Pattern Synthesis for Arbitrary Arrays Using an Adaptive Array Method

Philip Yuanping Zhou and Mary Ann Ingram

Abstract— This paper presents a new pattern synthesis algorithm for arbitrary arrays based on adaptive array theory. With this algorithm, the designer can efficiently control both mainlobe shaping and sidelobe levels. The element weights optimize a weighted L_2 norm between desired and achieved patterns. The values of the weighting function in the L_2 norm, interpreted as imaginary jammers as in Olen and Compton's method, are iterated to minimize exceedance of the desired sidelobe levels and minimize the absolute difference between desired and achieved mainlobe patterns. The sidelobe control can be achieved by iteration only on sidelobe peaks. In comparison to Olen and Compton's method, the new algorithm provides a great improvement in mainlobe shaping control. Example simulations, including both nonuniform linear and planar arrays, are shown to illustrate the effectiveness of this algorithm.

Index Terms—Antenna array synthesis, array pattern synthesis, beamformer.

I. INTRODUCTION

OVER the last several decades, there has been significant attention paid to the area of array pattern synthesis. A classic paper by Dolph [1] showed how to obtain the weights for an uniform linear array (ULA) to achieve a Chebyshev pattern, which is optimal in the sense that it yields a minimum uniform sidelobe level for a given mainlobe width. Other pattern synthesis approaches for ULA's have been presented in literature [2].

A more challenging problem is to synthesize patterns for arrays with arbitrary element positions. Perini [6] proposed a pattern synthesis method using a steepest descent technique for nonuniform arrays. The algorithm iteratively updates the weights of array while searching for the point of minimum sum of squared errors between the synthesized pattern and the desired pattern. Ng *et al.* [5] developed a noniterative method to minimize the L_2 norm using quadratic programming. Even though the L_2 norms could be weighted as noted by Perini to emphasize certain portions of the pattern, there was no method given to adjust such weighting values in the iterative process. As a result, the abovementioned algorithms can guarantee neither a particular pattern shape in certain area nor a specific response level over a sidelobe region. Tseng and Griffiths [3] proposed an algorithm that iterates the constraints in the solution of a linearly constrained least square problem to control sidelobe peaks.

A different approach to synthesis is to apply adaptive array theory. In an early paper, Sureau and Keeping [7] employed adaptive array techniques to synthesize the patterns for cylindrical arrays. The imaginary jammer powers were varied depending on desired sidelobe levels. Although they achieved reasonable sidelobe control, they did not provide a systematic approach to adjust jammer powers so that sidelobe levels could meet the desired specifications. Olen and Compton presented a systematic approach; a simple recursion is driven by the difference between the current synthesized pattern and the desired pattern over sidelobe regions [4]. The artificial interferers of various power levels are assigned in sidelobe regions to control sidelobe levels of the synthesized pattern. This algorithm is very effective and generally yields satisfactory array patterns. However, there is no pattern control mechanism in mainlobe region. In many applications, a mainlobe with a particular shape is desired, e.g., flat top. Without effective pattern control in the mainlobe region, specified pattern shaping can hardly be achieved.

The mainlobe shaping problem might be fixed by applying constraints. However, when more constraints are used, the sidelobes become more difficult to control because the degrees of freedom are reduced due to the constraints. Furthermore, an additional matrix inverse is required when using constraints, which adds computational complexity and makes the algorithm slow if many patterns must be synthesized, for example, when pattern requirements change with look direction and multiple look directions are anticipated [12].

In this paper, we present a new pattern synthesis algorithm for arbitrary arrays based on adaptive array theory. The problems mentioned above can be eliminated completely with the new algorithm. The new algorithm shapes the mainlobe with an iterative procedure and employs an efficient sidelobe peak iteration technique to reduce computational complexity.

We formulate the problem as finding the optimal array weight vector that minimizes the weighted L_2 norm of the difference between the synthesized pattern and the desired pattern. The difference between our algorithm and others that use the L_2 norm is that our algorithm iterates the values of the weighting function in order to minimize the exceedance the desired sidelobe levels and to minimize the absolute difference between desired and achieved patterns in the mainlobe region.

This algorithm offers a high flexibility and can easily handle arbitrary arrays with nonisotropic elements. The new algorithm offers a significant improvement to Olen and Compton's method [4] in regard to mainlobe shape control. Simulation results are presented to illustrate the effectiveness of this
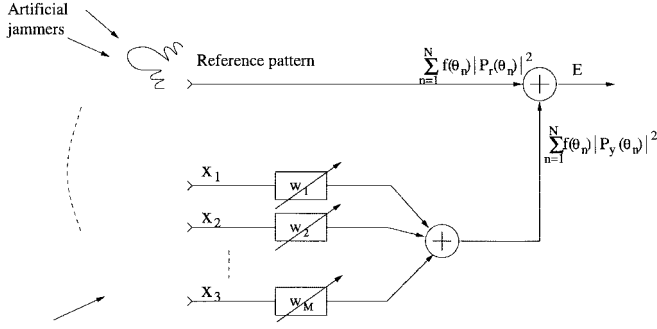
Fig. 1. An sidelobe canceler interpretation.

method. Following is the organization of this paper. Section II is the problem formulation, Section III describes the pattern synthesis algorithm, Section IV shows the simulation examples, and Section V is the conclusion.

