Neural Network based beamforming for interference cancellation

A.H.EL Zooghby, C. G. Christodoulou, and M. Georgiopoulos
Electrical and Computer Engineering Department
University of Central Florida
Orlando, Florida 32816

Abstract

A novel approach to the problem of finding the weights of an adaptive array is presented. In cellular and satellite mobile communications systems, desired as well as interfering signals are mobile. Therefore, a fast tracking system is needed to constantly estimate the directions of those users and then adapt the radiation pattern of the antenna to direct multiple beams to desired users and nulls to sources of interference. In this paper, the computation of the optimum weights is approached as a mapping problem which can be modeled using a suitable artificial neural network trained with input output pairs. A study of a three-layer Radial Basis Function Neural Network (RBFNN) is conducted. RBFNN were used due to their ability for data interpolation in higher dimensions. The network weights are modified using the normalized cumulative delta rule. The performance of this network is compared to the Wiener solution. It was found that networks implementing these functions were successful in tracking mobile users as they move across the antenna's field of view .

I. Introduction:

As the number of users in wireless and personal communications systems increase, the likelihood of interfering with one another increases and more efficient use of the available spectrum is required . Multiple access techniques are therefore used to maximize the number of users a system can accommodate. However, bandwidth limitations in a Frequency Division Multiple Access (FDMA) system, or time limitations in a Time Division Multiple Access (TDMA) or limitations on the number of good codes in a Code Division Multiple Access (CDMA) system, limit the number of users that a system can accommodate simultaneously. Hence, frequency reuse is used to increase the capacity of a cellular system. With frequency reuse the same frequency is used in two different cells separated far enough so that users in one cell do not interfere with the users in the other cell. In satellite-based Personal Communication Systems (PCS), geosynchronous satellites can interfere with each other as well as with Low Earth Orbit satellites, which limits their capacity. Global Positioning Systems (GPS) applications will also experience narrowband and broadband interferences. Interference rejection is therefore essential, and often represents an inexpensive way to increase the system capacity by allowing closer proximity of cofrequency cells or beams providing additional frequency reuse[1] in a cellular system. Interference rejection can be accomplished in two steps. First, an Angle Of Arrival (AOA) estimation algorithm is used to locate desired as well as cochannel mobile users [2,3]. Then an adaptive array [4] can be designed so that it directs the maxima of its radiation pattern toward the mobiles of interest while it directs nulls of its radiation pattern toward the interfering sources [5]; this adaptive array is able to track the mobiles of interest in real-time.

Neural networks are gaining momentum in the field of signal processing [6,7] mainly because of their general-purpose nature, fast convergence rates, and new 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 [8]. In this new 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. The organization of the paper is as follows: In section II a brief derivation of the optimum array weights in 1-D adaptive beamforming is presented. In Section III the method is applied to 2-D arrays. The RBFNN approach for the computation of the adaptive array weights is introduced in section IV. Finally, Section V presents the simulation results and Section VI offers conclusive remarks.

II. Adaptive beamforming using 1-Dimensional linear arrays

Consider a linear array composed of M elements as shown in Figure 1 . Let K (K<M) be the number of narrowband plane waves, centered at frequency ω_0 impinging on the array from directions $\{\theta_1 \quad \theta_2 \quad \cdots \quad \theta_K$. Using complex signal representation, the received signal at the ith element can be written as,

$$x_i(t) = \sum_{m=1}^{K} s_m(t) e^{-j(i-1)k_m} + n_i(t) \qquad ; i = 1,2,\cdots M \tag{1}$$

where $s_m(t)$ is the signal of the mth wave, $n_i(t)$ is the noise signal received at the ith sensor and

$$k_m = \frac{\omega_0 d}{c} \sin(\theta_m) \tag{2}$$

where d is the spacing between the elements of the array, and c is the speed of light in free space. Using vector notation we can write the array output on the matrix forms:

$$\mathbf{X}\,(t) = \mathbf{A}\,\mathbf{S}(t) + \mathbf{N}(t) \tag{3}$$

Where, **X** (t), **S**(t) and **N**(t) are given by:

