defined as

$$\boldsymbol{P}_{s_i} = \boldsymbol{s}_i\left(\boldsymbol{s}_i^h\boldsymbol{s}_i\right)^{-1}\boldsymbol{s}_i^h. \tag{47}$$

By comparison with the earlier development, the new synaptic and external input values are found to be as follows [12]:

$$t_{ij} = \begin{cases} -\mathrm{Re}\left[\boldsymbol{y}^h\boldsymbol{P}_{s_i}^h\boldsymbol{P}_{s_j}\boldsymbol{y}\right] & \text{for } i \neq j \\ 0 & \text{for } i = j \end{cases} \tag{48}$$

$$I_i = \frac{1}{2}\ \mathrm{Re}\left[\boldsymbol{y}^h\boldsymbol{P}_{s_i}\boldsymbol{y}\right]. \tag{49}$$

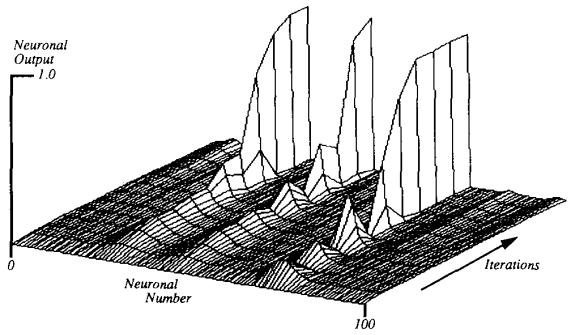


Fig. 3. The evolution of the network to the digital state: the gain of the neurons are increased for successive iterations. The network settled into a solution in 13 iterations.

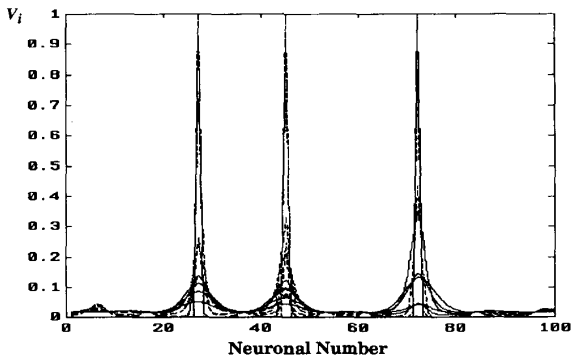


Fig. 4. Three sources at 12.0, 20.0 and 32.0 degrees were used to generate data at an SNR of 5 dB. A linear array of 10 equispaced sensors was used to collect 25 snapshots of data. The 26th, 48th, and 72nd neurons represent the hypothesis that the sources are located at these bearings.

Therefore the cost of eliminating amplitude and frequency from the minimization process has been to make the synapse dependent upon the input data: the synapse has to be calculated for each input snapshot. The above development is based upon a single-data vector, but it can be extended to a minimum-mean square formulation, by redefining the error function as [14]:

$$\boldsymbol{Q} = \frac{1}{L}\sum_{l}^{L} \|\boldsymbol{y}_l - [P_1\boldsymbol{y}_l, P_2\boldsymbol{y}_l, \cdots, P_p\boldsymbol{y}_l]\,\boldsymbol{\nu}\|_2^2. \quad (50)$$

By further analysis of this equation, the new synaptic values are found to be

$$t_{ij} = \begin{cases} -\mathrm{Re}\left[\dfrac{1}{L}\displaystyle\sum_{l}^{L} \boldsymbol{x}_{i,l}^h \boldsymbol{x}_{j,l}\right] & \text{for } 1 \le i,j \le p \\ 0 & \text{for } i \ne j \end{cases} \quad (51)$$

$$I_i = \frac{1}{2}\,\mathrm{Re}\left[\frac{1}{L}\sum_{l}^{L} \boldsymbol{y}_l^h \boldsymbol{x}_{i,l}\right] \quad \text{for } 1 \le i \le p \quad (52)$$

where for L data snapshots:

$$\boldsymbol{x}_{i,l} = \boldsymbol{P}_{s_i}\boldsymbol{y}_l \quad \text{for } 1 \le i \le p;\ 1 \le l \le L. \quad (53)$$

Equations (51) and (52) present the minimum-mean square mapping of the bearing estimation problem onto the neural energy function, and under certain assumptions (Gaussian uncorrelated noise and linear equispaced sensor array), this is equivalent to the maximum likelihood mapping of the problem. Using this mapping, the DoA problem may be solved by the neural algorithms given in Section II.