II. THE PROBLEM FORMULATION

The problem of array pattern synthesis can be stated as follows. Given the number of array elements and the element positions, we want to find a set of complex weights w_i such that the array pattern $P_y(\theta)$ has a maximum at the desired direction θ_d with a certain beamwidth and also the sidelobe levels meet the specified values. Let's consider the sum of a weighted pattern errors E over the set of angles $\theta_1, \theta_2, \cdots, \theta_N$,

$$E = \sum_{i=1}^{N} f(\theta_i)|P_y(\theta_i) - P_r(\theta_i)|^2 \qquad (1)$$

where

$$P_y(\theta) = \mathbf{V}_{\mathbf{s}}'(\theta)\mathbf{W} \qquad (2)$$

and $\mathbf{V}_{\mathbf{s}}(\theta_i) = [g_1(\theta_i)e^{j\phi_1(\theta_i)} \quad g_2(\theta_i)e^{j\phi_2(\theta_i)} \quad \cdots \quad g_M(\theta_i)e^{j\phi_M(\theta_i)}]'$ is the steering vector of the array, $'$ is the conjugate transpose operation, $g_i(\theta)$ is the ith element pattern, $P_r(\theta_i)$ is the reference pattern, $f(\theta_i)$ is the weighting function, $\phi_i(\theta) = \mathbf{k} \cdot \mathbf{x}_i$ is the phase due to propagation where $\mathbf{k}$ is the wavenumber vector and $\mathbf{x}_i$ is the ith element position, and $\mathbf{W} = [w_1, w_2, \cdots, w_M]^T$ is the weight vector. When the error E is expressed as

$$E = \sum_{i=1}^{N} f(\theta_i)|\mathbf{V}'(\theta_i)\mathbf{W} - P_r(\theta_i)|^2 \qquad (3)$$

we observe that E may be interpreted as the average output power of a "sidelobe canceller" with main channel response $P_r(\theta)$ to a collection of jammers (Fig. 1), where the ith jammer has the location θ_i and the power $f(\theta_i)$. The key to this algorithm is that the jammer powers are adjusted to emphasize selected parts of the achieved pattern, particularly the mainlobe and sidelobe peaks.

The weight vector that minimizes the error E is the solution of a well-known least squares problem

$$\mathbf{W_{opt}} = \mathbf{R_s}^{-1}\mathbf{R_d} \qquad (4)$$
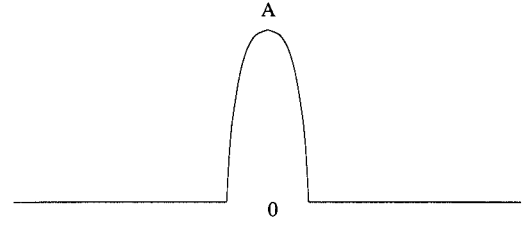
Fig. 2. The reference pattern $P_r(\theta)$.

where $\mathbf{R_s}$ is the covariance matrix and $\mathbf{R_d}$ is the cross-correlation vector defined as

$$\mathbf{R_s} = \sum_{n=1}^{N} f(\theta_n)\mathbf{V_s}(\theta_n)\mathbf{V_s}'(\theta_n) \qquad (5)$$

$$\mathbf{R_d} = \sum_{n=1}^{N} f(\theta_n)P_r(\theta_n)\mathbf{V_s}(\theta_n). \qquad (6)$$

Furthermore, when constraints are needed, the pattern synthesis problem can be formulated as

$$\min_{\mathbf{W}} \sum_{i=1}^{N} f(\theta_i)|\mathbf{V}'(\theta_i)\mathbf{W} - P_r(\theta_i)|^2, \qquad \text{subject to } \mathbf{CW} = \mathbf{h}$$

where $\mathbf{C}$ is the constraint matrix and $\mathbf{h}$ is the constraint vector. The solution for optimal weight vector is

$$\mathbf{W_{opt}} = \mathbf{R_s}^{-1}\mathbf{R_d} + \mathbf{R_s}^{-1}\mathbf{C}'(\mathbf{CR_s}^{-1}\mathbf{C}')^{-1}(\mathbf{h} - \mathbf{CR_s}^{-1}\mathbf{R_d}). \qquad (7)$$

The error E is found to be [10]

$$E = E_{\min} + (\mathbf{W} - \mathbf{W_{opt}})'\mathbf{R_s}(\mathbf{W} - \mathbf{W_{opt}})$$

where $E_{\min}$ is the minimized error E

$$E_{\min} = \sum_{i=1}^{N} f(\theta_i)|P_r(\theta_i)|^2 - \mathbf{W}_{\mathbf{opt}}'\mathbf{R_s}\mathbf{W_{opt}}.$$

The array response at each angular location depends on the weighting function $f(\theta_n)$. Different values of $f(\theta_n)$ put different emphasis on array responses at pertinent directions and, therefore, would result in a different array pattern. By making $f(\theta_n)$ large, our cost function makes it possible to ensure sidelobe peaks are below a certain value.