$$\mathbf{X}(t) = [x_1(t) \quad x_2(t) \quad \cdots \quad x_M(t)]^T \tag{4}$$

$$\mathbf{N}(t) = [n_1(t) \quad n_2(t) \quad \cdots \quad n_M(t)]^T \tag{5}$$

$$\mathbf{S}(t) = [s_1(t) \quad s_2(t) \quad \cdots \quad s_K(t)]^T \tag{6}$$

In (4) and (5) and (6) the superscript "T" indicates the transpose of the matrix. Also in (3) **A** is the MxK steering matrix of the array towards the direction of the incoming signals defined as:

$$\mathbf{A} = [\mathbf{a}(\theta_1) \quad \mathbf{a}(\theta_2) \quad \cdots \quad \mathbf{a}(\theta_K)] \ , \tag{7}$$

where $\mathbf{a}(\theta_i)$ is defined as

$$\mathbf{a}(\theta_i) = \left[1 \quad e^{-jk_i} \quad e^{-j2k_i} \quad \cdots \quad e^{-j(M-1)k_i}\right] \tag{8}$$

Assuming that the noise signals $\{n_i(t),\ i = 1{:}M\}$, received at the different sensors are statistically independent, white noise signals, of zero mean and variance σ^2 and also independent of S(t) , then the received spatial correlation matrix, **R**, of the received noisy signals can be expressed as:

$$\begin{aligned}\mathbf{R} &= E\{\mathbf{X}(t)\mathbf{X}(t)^H\} = A\,E[\mathbf{S}(t)\mathbf{S}^H(t)]A^H + E[\mathbf{N}(t)\mathbf{N}^H(t)] \\ &= APA^H + \sigma^2 I = \sum_{i=1}^{M} \lambda_i \ e_i e_i^H\end{aligned} \tag{9}$$

In the above equation, $\mathbf{P} = E\{\mathbf{S}(t)\mathbf{S}(t)^H$ designates the signal covariance matrix and I is the identity matrix. Also, in the above equation "H" denotes the conjugate transpose. Finally, λ_i ($1 \le i \le M$) and e_i ($1 \le i \le M$) stand for the eigenvalues and eigenvectors of the matrix **R**, respectively. The weights of the array element outputs can be represented as an M-dimensional vector:

$$\mathbf{W} = [w_1 \quad w_2 \quad \cdots \quad w_M]^T \tag{10}$$

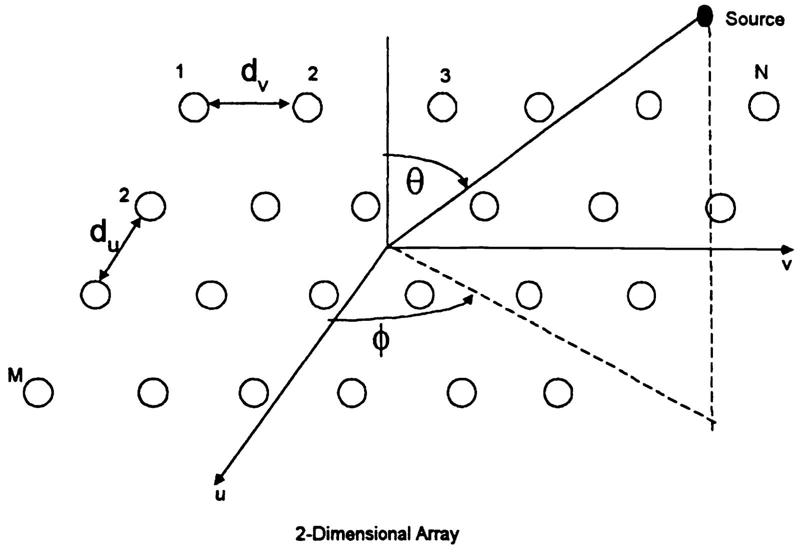
Then the array output becomes

$$y(t) = \sum_{i=1}^{M} w_i^* x_i(t) = \mathbf{W}^H \mathbf{X}(t) \tag{11}$$

The mean output power is thus given by:

$$P(\mathbf{W}) = E[y(t)y^*(t)] = \mathbf{W}^H \mathbf{R} \mathbf{W} \tag{12}$$

where * denotes the conjugate. 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. In other words:



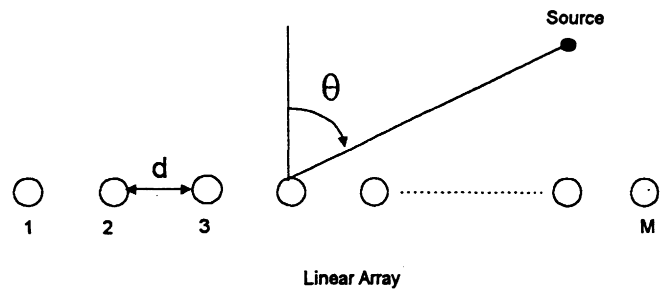


Figure 1 1& 2-D Array Geometry

$$\min \mathbf{W}^H \mathbf{R} \mathbf{W} \qquad \textit{subject to } \mathbf{W}^H \mathbf{S}_d = \mathbf{r} \tag{13}$$

In the above equation, **r** is the V x 1 constraint vector, where V is the number of desired signals, and S_d is the steering vector associated with the look direction as defined in (8). The method of Lagrange multipliers is used to solve the constrained minimization problem in (13). It can be shown that the optimum weight vector is given by the following equation:

$$\hat{\mathbf{W}}_{opt} = \mathbf{R}^{-1}\mathbf{S}_d\left[\mathbf{S}_d{}^{H}\mathbf{R}^{-1}\mathbf{S}_d\right]^{-1}\mathbf{r} \tag{14}$$

Since the above equation is not practical for real time implementation, an adaptive algorithm must be used to adapt the weights of the array in order to track the desired signal and to place nulls in the direction of the interfering signals.

III. Adaptive beamforming using 2-D rectangular arrays

Consider a general M x N rectangular array as shown in Figure 1 receiving K signals and let the received signal data matrix be given by [9]:

$$x_{mn}(t) = \sum_{i=1}^{K} s_i(t) A_{ui}(m) A_{vi}(n) + n_{mn}(t) \tag{15}$$

where

$$A_{ui}(m) = \exp\left\{ j(m-1)\frac{2\pi d_u}{\lambda}\sin\theta_i\cos\phi_i \right\} \tag{16}$$

$$A_{vi}(n) = \exp\left\{ j(n-1)\frac{2\pi d_v}{\lambda}\sin\theta_i\sin\phi_i \right\} \tag{17}$$

In the above equation , d_u and d_v are the spacings between the elements along the column and row directions, respectively, while θ_i and ϕ_i are the elevation and azimuth angles of the ith source, respectively, and m=1,2,...,M; n=1,2, ...,N; and i=1,2, ...,K.
The received signal data can be arranged in a 1x MN vector given by

$$\mathbf{X}(t) = \sum_{i=1}^{K} s_i(t)\mathbf{A}_i + \mathbf{N}(t) \tag{18}$$

In (16) the matrices X(t), and N(t) are defined as

$$\mathbf{X}(t) = [x_{11}(t) \quad x_{21}(t) \quad \cdots \quad x_{M1}(t) \quad x_{12}(t) \quad \cdots \quad x_{MN}(t)]^T \tag{19}$$

$$\mathbf{N}(t) = [n_{11}(t) \quad n_{21}(t) \quad \cdots \quad n_{M1}(t) \quad n_{12}(t) \quad \cdots \quad n_{MN}(t)]^T \tag{20}$$

while the ith signal direction vector $\mathbf{A}_i = \mathbf{A}_{vi} \otimes \mathbf{A}_{ui}$ is defined in terms of the Kronecker product of A_{vi} and A_{ui} ,which are given by

$$\mathbf{A}_{ui} = [A_{ui}(1) \quad A_{ui}(2) \quad \cdots \quad A_{ui}(M)]^T \quad \text{and}$$
$$\mathbf{A}_{vi} = [A_{vi}(1) \quad A_{vi}(2) \quad \cdots \quad A_{vi}(M)]^T \tag{21}$$