IV. Simulation Results

A. Data Generation

Narrow-band synthetic data was generated at the normalized frequency of 0.2, for a linear array of 10 equispaced sensor, with an intersensor separation of $\lambda/2$. The deterministic data vector was generated using the equations presented in Section III A. The covariance matrix $\boldsymbol{C}_y$ was calculated to be

$$\boldsymbol{C}_y = \frac{1}{N}\sum_{i}^{N} \boldsymbol{y}_i\boldsymbol{y}_i^h. \quad (54)$$

The square root of this matrix was calculated as

$$\boldsymbol{C}_y^{1/2} = \frac{1}{N}\sum_{i}^{N} \lambda_i^{1/2}\boldsymbol{u}_i\boldsymbol{u}_i^h \quad (55)$$

where λ_i are the eigenvalues, and $\boldsymbol{u}_i$ are the corresponding eigenvectors of the covariance matrix. The stochastic data vector $\boldsymbol{x}_t$ was generated by transforming a vector n_t of complex Gaussian independent variables (with a mean of zero and variance of one) by the square root of the covariance matrix:

$$\boldsymbol{x}_t = \boldsymbol{C}_y^{1/2}\boldsymbol{n}_t. \quad (56)$$

B. Simulations Using Gain Annealing

A network of 100 neuron was used to discretize the bearing space between 0 and 45 degrees. Three sources at 12.0, 20.0 and 32.0 degrees were used to generate the data at 5 dB signal-to-noise ratio (SNR). Following gain annealing scheme was used:

$$\lambda_t = 10.0 \times (0.85)^t. \quad (57)$$

The dynamic equation ((11)) was discretized to the following form:

$$u_i(t+1) = \left(1 - \frac{1}{\tau}\right)u_i(t) + \sum_{j}^{P} t_{ij}\nu_j + I_j. \quad (58)$$

The values of t_{ij} and I_i were given by (51) and (52). Twenty-five data snapshots were used. Figs. 3 and 4 show the evolution of the network and the direction estimates. The hypothesis of the 28th, 46th, and 72nd neurons represent the position of the sources. The network settled to the digital value in 14 neural update iterations. The variance of the estimate at different noise levels is shown in Fig. 5.

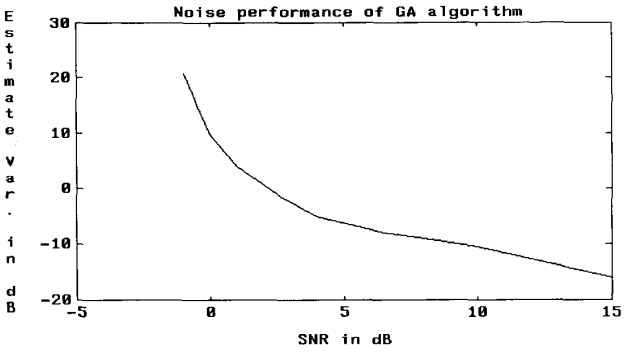


Fig. 5. A 100 neurons were used to discretize the bearing space between 0.0 and 45.0 degrees. The gain annealing procedure was used to find the estimates.

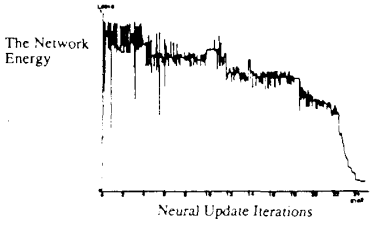


Fig. 6 The figure shows the energy evolution of the stochastic network. A 100 neurons were used discretize 0.0 to 10.0 degrees. One source at 7.5 (SNR 0.0 dB) was used to generate the data.

