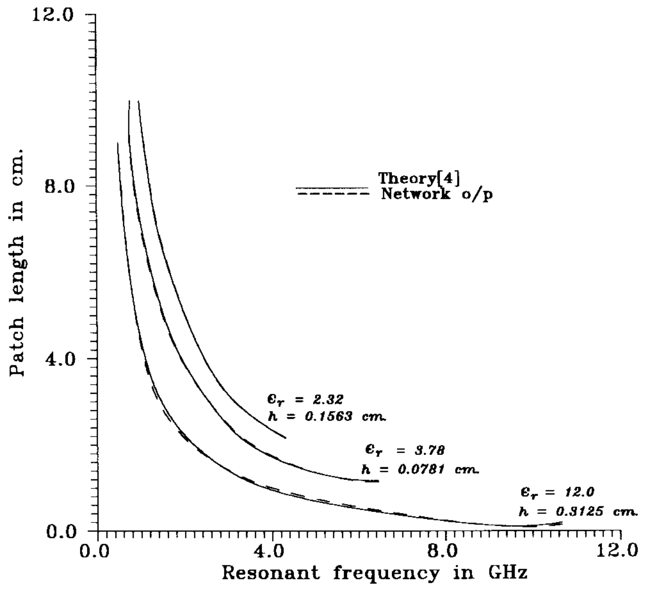

Fig. 1. Comparison of network output with theory [4].

objective function $G(\lambda) = \lambda E + (1-\lambda)E'$, where E is the mean-square error function. λ changes during the training according to the rule $\lambda = \lambda(E) = \exp(-\mu/E^2)$.

Out of the 1290 data generated, 645 were used for training and the rest were used for testing of the trained neural network. The optimized values of different parameters, which are obtained by trial and error for the training of the network, are: 1) the number of input nodes $= 3$; 2) the number of neurons in the hidden layer $= 40$; 3) the number of output node $= 1$; 4) the learning rate parameter $(\alpha) = 0.002$; 5) the learning rate adaption $(\mu) = 1.5$; and 6) momentum $(\beta) = 5.0$.

IV. Results

Network testing was performed both for theoretical and experimental data. Fig. 1 demonstrates the comparison between theoretical patch length and that obtained using the developed ANN model. A square-patch antenna with $l = 4.14$ cm, $\epsilon_r = 2.5$, $h = 0.1524$ cm is experimentally found to resonate at 2.228 GHz. When the developed network model is used for the same antenna, the patch length is found to be 4.151 cm.

V. Discussion

A distinct advantage of neurocomputing is that, after proper training, a neural network completely bypasses the repeated use of complex iterative processes for new cases presented to it. Our model produces the length of the side of a square patch almost instantaneously for the other three specified parameters of the square-patch antenna. The training time on an HP 9000/712 workstation platform is 13 min. Besides that, this single network structure can predict the results for frequency range up to 15 GHz provided the values of ϵ_r and h are in the domain of training values.

VI. Conclusion

A neural network-based CAD model is developed for the design of a square-patch antenna, which is robust both from the angle of time of computation and accuracy.

References

[1] L. Vegni and A. Toscano, "Analysis of microstrip antennas using neural networks," *IEEE Trans. Magn.*, vol. 33, pp. 1414–1419, Mar. 1997.
[2] R. K. Mishra and A. Patnaik, "Design of circular microstrip antenna using neural network," *Inst. Electron. Telecommun. (IETE) Eng. J. Res.* vol. 44, nos. 1–2, pp. 35–39, Jan.–Apr. 1998.
[3] A. Patnaik, R. K. Mishra, G. K. Patra, and S. K. Dash, "An artificial neural network model for effective dielectric constant of microstrip line," *IEEE Trans. Antennas Propagat.*., vol. 45, p. 1697, Nov. 1997.
[4] I. Wolf and N. Knoppik, "Rectangular and circular microstrip disk capacitors and resonators," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-22, pp. 857–864, Oct. 1974.
[5] N. B. Karayiannis and A. N. Venetsanopoulos, "Fast learning algorithm for neural networks," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Processing*, vol. 39, pp. 453–473, 1992.
[6] S. Haykins, *Neural Networks: A Comprehensive Foundation*. New York: IEEE Press/IEEE Computer Society Press, 1994.