III. THE PATTERN-SYNTHESIS ALGORITHM

A. Full Iteration

The most common objective for pattern synthesis is to obtain a pattern with sidelobe level lower than a specified value over certain regions while maintaining a certain gain at look angle θ_d. Here, we select the reference pattern $P_r(\theta)$, as shown in Fig. 2, in which all the responses in sidelobe regions are zeros and the mainlobe peak response is a value A. The mainlobe shape is specified by the designer and could be, for example, a parabola. While it is impractical to have all zero sidelobe levels, we can induce lower and lower sidelobes by increasing the weighting function $f(\theta)$ in selected areas. We use a realistic desired pattern $P_d(\theta)$ to iteratively adjust $f(\theta)$ until the sidelobe requirements in $P_d(\theta)$ are met.

The weighting function is updated through an iteration procedure similar to that of Olen and Compton's [4], which leads to a satisfactory array pattern. The iteration is as in (8) and (9), shown on the bottom of the page, where $n = 1, 2, \cdots N$ indexes the points in angle over which we are interested in controlling the pattern. $f_k(\theta_n)$ and $P_{yk}(\theta_n)$ are the weighting function and the synthesized pattern, respectively, at the kth iteration and ϵ is a small number for an error tolerance between the synthesized pattern and the desired pattern in mainlobe region. $P_d(\theta_n)$ is the desired pattern; K_m and K_p are the iteration gains. Observe that for θ_n in the mainlobe region, $f_k(\theta_n)$ is never decreased from its initial value. The desired pattern $P_d(\theta)$ is set up to facilitate the iteration process whereas the reference pattern $P_r(\theta)$ is used to define the pattern errors that are to be minimized. In general, $P_d(\theta)$ and $P_r(\theta)$ are the same in mainlobe regions but different in sidelobe regions. The sidelobe part of $P_d(\theta_n)$ should be chosen according to a realistic specification or a reasonable estimation.

We next use $f_{k+1}(\theta_n)$ to compute new weights. Let N_L and N_R be the boundary points for mainlobe region, i.e. $N_L \le \theta_n \le N_R$ defines the mainlobe. Since the reference pattern is zero outside of this region, the cross-correlation vector and the covariance matrix become

$$\mathbf{R_d}(k+1) = \sum_{n=N_L}^{N_R} f_{k+1}(\theta_n) P_r(\theta_n) \mathbf{V'_s}(\theta_n) \tag{10}$$

$$\mathbf{R_s}(k+1) = \sigma^2 \mathbf{I} + \sum_{n=1}^{N} f_{k+1}(\theta_n) \mathbf{V_s}(\theta_n) \mathbf{V'_s}(\theta_n) \tag{11}$$

where a small quantity σ^2 is added to each diagonal element of the covariance matrix to prevent it from being ill conditioned [3]. Then the next weight vector is

$$\mathbf{W}(k+1) = \mathbf{R_s^{-1}}(k+1) \mathbf{R_d}(k+1). \tag{12}$$

The iteration stops when the errors between $P_{yk}(\theta)$ and $P_d(\theta)$ are small enough in the mainlobe region and the sidelobe levels of $P_{yk}(\theta)$ are equal to or lower than $P_d(\theta)$.

The procedure here is different from Olen and Compton's method [4] in three respects: 1) the summation for $\mathbf{R_s}(k)$ here includes both mainlobe and sidelobe regions whereas only sidelobe regions are included for $\mathbf{R_s}(k)$ in [4]; 2) the cross-corelation vector $\mathbf{R_d}(k)$ here includes all or selected steering vectors in the mainlobe region whereas only a single steering vector in the look direction is included in [4]; and 3) the iteration here occurs in both sidelobe and mainlobe regions whereas it only occurs in sidelobe regions in [4]. Mainlobe shaping is conveniently achieved by minimizing the output power (from the imaginary jammers) of the *difference* between two patterns (as in Fig. 1) rather than minimizing the output power of just the synthesized pattern.
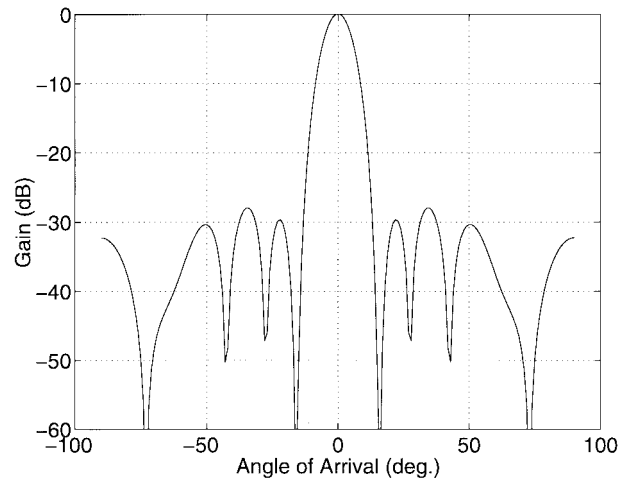

Fig. 3.   The initial pattern for the nonuniform linear array.