C. Simulations Using Stochastic Network

To test the algorithm for higher resolution estimates, a network of 100 neurons were used to discretize the bearing space between 0 and 10 degrees. The dynamic equation of the SN was discretized to the following form:

$$u_i(t+1) = \left(1 - \frac{1}{\tau}\right) u_i(t) + \sum_j^P t_{ij}\nu_j + I_i + 2\sqrt{2T\lambda}\,\cosh\frac{u_i(t)}{2\lambda}\eta_i(t). \tag{59}$$

The following cooling schedules were used for T and λ:

$$T_t = \left[\frac{T_0}{\log(t+2)}\right]^{-2}, \qquad \lambda_t = \frac{\lambda_0}{\log(t+2)}. \tag{60}$$

The initial value of T_0 and λ_0 were chosen to be 5.0 and 2.0 respectively. One source at 7.5 degrees was used to generate the data at 0.0 dB. Fig. 6 shows the energy evolution of the network. The SN settled into a solution after around 2500 neural update iteration. There are a number of parameters like T_0, λ_0, in the SN affect the performance of this algorithm; the values for these were determined by *ad hoc* means.

D. Simulation Using Iterated Descent

A network of 100 neurons were used to discretize the bearing-space between 0.0 and 10.0 degrees. The network was allowed to evolve using the gain annealing procedure until the following condition was satisfied:

$$\sum_{i=1}^{p} |\nu_i(t+1) - \nu_i(t)| < 0.000001. \tag{61}$$

Noise was added to the network dynamics using the stochastic network (60) for one iteration. In these simulations, the value of λ was not annealed: it was kept constant throughout at 2.0. The temperature cooling schedule was as follows:

$$T_t = \frac{1}{2t} \tag{62}$$

and 25 data snapshots were used to determine the minimum-mean square mapping. One source at 3.75 degree was used to determine the variance of the estimate at various SNRs. Each simulation was run for 250 ID cycles; on average each ID cycle took only 7 neural update iteration to satisfy (62). Fig. 7 presents a plot of variance of estimate versus SNR. The estimates become unreliable around the 0.0-dB SNR.

V. The Implementation on the Transputer Array

The computation involved in the neural bearing estimator can be divided into two distinct parts: the preprocessing required to calculate t_{ij} and I_i from the incoming data and the hypothesis vectors, and the computation involved in executing the neural dynamic equation until the network settles into a solution. The preprocessing requires simple linear algebraic evaluations (matrix-matrix and matrix–vector multiplication), and require $0(n^3)$ real operation. Well established parallel array architectures exist for these operations [23]; here only the mapping of the neural-update procedure onto a parallel array architecture is considered.

A. The Neural-Update Computation

The dynamics of the analog Hopfield network is defined by the differential equation ((11)), and it is normally solved using the Newton–Raphson method. However, to make the update computation more amenable to implementation, this differential equation is discretized to the form of (58). Kung *et al* [8] have proposed the use of a systolic array to implement this equation in parallel, but their scheme does not implement

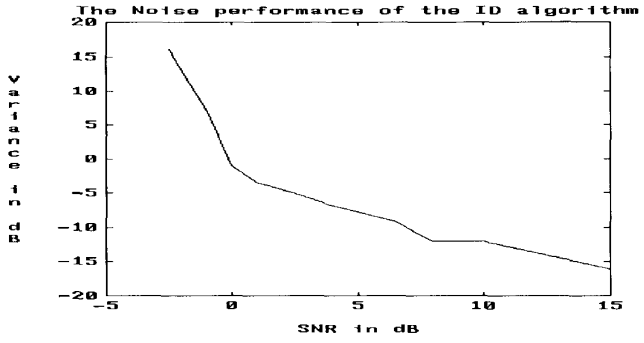


Fig. 7. A 100 neurons were used to discretize the bearing space between 0.0 and 10.0 degrees. The ID procedure was allowed to continue for 250 iterations.

a random asynchronous update. The Hopfield model requires that randomly selected neurons update their output upon randomly sampling the output of other neurons in the network, but as each neuron required the knowledge of the output-states of all the neurons in the network to update itself, this has proved difficult to implement in parallel. We have implemented this computation on a closely-coupled array of transputers[1], using a technique called *chaotic relaxation* [10]. Under this scheme, a variable block of neurons is assigned to each processor, and the processors are allowed to become *out of step* with each other, thus eliminating the communications bottleneck.