Neural Network-Based Adaptive Beamforming for One- and Two-Dimensional Antenna Arrays

A. H. El Zooghby, C. G. Christodoulou, and M. Georgiopoulos

***Abstract*— In this letter, we present a neural network approach to the problem of finding the weights of one- (1-D) and two-dimensional (2-D) adaptive arrays. In modern cellular satellite mobile communications systems and in global positioning systems (GPS's), both desired and interfering signals change their directions continuously. Therefore, a fast tracking system is needed to constantly track the users and then adapt the radiation pattern of the antenna to direct multiple narrow beams to desired users and nulls interfering sources. In the approach suggested in this paper, the computation of the optimum weights is accomplished using three-layer radial basis function neural networks (RBFNN). The results obtained from this network are in excellent agreement with the Wiener solution.**

***Index Terms*—Adaptive arrays, antenna arrays, beamforming, neural network applications, tracking.**

I. Introduction

Neural networks are gaining momentum in the field of signal processing [1], [2] mainly because of their general-purpose nature, fast convergence rates, and new very large scale integration (VLSI) implementations. Motivated by these inherent advantages, this paper presents the development of a neural network-based algorithm to compute the weights of an adaptive array antenna [3]. In this new

Manuscript received November 14, 1997; revised July 27, 1998. This work was supported by the Florida Space Consortium and NeuralWare Inc.

A. H. El Zooghby and M. Georgiopoulos are with the Electrical and Computer Engineering Department, University of Central Florida, Orlando, FL 32816 USA.

C. G. Christodoulou is with the Electrical and Computer Engineering Department, University of New Mexico, Albuquerque, NM 87131 USA.

Publisher Item Identifier S 0018-926X(98)09723-3.

approach, the adaptive array can detect and estimate mobile users' locations, track these mobiles as they move within or between cells, and allocate narrow beams in the directions of the desired users while simultaneously nulling unwanted sources of interference. This adaptive antenna results in an increased system capacity for the existing cellular and mobile communications systems. In this letter, a brief derivation of the optimum array weights for adaptive beamforming using the radial basis function neural networks (RBFNN) approach is presented and discussed.

II. Adaptive Beamforming

Consider a linear array of M elements. We can write the array output on the M-dimensional vector

$$\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t) + \mathbf{N}(t) \tag{1}$$

where $\mathbf{A}$ is the steering matrix and $\mathbf{S}$ is the vector of received signals. For a general $M \times N$ rectangular array, the received signal data can be arranged in a $1 \times MN$ vector given by

$$\mathbf{X}(t) = \sum_{i=1}^{K} s_i(t)\mathbf{A}_i + \mathbf{N}(t) \tag{2}$$

where $\mathbf{N}(t)$ represents the noise. To derive the optimal weight vector, the array output is minimized so that the desired signals are received with specific gain, while the contributions due to noise and interference are minimized. This yields

$$\hat{\mathbf{W}}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{S}_d\left[\mathbf{S}_d^{\mathbf{H}}\mathbf{R}^{-1}\mathbf{S}_d\right]^{-1}\mathbf{r} \tag{3}$$

where $\mathbf{R}$ is defined as $E\{\mathbf{X}(t)\mathbf{X}^H(t)\}$ and $\mathbf{S}_d$ is the steering matrix pointing to the desired signals given by

$$\mathbf{S}_d = [\mathbf{S}_d(\theta_1) \quad \mathbf{S}_d(\theta_2) \quad \cdots \quad \mathbf{S}_d(\theta_V)] \tag{4}$$

and $\mathbf{S}_d(\theta_i)$ is defined as

$$\mathbf{S}_d(\theta_i) = \begin{bmatrix} 1 & e^{-jk_i} & e^{-j2k_i} & \cdots & e^{-j(M-1)k_i} \end{bmatrix} \tag{5}$$

and $\mathbf{r}$ is the $V \times 1$ constraint vector, where V is the number of desired signals.

III. Neural Network-Based Interference Cancellation

The optimum weight vector is a nonlinear function of the correlation matrix and the constraint matrix. Therefore, it can be approximated using a suitable architecture such as a RBFNN [4], [5]. The correlation matrix $\mathbf{R}$ is presented to the input layer of the RBFNN, and the vector $\mathbf{W}_{\text{opt}}$ is produced at the output layer consisting of J nodes ($J = 2M$ or $2MN$) and the hidden layer. As it is the case, with most neural networks, the RBFNN is designed to perform an input–output mapping trained with examples $(\mathbf{R}^l; W_{\text{opt}}^l)$; $l = 1, 2, \cdots, N_T$, where N_T stands for the number of examples contained in the training set. The purpose of the hidden layer in a RBFNN is to transform the input data $\mathbf{R}$ from an input space of dimensionality D to a space of higher dimensionality L, where L is the number of hidden nodes. An unsupervised learning algorithm (such as the K-Means [6]) is initially used to identify the centers of the Gaussian functions used in the hidden layer. Then, an *ad hoc* procedure is used to determine the widths (standard deviations) of these Gaussian functions. According to this procedure the standard deviation of a Gaussian function of a certain mean is the average distance to the first few nearest neighbors of the means of the other Gaussian functions. The weights from the hidden layer to the output layer are identified by following a supervised learning procedure, applied to a single layer network (the network from hidden to output layer). This supervised rule is referred to as the *delta rule*. In the *performance phase*, the neural network is supposed to generalize, that is respond to inputs ($\mathbf{R}$'s) that it has never seen before, but drawn from the same distribution as the inputs used in the training set. During the performance phase, the RBFNN produces outputs to previously unseen inputs by interpolating between the inputs used (seen) in the training phase.