The Kronecker product of two matrices B (of size p x q) and C (of size m x n) is defined as

$$\mathbf{B}\otimes\mathbf{C} = \begin{bmatrix} b_{11}\mathbf{C} & b_{12}\mathbf{C} & \cdots & b_{1q}\mathbf{C} \\ b_{21}\mathbf{C} & b_{22}\mathbf{C} & \cdots & b_{2q}\mathbf{C} \\ \vdots & \vdots & \vdots & \vdots \\ b_{p1}\mathbf{C} & b_{p2}\mathbf{C} & \cdots & b_{pq}\mathbf{C} \end{bmatrix} \tag{22}$$

where **B**⊗**C** is an pm x qn matrix and b_{ij} are the elements of the matrix **B**. It can be shown that in this case, the vector of optimum weights is given by

$$\hat{\mathbf{W}}_{opt} = \mathbf{R}^{-1}\mathbf{S}_d\left[\mathbf{S}_d^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{S}_d\right]^{-1}\mathbf{r} \tag{23}$$

where **R** is defined as E{X(t) X^H(t)} and S_d is the steering vector associated with the desired signals given by $\mathbf{S_d} = \mathbf{S_{dv}} \otimes \mathbf{S_{du}}$ where S_{dv} and S_{du} are vectors defined as

$$\mathbf{S}_{du} = [S_{du}(1) \quad S_{du}(2) \quad \cdots \quad S_{du}(M)]^T \tag{24}$$

$$\mathbf{S}_{dv} = [S_{dv}(1) \quad S_{dv}(2) \quad \cdots \quad S_{dv}(N)]^T \tag{25}$$

whose elements are given by

$$S_{du}(i) = \exp\left\{ j(m-1)\frac{2\pi d_u}{\lambda}\sin\theta_i\cos\phi_i \right\} \tag{26}$$

$$S_{dv}(i) = \exp\left\{ j(n-1)\frac{2\pi d_v}{\lambda}\sin\theta_i\sin\phi_i \right\} \tag{27}$$

IV. Neural Network –based interference cancellation:

This section describes a new implementation for the problem of beamforming using neural networks. The optimum weight vector is a nonlinear function of the correlation matrix and the constraint matrix (see equations (14) and (23)). Therefore it can be approximated using a suitable architecture such as a Radial Basis Function Neural Network [11]. Note that a Radial Basis Function Neural Network can approximate an arbitrary function from an input space of arbitrary dimensionality to an output space of arbitrary dimensionality[10]. The block diagram of a RBFNN is shown in Figure 2. As it can be seen from Figure 2,the RBFNN consists of three layers of nodes, the input layer, the output layer and the hidden layer.In our application the input layer consists of $J=2M$ nodes (1-D array case), or $J=2MN$ nodes (2-D array case) to accommodate both the real and the imaginary part of the input vector (i.e., $\mathbf{X(t)}$). The output layer consists of $2M$ nodes (1-D case) or $2MN$ nodes (2-D case) to accommodate the output vector (i.e., $\mathbf{W}_{opt}$). As it is the case, with most neural networks the RBFNN is designed to perform an input-output mapping trained with examples ($\mathbf{X}^l$(t); $\mathbf{W}^l{}_{opt}$) ;$l = 1,2,\ldots,N_T$, 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 X(t) from an input space of dimensionality J to a space of higher dimensionality L (see Figure 2).There are a lot of learning strategies that have appeared in the literature to train a RBFNN. The one used in this paper was introduced in [11], where an unsupervised learning algorithm (such as the K-Means[12]) 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 aforementioned unsupervised learning procedure allows you to identify the weights (means and standard deviations of the Gaussian functions) from the input layer to the hidden layer. 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*. The delta rule is essentially a gradient decent procedure applied to an appropriately defined optimization problem. For more details about the delta rule, and how it is applied to single layer networks see[10]. Once training of the RBFNN is accomplished, the training phase is complete, and the trained neural network can operate in the performance mode (phase). In the *performance phase*, the neural network is supposed to generalize, that is respond to inputs (X(t)'*s*) that it has never seen before, but drawn from the same distribution as the inputs used in the training set. One way of explaining the generalization exhibited by the network during the performance phase is by remembering that after the training phase is complete the RBFNN has established an approximation of the desired input/output mapping. Hence, during the performance phase the RBFNN produces outputs to previously unseen inputs by interpolating between the inputs used (seen) in the training phase. The step-by-step procedure to produce the training data {$\mathbf{X}^l$(t); $\mathbf{W}^l{}_{opt}$;$l = 1,2,\ldots,N_T$}for the RBFNN in this application is provided below.