B. The Architecture

Fig. 8 shows the architecture of the transputer array used for performing the random asynchronous neural update computation. Fig. 8(a) shows the architecture for 4 processors and Fig. 8(b) shows the same for a 9 processor array. According to their functionality, the processors can be divided into three classes:

The Worker Processors (WP): The WPs are allocated a block of neurons, and these neurons are updated using (58). Each WP communicates to and receives from its neighbors the updated output states of the neurons allocated to it. WPs also communicate the updated states of the neurons to the master or controller processor, and receive the full updated $\boldsymbol{v}$ vector from them. All communications are carried out asynchronously.

The Controller Processors (CP): The CPs are only necessary for larger arrays. The task of the CPs is to increase the level of hierarchy and help communications. It receives updated parts of the output state vectors from the WPs connected to them, and passes it on to the master processor. CPs also receive the fully updated output state vectors from the MP and pass it on to the WPs. The CPs may themselves be allocated a block of neurons to update.

The Master Processor (MP): The MP receives blocks of updated output state vectors from CPs or WPs (depending upon the size of the array). It forms the fully updated output vector, checks for convergence ((58)), and passes on the updated vector to the processors connected to it.

[1]Transputer is a microprocessor specifically designed for parallel processing. It has four fast point-to-point links that can be used to connect several processors in an array without the need for additional hardware. The Transputer array was programmed using PAR_C [24].

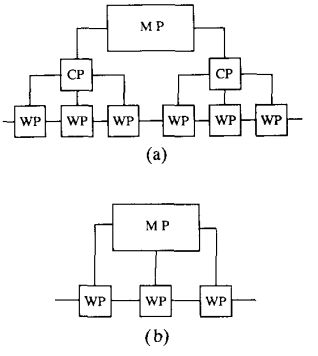


Fig. 8. Closely coupled transputer arrays for the implementation of the Hopfield mode neural network. (a) A nine-processor array. (b) A four-processor array. Legend: MP=Master Processor; CP=Controller Processor; WP=Worker Processor.

The effect of this scheme is that each WP has updated information about the states of its neighboring neurons (because of direct communication between neighboring WPs), but less updated information about farther away neurons. This information reaches the WPs through the CPs and MP. This effect correlates closely with the greater effect of the neighbor on the neural state in biological neuron.

C. Results

Two arrays of 4 and 9 transputers were used to update two networks of 88 and 180 neurons. The results of the time required for the network to settle are shown in Table I and Table II. T_{init} is the time required for the preprocessing, while T_{iter} is the time required for the neural update computation. It should be noted that the number of updates required are

$$\text{No. of neural updates} = \frac{\text{No. of iterations} \times \text{No. of neurons}}{\text{No. of processors}}. \tag{63}$$

Therefore, for larger arrays, the number of neural updates required for convergence were smaller.

TABLE I
THE TIME (IN SECONDS) FOR THE NEURAL BEARING ESTIMATOR IMPLEMENTATION ON THREE DIFFERENT ARRAYS[a]

	1 Transputer	4 Transputers	8 Transputers
T_{init}	16.551	8.084	6.971
T_{iter}	11.945	1.826	0.458
T_{total}	28.496	9.910	7.429
Iterations	17	26	35

[a] For these simulations: The number of neurons in network=88; number of sensors in the array=10.

TABLE II
THE TIME (IN SECONDS) FOR THE NEURAL BEARING ESTIMATOR IMPLEMENTATION ON THREE DIFFERENT ARRAYS[a]

	1 Transputer	4 Transputers	8 Transputers
T_{init}	54.707	21.168	16.177
T_{iter}	49.063	7.073	2.165
T_{total}	103.77	28.241	18.342
Iterations	17	24	35

[a] For these simulations: the number of neurons in network=180; number of sensors in the array=10.

VI. CONCLUSION

This paper has presented two mappings of the DoA problem onto the energy function of the Hopfield model neural networks. The first, the *p-neuron* model, requires only p neurons to represent the hypothesis, but at the expense of data-dependent synaptic matrix, which requires computationally expensive preprocessing. The second mapping, the *p^3-neuron* model, on the other hand has a data-dependent synaptic matrix, and it requires only a nominal preprocessing to calculate the external stimulus. However, the *p^3-neuron* model requires greater amount of hardware, and presents significantly harder optimization problem.