B. Peak-Only Iteration

In this section, we consider an alternative form of the algorithm that iterates the weighting function values only in the mainlobe and on the peaks of the sidelobes. Originally, we thought that this version would be faster, and indeed the amount of computation per iteration is significantly reduced. However, the number of iterations increases, so this version takes about the same amount of time overall as the version discussed in the previous section. In spite of the lack of time savings, we include this form because it is easier to program than the original form and because it may inspire a future algorithm that has the sought-after time savings.

We now motivate the alternative form. Suppose that the sidelobe part of the desired pattern $P_d(\theta_n)$ is some low but admissible sidelobe level[1]. Then, if the original version of the algorithm is allowed to continue after the maximum sidelobe specification is met, we find that all weighting function values outside of the mainlobe go to zero except those on the peaks of the sidelobes. This happens because the polarity on the $K_p[P_{yk}(\theta_n) - P_d(\theta_n)]$ term in (8) becomes negative for all θ_n in the sidelobes that don't correspond to peak locations. Eventually, the left argument of the max function becomes negative, and $f_{k+1}(\theta_n)$ becomes zero.

This phenomenon is illustrated for a 13-element nonuniform linear array. The selected desired pattern $P_d(\theta_n)$ has a sidelobe level of -30 dB. Figs. 3–8 show the synthesized patterns along with the weighting functions at different iteration steps. The synthesis process starts with unity values of weighting function. It is clear that the values of weighting function will only exist on the peaks in sidelobe regions as $k \to \infty$. This suggests an alternative iteration scheme in which sidelobe

[1] In [13], we show how to identify the lowest admissible sidelobe level.

$$f_{k+1}(\theta_n) = \begin{cases} h_k(\theta_n), & \theta_n \text{ in mainlobe region} \\ \max\{f_k(\theta_n) + K_p[P_{yk}(\theta_n) - P_d(\theta_n)], 0\}, & \theta_n \text{ in side lobe region} \end{cases} \tag{8}$$

$$h_k(\theta_n) = \begin{cases} f_k(\theta_n), & \text{if } |P_{yk}(\theta_n) - P_d(\theta_n)| \le \epsilon \\ f_k(\theta_n) + K_m|P_{yk}(\theta_n) - P_d(\theta_n)|, & \text{otherwise} \end{cases} \tag{9}$$
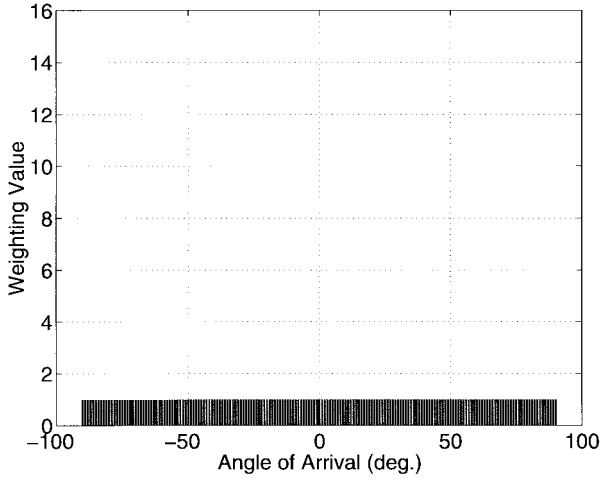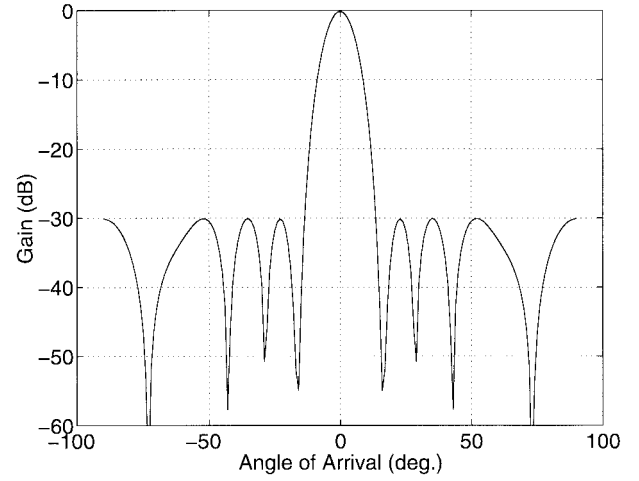
Fig. 4. The weighting function for the initial pattern.
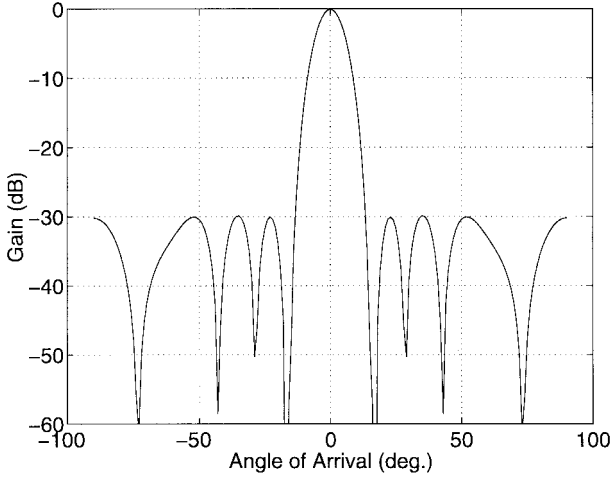