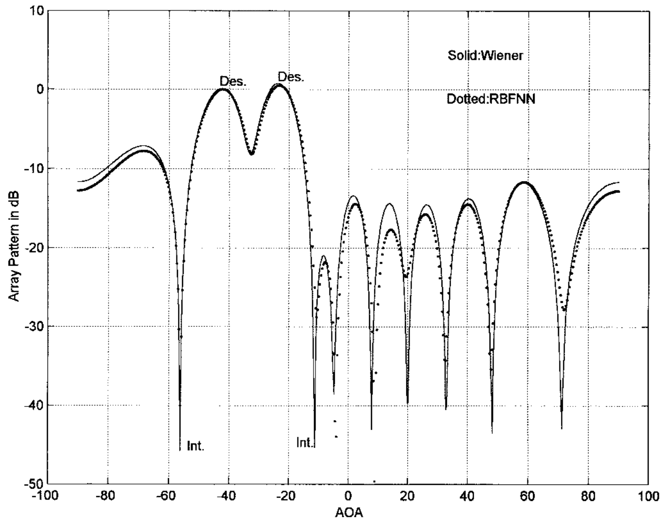

Fig. 1. Adapted pattern of a ten-element linear array for two desired and two interfering signals with $\Delta\theta = 15°$.

A. Generation of Training Data

First the correlation matrix $\{\mathbf{R}^l;\ l = 1, 2, \cdots, N_T\}$ is generated and arranged in a vector. Each vector is then normalized by its norm. Next, using (3), the vectors $\{\mathbf{W}_{\text{opt}}^l;\ l = 1, 2, \cdots, N_T\}$ are evaluated to produce the required training input/output pairs of the training set; that is $\{(\mathbf{R}^l; \mathbf{W}_{\text{opt}}^l);\ l = 1, 2, \cdots, N_T\}$. In this application, the training data were generated by assuming that sources were located at elevation angles θ ranging from $-90°$ to $+90°$ for the one-dimensional (1-D) case. In the two-dimensional (2-D) array, in addition to angles θ, azimuth angles ϕ can be made to range from 0° to 360° in order to span the field of view of the antenna. Once the RBFNN is trained with a representative set of training input–output pairs it is ready to function in the performance phase.

B. Performance Phase of the RBFNN

1) Generate the array output vector $\hat{\mathbf{X}}(t)$. Normalize this array output vector by its norm.
2) Present the normalized array output vector at the input layer of the trained RBFNN. The output layer of the trained RBFNN will produce as an output the estimates of optimum weights for the array outputs (i.e., $\hat{\mathbf{W}}_{\text{opt}}$). Unlike the least mean square (LMS), recursive least squares (RLS), or the sample matrix inversion (SMI) algorithms, where the optimization is carried out whenever the directions of the desired or interfering signals change; in our approach the weights of the trained network can be used to produce the optimum weights needed to steer the narrow beams of the adaptive array in real time.

IV. Simulation Results

Figs. 1 and 2 show the adapted pattern of a ten-element linear array obtained from the RBFNN and how it compares with the optimum

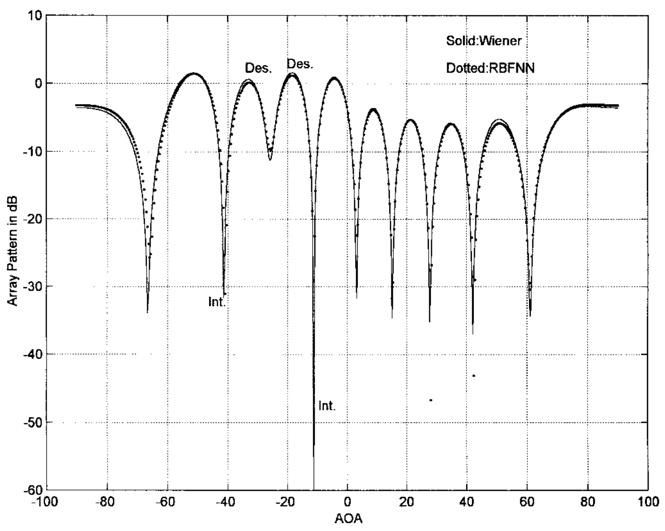

Fig. 2. Adapted pattern of a ten-element linear array for two desired and two interfering signals with $\Delta\theta = 10°$.