It is in attempting to solve the problems of function optimization and neural implementation on parallel arrays that this paper makes novel contributions. Firstly, to increase the probability of finding the global minimum of the cost function, novel modifications to the Hopfield model network dynamics have been proposed. Iterated descent (ID) combines the benefits of the fast relaxation of the Hopfield model with the convergence properties of the stochastic networks. Simulation results have been presented to characterize the performance of this algorithm in solving the DoA problem. Algorithmically, the neural approach offers the following benefits over the previously proposed methods: it does not require an estimate of the direction of sources, and it offers low variance estimates for correlated sources under limited data conditions (only 25 snapshots of data were used for the simulation results presented). The neural method has also been applied to real data with success [25].

Secondly, in the absence of practical analog neural devices, a novel parallel array architecture has been presented in this paper for performing the random asynchronous neural-update computation, and results have been presented to show the fast relaxation of the Hopfield network. The proposed architecture offers flexibility of mapping and the potential for acquiring real-time DoA estimators.

ACKNOWLEDGMENT

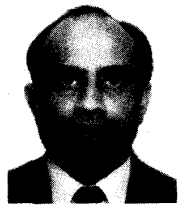
The authors would like to thank Gary Kelieff for his help in programming the transputer array, and Prof. Moss Kaveh for many valuable discussions.

REFERENCES

[1] S. M. Kay and S. L. Marple, Jr., "Spectrum analysis—A modern perspective," *Proc. IEEE*, vol. 69, Nov. 1981, pp. 1380–1419.
[2] D. H. Johnson and S. R. DeGraaf, "Improving resolution of bearing in passive sonar arrays by eigenvalue analysis," *IEEE Trans. Acoust. Speech Signal Processing* vol. ASSP-30, Aug. 1982.
[3] M. Wax and T. Kailath, "Determining the number of sources by information theoretic criteria," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing*, San Diego, CA, 1984, pp. 6.3.1–6.3.4.
[4] A. Paulraj and T. Kailath, "Eigenstructure methods for direction of arrival estimation in the presence of unknown noise fields," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-34, Feb. 1986.
[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation," in *Parallel Distributed Processing*, Rumelhart and McClelland, Eds. Cambridge, MA: The MIT Press, 1986, ch. 8.
[6] J. J. Hopfield and D. W. Tank, "Neural Computation of decisions in optimisation problems," *Biol Cybern.* vol. 52, pp. 141–152, 1985.
[7] A. Murray and A. V. W. Smith, "Asynchronous VLSI neural networks using pulse stream arithmetic," *IEEE J. Solid State Circuits Syst.*, vol. 23, no. 3, pp. 688–697, 1988.
[8] S. Y. Kung and J. N. Hwang, "Parallel architecture for artificial neural nets," *Proc. 1st IEEE International Conf. Neural Networks*, San Diego, 1987.
[9] B. Forrest, D. Roweth, N. Stroud, D. Wallace, and G. Wilson, "Implementing neural network models on parallel computers," *The Comput. J.*, vol. 30, no. 5, 1987.
[10] D. Chazan and W. Miranker, "Chaotic relaxation," *Linear Algebra and Its Applications*, vol. 2, no. 2, pp. 199–222, Apr. 1969.
[11] J. J. Hopfield, "Neural Networks and physical systems with emergent collective computational abilities," in *Proc. Natl. Acad. Sci.*, Apr. 1982.
[12] R. Rastogi, P. K. Gupta, and R. Kumerason, "Array signal processing with inter-connected neuron-like elements," in *Proc. Int. Conf. Acoust. Speech Signal Processing*, 1987, pp. 54.8.1–54.8.4.
[13] S. Jha, R. Chapman, and T. S. Durrani, "Investigation into neural networks for bearing estimation," in *Signal Processing-IV: Theories and Applications*, J. L. Lacoume, A. Chehikean, N. Martyin, and J. Malbos, Eds. Elsevier Science Publishers, 1988.
[14] D. Goryn and M. Kaveh, "Neural networks for narrowband and wideband direction finding," in *Proc. Int. Conf. Acoust., Speech Signal Processing*, 1988, pp. 2164–2167.
[15] S. Jha and T. Durrani, "Bearing Estimation using neural optimisation methods," *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing*, Albuquerque, NM, 1990, pp. 889–892.
[16] J. Cadzow, "Direction finding: The signal subspace approach," submitted to *IEEE Trans. Acoust. Speech Signal Processing*.
[17] K. Sharman, "Maximum likelihood parameter estimation by simulated annealing," in *Proc. Int. Conf. Acoust. Speech Signal Processing*, 1988, pp. 2741–2744.
[18] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, May 13, 1983.
[19] H. Szu, "Fast simulated annealing," *AIP Conf. Proc. 151: Neural Network for Computing*, Snowbird, UT, 1986.
[20] D. H. Ackley, G. E. Hinton, and T. J. Sjenowski, "A learning algorithm for Boltzmann machine," *Cognitive Sci.*, vol. 9, pp. 147–169, 1985.
[21] B. C. Levy, M. B. Adams, "Global optimisation with stochastic neural networks," *Proc. 1st IEEE Conf. Neural Networks*, San Diego, pp. 111-688, June 1987.
[22] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci.*, vol. 81, 1984, pp. 3088–3092.
[23] S. Y. Kung, "On Supercomputing with systolic/wavefront arrays," *Proc. IEEE*, vol. 72, no. 3, 1984.
[24] INMOS, *Transputer Application Notebook: Architecture and Software*, Inmos. Ltd.
[25] M. Kaveh, personal communications, July 1990.