Fig. 5. The synthesized pattern at the 27th iteration.

Fig. 6. The weighting function at the 27th iteration.
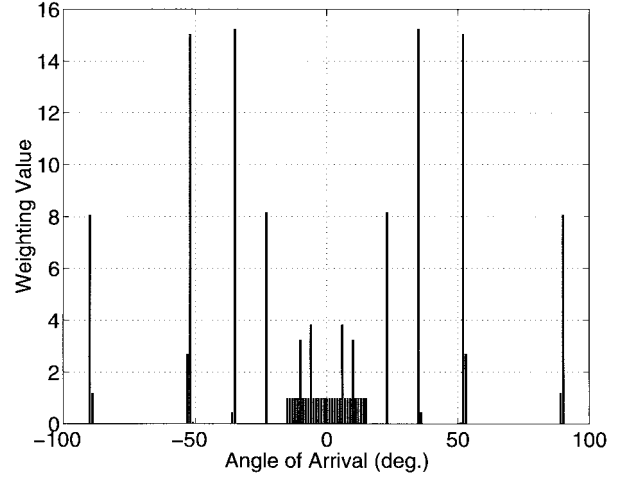

Fig. 7. The synthesized pattern at the 16 336th iteration.

Fig. 8. The weighting function at the 16 336th iteration

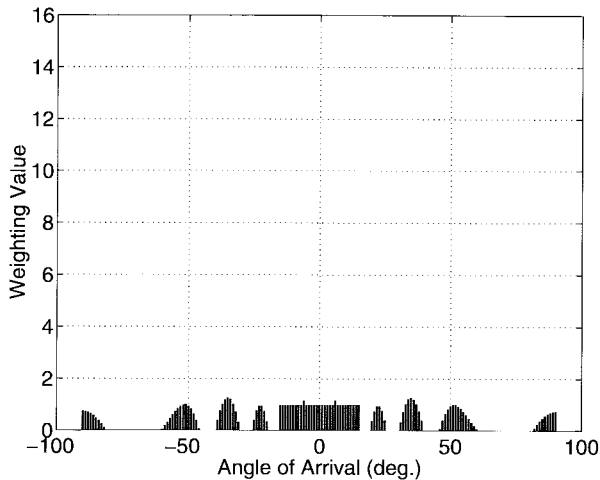The peak sidelobe locations will generally change from iteration to iteration. Therefore, the updates in (10) and (11) in Section III are no longer appropriate because they keep the same set of angles throughout the synthesis procedure. We can rewrite (8)–(12) in a form that allows the update terms to involve only the angles of sidelobe peaks. The covariance matrix and cross-correlation vector can be expressed in terms of a residual covariance matrix $\tilde{\mathbf{R}}_{\mathbf{s}}(k)$ and a residual cross-correlation vector $\tilde{\mathbf{R}}_{\mathbf{d}}(k)$, which are added to the current ones, i.e.,

$$\mathbf{R}_{\mathbf{s}}(k+1) = \mathbf{R}_{\mathbf{s}}(k) + \tilde{\mathbf{R}}_{\mathbf{s}}(k) \tag{13}$$

$$\mathbf{R}_{\mathbf{d}}(k+1) = \mathbf{R}_{\mathbf{d}}(k) + \tilde{\mathbf{R}}_{\mathbf{d}}(k) \tag{14}$$

where

$$\tilde{\mathbf{R}}_{\mathbf{s}}(k) = \sum_n r_k(n)\mathbf{V}_{\mathbf{s}}(\theta_n)\mathbf{V}'_{\mathbf{s}}(\theta_n) \tag{15}$$

$$\tilde{\mathbf{R}}_{\mathbf{d}}(k) = \sum_n r_k(n)P_r(\theta_n)\mathbf{V}'_{\mathbf{s}}(\theta_n). \tag{16}$$