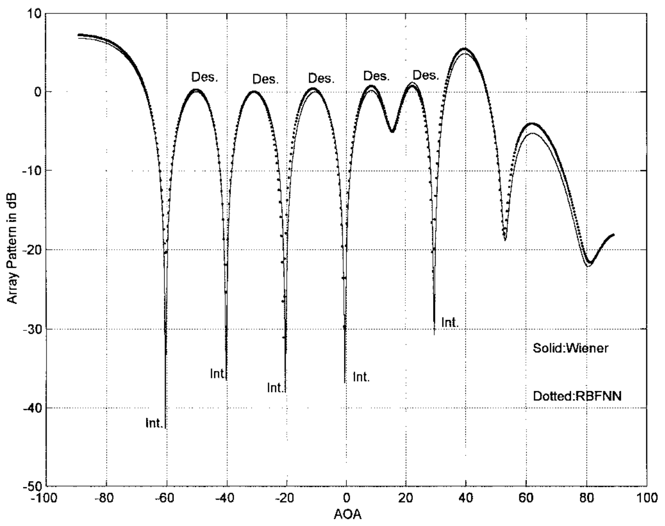

Fig. 3. An 8 × 8 rectangular array, tracking ten signals with $\Delta\theta = 10°$, $\phi = 30°$. Comparison between the RBFNN (dotted curve) and Wiener optimum weight solution (solid curve) at different source locations.

Wiener solution for angular signal separations of $\Delta\theta = 15°$ and $10°$, respectively. It can be concluded from these figures that the RBFNN produced a solution for the beamforming weight vector that is very close to the optimum solution. In practice, 2-D arrays are used to enable the system to track a larger number of users. Hence, for illustration purposes our method was applied to 2-D array of isotropic elements. In Fig. 3, an 8 × 8 array is used to track ten different users, with $\Delta\theta = 10°$ and $\phi = 30°$. The adapted pattern obtained from a RBFNN with 150 nodes in the hidden layer is compared with the optimum solution. The network successfully, tracked the desired signals and placed nulls in the direction of the interfering users.

V. Conclusion

A new approach to the problem of adaptive beamforming was introduced. The weights were computed using an RBFNN that approximates the Wiener solution. The network was successful in tracking multiple users while simultaneously nulling interference caused by cochannel users. Both 1-D and 2-D arrays were simulated and the results have been very good in every case.

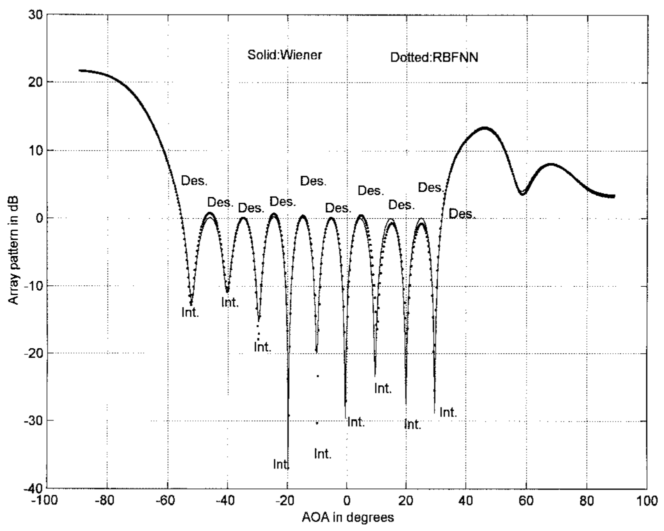

Fig. 4. Gain of a 10 × 10 rectangular array, tracking 19 signals (nine desired users, ten cochannel signals) with $\Delta\theta = 5°$, $\phi = 45°$. Comparison between the RBFNN (dotted curve) and Wiener optimum weight solution (solid curve).

References

[1] H. L. Southall, J. A. Simmers, and T. H. O'Donnell, "Direction finding in phased arrays with a neural network beamformer," *IEEE Trans. Antennas Propagat.*, vol. 43, p. 1369, Dec. 1995.
[2] A. H. El Zooghby, C. G. Christodoulou, and M. Georgiopoulos, "Performance of radial basis function networks for direction of arrival estimation with antenna arrays," *IEEE Trans. Antennas Propagat.*, vol. 45, pp. 1611–1617, Nov. 1997.
[3] Mozingo and Miller, *Introduction to Adaptive Arrays*. New York: Wiley, 1980.
[4] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Ontario, Canada: Macmillan College Publ., 1994.
[5] T. J. Moody and C. J. Darken, "Fast learning in networks of locally tuned processing units," *Neural Computation*, vol. 1, p. 281, 1989.
[6] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*. Reading, MA: Addison Wesley, 1976.