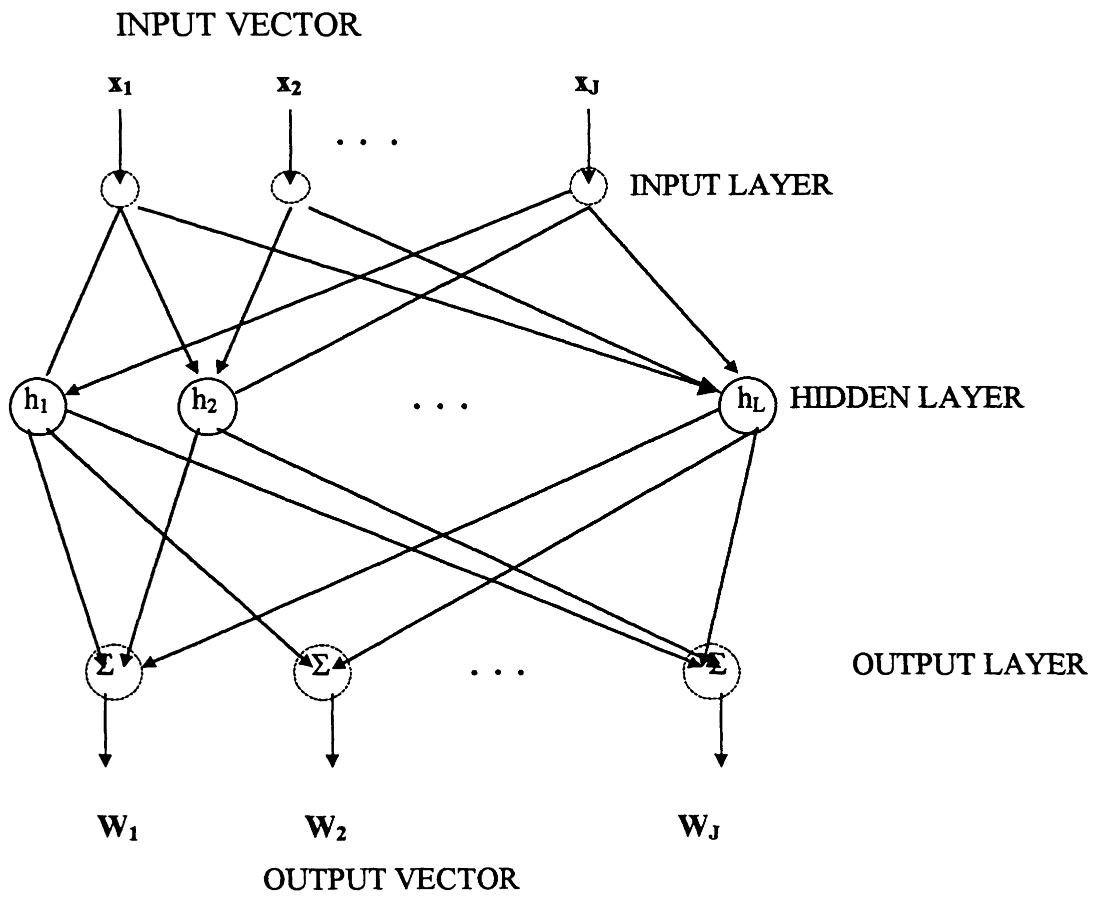


Figure 2 Architecture of a three layer RBFNN.

Generation of Training Data

1. Generate array output vectors $\{\mathbf{X}^l(t); l = 1,2,\ldots,N_T\}$ using equations (3) [1-D case] or (18) [2-D case].
2. Normalize each one of the above array output vectors by its norm. For simplicity of notation we still refer to these vectors by $\mathbf{X}(t)$'*s*.
3. Evaluate the correlation matrix $\mathbf{R}^l$ ($l = 1,2,\ldots, N_T$) for each of the array output vectors generated in Step 1; to do so use equation (9). Using the calculated $\mathbf{R}^l$'s calculate the vectors $\{\mathbf{W}^l_{\text{opt}}; l = 1,2,\ldots, N_T\}$from equation (14) [1-D case] and equation (23) [2-D case].
4. Produce the required training input/output pairs of the training set, that is $\{(\mathbf{X}^l(t); \mathbf{W}^l_{\text{opt}}) ; l = 1,2,\ldots,N_T\}$

In this application, the training data were generated by assuming that sources were located at elevation angles θ ranging from -90^0 to $+90^0$ with increments of $\Delta\theta$ for the 1-dimensional case. In the 2-dimensional array, in addition to angles θ, azimuth angles ϕ can be made to range from 0^0 to 360^0 in order to span the field of view of the antenna.