Here, $r_k(n)$ is a residual weighting function that indicates how much correction is needed for the mainlobe and the sidelobe

levels can be sufficiently controlled by updating the weighting function only on peak sidelobe locations. To identify the locations of the sidelobe peaks, various peak-finding schemes are available. One of the easy ways to do it is to obtain peak locations by just comparing the neighboring response values.
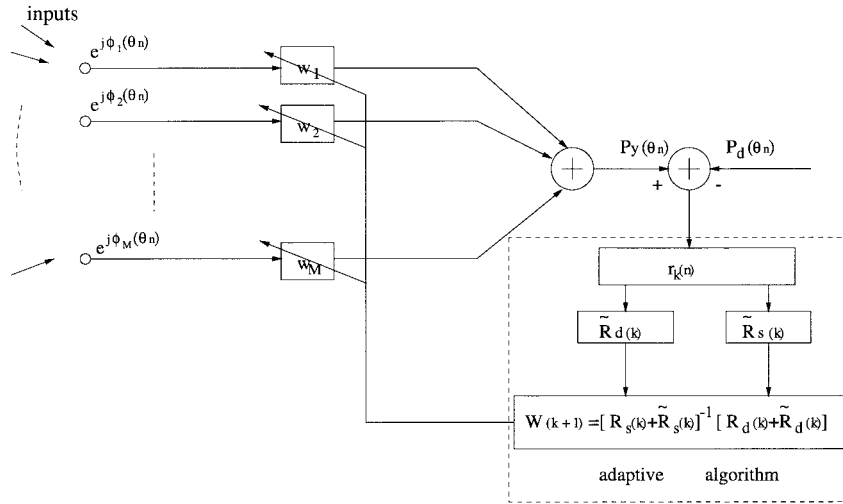
Fig. 9.   An adaptive spatial filter model.

peaks at the current iteration. $r_k(n)$ is calculated as follows:

$$
r_k(n) = \begin{cases} s_k(n) & \theta_n \text{ in mainlobe} \\ \max\{0, K_p[P_{yk}(\theta_n) - P_d(\theta_n)]\} & \theta_n \text{ in sidelobe peak} \end{cases}
$$
$$(17)$$

where

$$
s_k(n) = \begin{cases} 0, & \text{if } P_{yk}(\theta_n) - P_d(\theta_n)| \leq \epsilon \\ K_m|P_{yk}(\theta_n) - P_d(\theta_n)|, & \text{otherwise.} \end{cases}
$$
$$(18)$$

The next weight vector is

$$
\mathbf{W}(k+1) = [\mathbf{R_s}(k) + \tilde{\mathbf{R}}_\mathbf{s}(k)]^{-1}[\mathbf{R_d}(k) + \tilde{\mathbf{R}}_\mathbf{d}(k)]. \quad (19)
$$

With the exception of (17), the algorithm in (13)–(20) is equivalent to the previous one. The exception is that the update to the covariance matrix is done only for sidelobe peak locations.

The pattern synthesis with this algorithm can be modeled as an adaptive spatial filter shown in Fig. 9. The inputs to this filter are the signals of unit amplitude with incident angle θ_n. The signal from each array element is weighted and then summed to give the array output, which is compared with desired array response over an angular range. The errors between the array outputs and the desired pattern are used to update the covariance matrix and the cross-correlation vector to obtain the next optimal weight vector. The adaptive algorithm minimizes the errors so that the synthesized array pattern $P_y(\theta_n)$ approaches the desired pattern $P_d(\theta_n)$. This system is adaptive in the sense that the algorithm finds the next optimal weight vector depending on the new error environment.

This adaptive filter model suggests the following step-by-step flow chart for programming implementation.

1) Select inputs as N signals of unit amplitude incident from various angles; for example, from $-90°$ to $+90°$ with one-degree spacing.

2) Obtain an initial weight vector $\mathbf{W}(0)$, i.e.,

$$
\mathbf{R_s}(0) = \sigma^2\mathbf{I} + \sum_{n=1}^{N} \mathbf{V_s}(\theta_n)\mathbf{V_s'}(\theta_n) \tag{20}
$$

$$
\mathbf{R_d}(0) = \sum_{n} P_r(\theta_n)\mathbf{V_s'}(\theta_n) \quad \theta_n \text{ in mainlobe} \tag{21}
$$

$$
\mathbf{W}(0) = \mathbf{R_s}^{-1}(0)\mathbf{R_d}(0). \tag{22}
$$

Here, σ^2 is usually a small number; for example, 0.001. Then obtain an initial array pattern $P_y(\theta)$ from (2).

3) Find all peak sidelobe locations. Then obtain peak sidelobe values of $P_y(\theta)$ as well as some values of $P_y(\theta)$ in the mainlobe region at selected points of interest.

4) At selected points in mainlobe region and the points of sidelobe peaks, calculate the differences of $P_y(\theta_n) - P_d(\theta_n)$.

5) Calculate $r_k(n)$ from the results of step (4) for pertinent points using (17) and (18). Usually, K_m is specified to be much smaller than K_p, for example, $K_m = 0.45$ and $K_p = 45$. Then, calculate $\tilde{\mathbf{R}}_\mathbf{s}(k)$ and $\tilde{\mathbf{R}}_\mathbf{d}(k)$ using (15) and (16).