Sanjay Kumar Jha was born in Ranchi (Bihar), India, on the 25th of February, 1963. He received the B. Eng. degree in electronic engineering in 1984 from the University of Liverpool, England, and the Ph.D. degree in electrical engineering in 1990 from the University of Strathclyde, Glasgow, Scotland.

After earning his first degree, he worked as a Research Engineer at the GEC Hirst Research Laboratory in London, U.K. between September 1984 and August 1986. In this period he worked on the design and implementation of the bit-level systolic convolver and contributed to the Alvey programme entitled *VLSI: Algorithms and Architectures*. Since August 1986 he has held the position of *Research Fellow* at University of Strathclyde. His research interests include the implementation of novel neural and parallel architectures in VLSI for applications in digital signal processing problems.



Tariq S. Durrani (M'79–F'89) was born in Amraoti, India on October 27, 1943. He received the B.Sc. (Hons) in electrical engineering in 1965 from the University of Engineering and Technology, Dacca, Bangladesh, and the M.Sc. and Ph.D. degrees in electronics in 1967 and 1970, respectively, from the University of Southampton, Southampton, England.

He worked as a SERC/MOD Research Fellow at Southampton from 1970 to 1975. He joined the University of Strathclyde, Glasgow, Scotland, in 1976, and is now Deputy Principal (Information Technology) and Professor of Signal Processing in the Department of Electronic and Electrical Engineering.

Dr. Durrani has extensive experience of ASSP Society activities. He was an Associate Editor of the IEEE TRANSACTIONS ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING from 1980 to 1985. He has been a member of the ASSP Technical Committee on Spectral Analysis from 1985–1988. He is on the Editorial Board of the ASSP Magazine, and was the General Chairman of ICASSP 89. He is a member of the IEEE Signal Processing Society Administrative Committee, and its Conference Board. He is a member of the Editorial Board of the *Eurasip Journal of Signal Processing*, the *J. Wiley International Journal of Adaptive Control and Signal Processing*, and the *International Journal of Imaging Systems and Technology*. He was Guest Editor of the *Proceedings of the IEE* (UK), Special Issue on Spectral Analysis (April 1983) and of the *Journal of Microprocessors and Microsystems*, Special Issue on Signal Processing Devices (December 1983). He is the Director of the Parallel Signal Processing Centre and the Scottish Regional Transputer Support Centre set up by the UK Government in Glasgow. He is Chairman of the UK Institution of Electrical Engineers (IEE) Professional Group on Signal Processing, and Past-Chairman of the Professional Group on Image Processing. He is the author/co-author of more than 100 papers and four books. His current research interests are in the area of algorithms and architectures for signal processing.