As we have emphasized before, once the RBFNN is trained with a representative set of training input/output pairs it is ready to function in the performance phase. In the performance phase, the RBFNN produces estimates of the optimum weights for the array outputs, through a simple, computationally inexpensive, two-step process, described below.

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., $\hat{\mathbf{W}}_{opt}$).

V. Simulation results

Figures 3 and 4 show the adapted pattern of a 10 element linear array obtained from the RBFNN and how it compares with the optimum Wiener solution for angular signal separations of 15^0 and 10^0, 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 Figure 5, an 8 x 8 array is used to track 10 different users, with $\Delta\theta = 15^0$ and $\phi = 30^0$. 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. Finally, an array of 10 x 10 elements was simulated to track 19 signals consisting of 10 desired users and 9 co-channel interferences. Figure 6 shows the adapted pattern as the network tracked all mobile users. The number of input/output pairs used in the training set for the 2-dimensional arrays was 181.

VI. 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. Comparison of the adapted pattern obtained by the RBFNN and the optimum solution proved the fast convergence rate of this approach as well as its high degree of accuracy. Future work will concentrate on (i) the effects of element patterns and (ii) finding the maximum number of simultaneous users that can be tracked by the antenna array using this new approach.

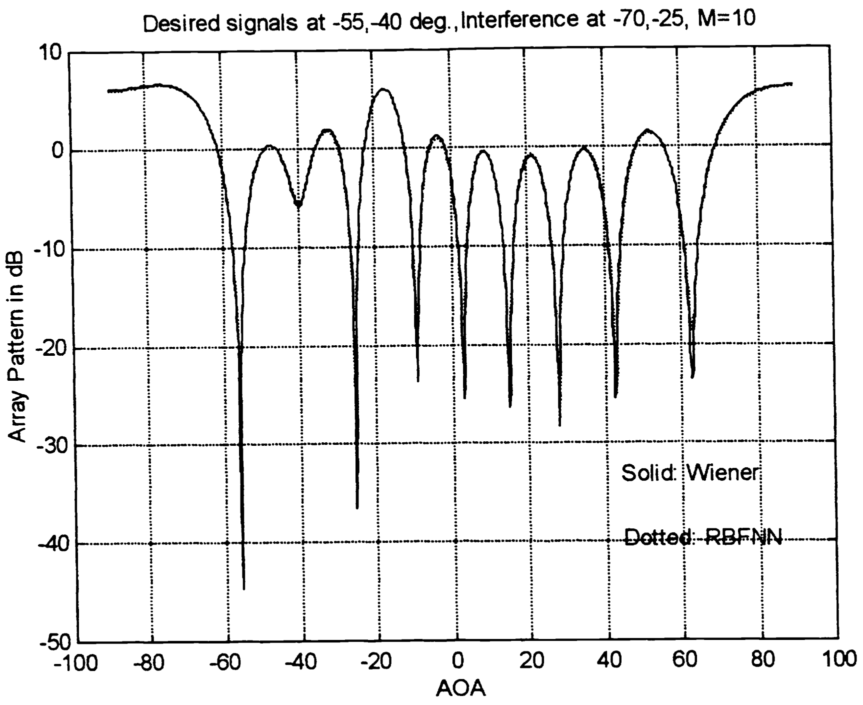


Figure 3 10 element linear array, tracking four signals , $\Delta\theta=15^0$.