6) Add $\tilde{\mathbf{R}}_\mathbf{s}(k)$ to the current covariance matrix $\mathbf{R_s}(k)$, and also $\tilde{\mathbf{R}}_\mathbf{d}(k)$ to the current cross-correlation vector $\mathbf{R_d}(k)$. Then obtain the next weight vector $\mathbf{W}(k+1)$ from (19).

7) Obtain the array pattern $P_y(\theta)$ from (2). If it is satisfactory, stop; otherwise, go to step (3).

The peak-only iteration scheme provides an alternative to the full iteration scheme. We found that peak-only iteration took less computational time for each iteration because the number of processing units involved in each iteration was greatly reduced. However, each iteration did not change the pattern as much compared to that of full iteration, therefore, more iterations were needed to obtain the same pattern. As a result, the overall computational times for both iteration schemes were about the same.
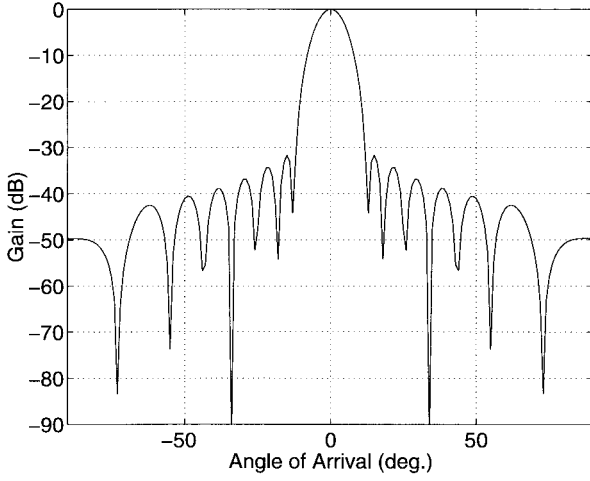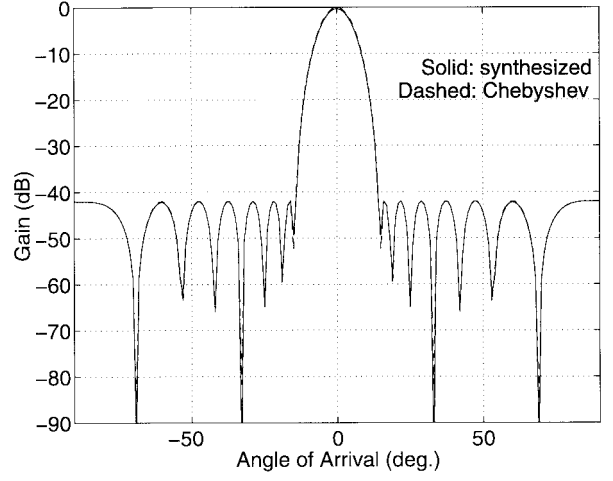
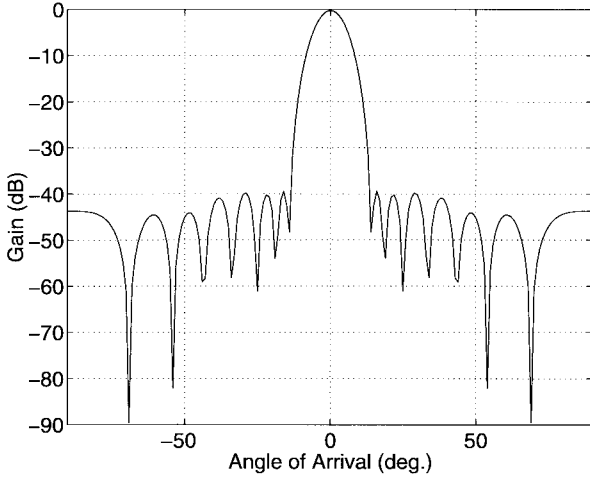Fig. 10.   Initial pattern.

Fig. 11.   Intermediate pattern.

C. Two-Dimensional Pattern Synthesis

This algorithm can be extended in a straightforward way to the synthesis of two-dimensional (2-D) array patterns. In the 2-D peak-only iteration scheme, the residual weighting function at kth iteration for the spatial location (ϕ_m, θ_n) becomes $r_k(m, n)$, where ϕ_m is an azimuth angle and θ_n is an elevation angle. θ_n is replaced with (ϕ_m, θ_n) in all previous expressions.

IV.   SIMULATION EXAMPLES

In this section, we will show a few pattern synthesis examples using our algorithm. The array elements used in following examples are assumed to be isotropic although such an assumption is not necessary in our algorithm. No mutual coupling was assumed. Also, the peak-only iteration scheme is used in the examples.

The first example is a Chebyshev pattern synthesis for a 15-element uniform linear array with a half wavelength spacing. Fig. 10 shows the initial pattern, Fig. 11 shows an intermediate pattern and Fig. 12 is the final synthesized pattern along with the ideal Chebyshev pattern for the same array; the solid line is the synthesized pattern and the dashed line is the Chebyshev pattern. The patterns are almost identical. Here we selected

Fig. 12.   Synthesized and Chebyshev patterns.