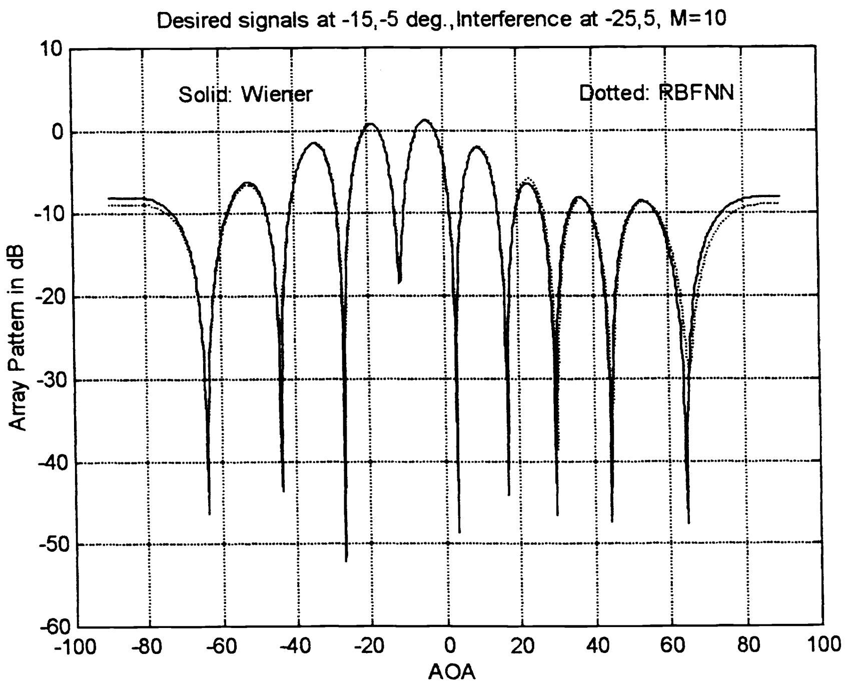


Figure 4 10 element linear array, tracking four signals , $\Delta\theta=10^0$.

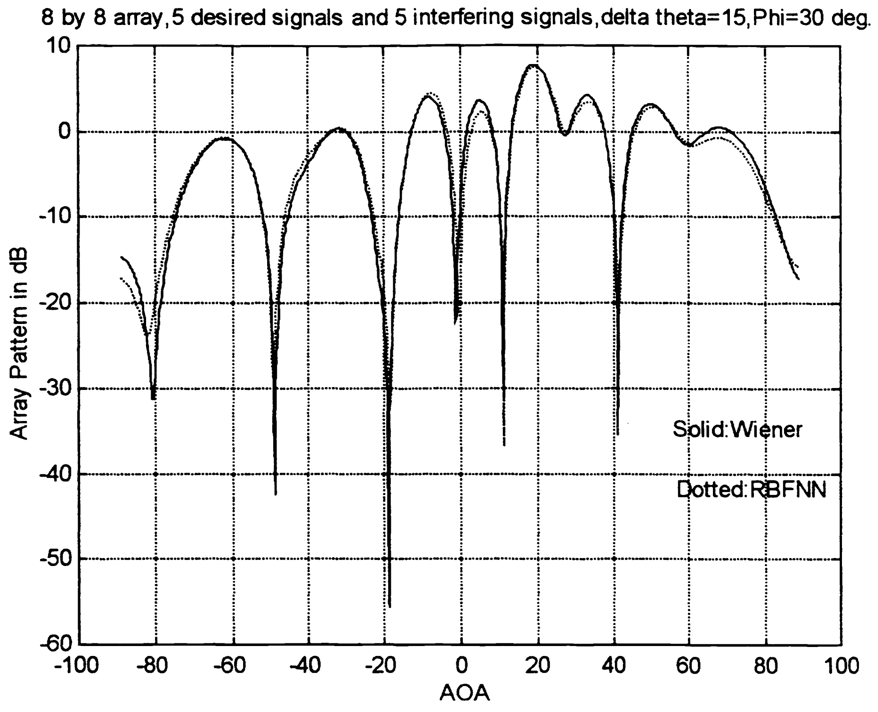


Figure 5 8x8 Array tracking 10 signals with $\Delta\theta=15^0, \phi=30^0$.

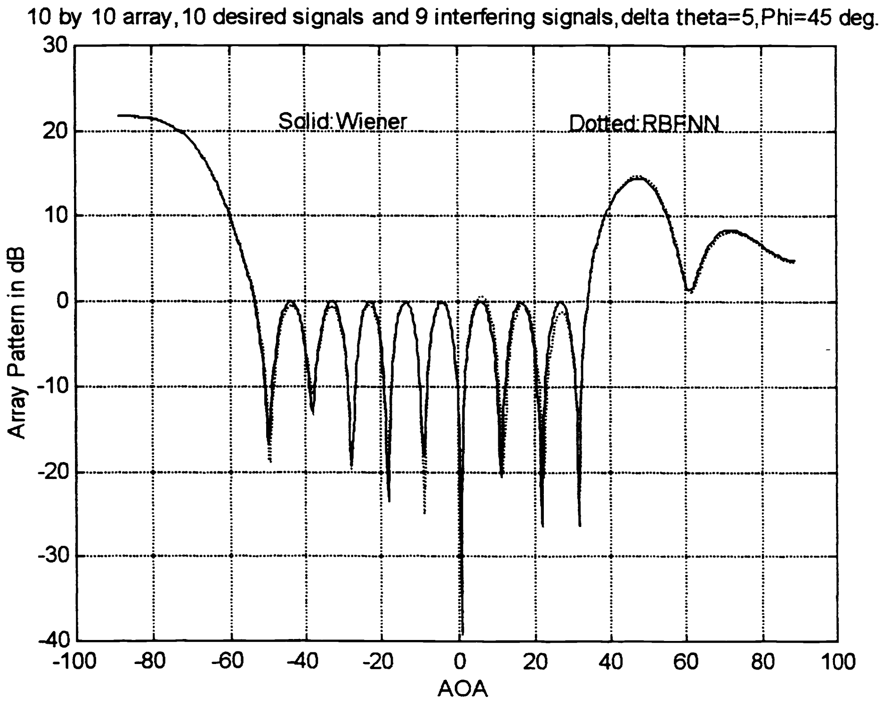


Figure 6 10x10 Array tracking 19 signals with $\Delta\theta=5^0, \phi=45^0$.

VII. References

[1] T.Gebauer, and H.G.Gockler, " Channel -individual adaptive beamforming for mobile satellite communications", *IEEE Journal on Selected Areas in Communications*, vol.13, No 2, pp. 439-448 ,February 1995.

[2] H.L.Southall, J.A.Simmers, and T.H.O'Donnell, " Direction finding in phased arrays with a neural network beamformer", *IEEE Transactions on Antennas and Propagation*, vol. 43, No.12, pp. 1369, December 1995.

[3] El Zooghby A. H., C.G. Christodoulou and M. Georgiopoulos," Performance of radial basis function networks for direction of arrival estimation with Antenna Arrays"; to appear in the *IEEE Trans .on Antennas and Propagation*, vol. 45, No.11, pp. 1611-1617, November 1997.

[4] Mozingo, Miller, Introduction to Adaptive arrays, John Wiley, 1980.

[5] A.F.Naguib, A.Paulraj, T.Kailath,"Capacity improvement with base-station antenna arrays in cellular CDMA", *IEEE Transactions Vehc.Technology*, vol. 43, No. 3,pp. 691, August 1994.

[6] Luo Long, Li Yan Da, "Real-time computation of the noise subspace for the MUSIC algorithm", *Proc. ICASSP* 1993, vol.1, pp.485 -488, April 1993.

[7] D.Goryn and M.Kaveh."Neural networks for narrowband and wideband direction finding", *Proc. ICASSP*, pp.2164-67, April 1988.

[8] P.R.Chang, W.H.Yang, and K.K.Chan,"A neural network approach to MVDR beamforming problem", *IEEE Transactions on Antennas and Propagation*, vol.40, No.3, pp. 313-322, March 1992.

[9] S.J.Yu, and J.H.Lee,"Design of Two-Dimensional Rectangular Array Beamformers with partial adaptivity", *IEEE Transactions on Antennas and Propagation*, vol.45, No.1, pp.157-167, January 1997.

[10] S.Haykin, Neural Networks A Comprehensive Foundation, Macmillan College Publishing, Ontario, 1994.

[11] T.J.Moody and C.J.Darken, "Fast learning in networks of locally tuned processing units", *Neural Computation*, vol.1, 281(1989).

[12] J.T.Tou and R.C.Gonzalez, Pattern Recognition Principles. Addison Wesley, Reading, MA, 1976.