TABLE I
SYNTHESIZED PATTERN FOR A 21-ELEMENT NONUNIFORM LINEAR ARRAY

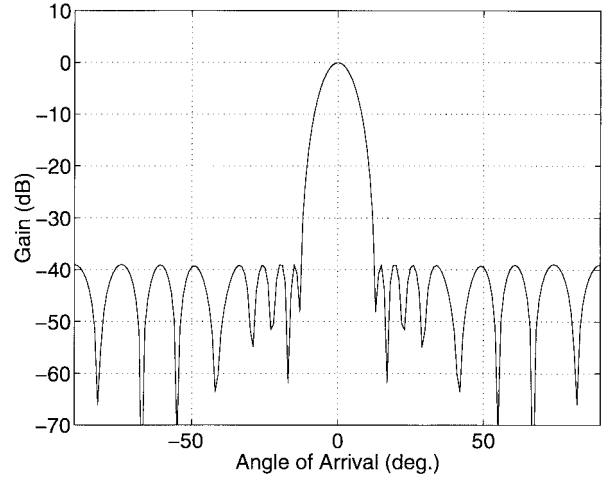| Element Nos.	Position	Element Nos.	Position
1, 21	$\pm 5.0000\lambda$	6, 16	$\pm 2.3497\lambda$
2, 20	$\pm 4.6065\lambda$	7, 15	$\pm 1.8494\lambda$
3, 19	$\pm 3.8098\lambda$	8, 14	$\pm 1.5302\lambda$
4, 18	$\pm 3.2995\lambda$	9, 13	$\pm 0.6299\lambda$
5, 17	$\pm 2.8973\lambda$	10, 12	$\pm 0.3749\lambda$
11	0λ		

Fig. 13.   The synthesized pattern.

$K_p = 36$ and $K_m = 3000$. The reference pattern is chosen as the same as that in Fig. 2 except that there is only one specified value unity at the desired signal direction in the mainlobe region. The final synthesized pattern was realized in 23 iterations.

The second example is to synthesize a pattern for a 21-element nonuniform linear array with element positions shown in Table I. Fig. 13 shows the result using our method with $P_r(\theta) = \cos^2(7\theta)$ in the mainlobe.

In the third example, the synthesized pattern for a 41-element nonuniform linear array has a flat top and a notch in the sidelobe region as shown in Fig. 14. Constraints can also be incorporated to place nulls in sidelobe regions by
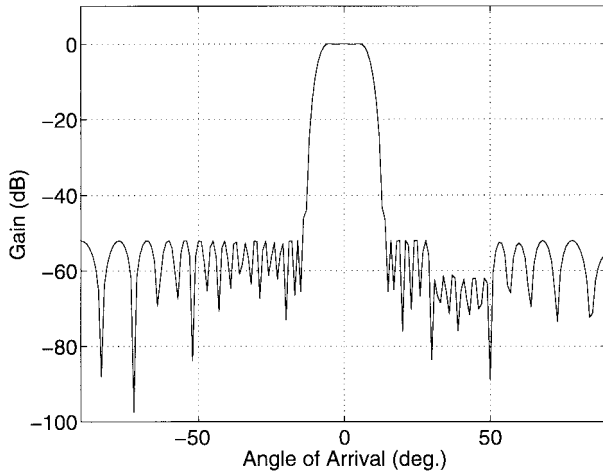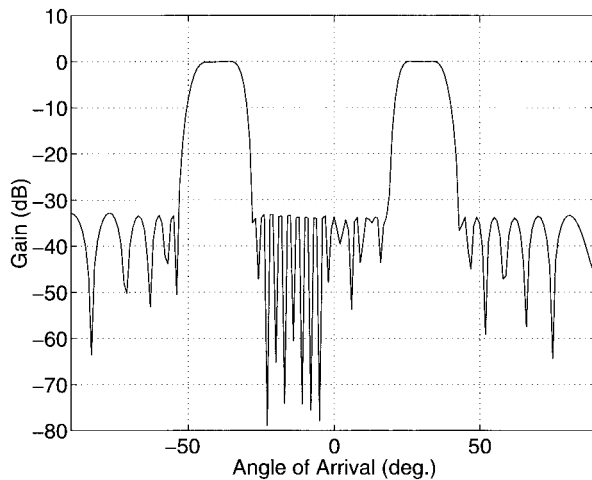
Fig. 14.   Flat-top pattern with notch.

Fig. 16.   Chebyshev/synthesized patterns.

Fig. 15.   Multibeam pattern.

Fig. 17.   A nonuniform array.

using (7) instead of (4). The fourth example is a multibeam pattern synthesis for the same array as the third example. The synthesized pattern is shown in Fig. 15. In above two cases, Olen amd Compton's method is not applicable because it does not provide any pattern control in the mainlobe region.

The fifth example is to synthesize a 2-D Chebyshev pattern for a 5×5 rectangular uniform planar array of 25 elements with half-wavelength spacing. The ideal Chebyshev pattern, shown as the solid line in Fig. 16 is used as both the reference pattern and the desired pattern. We used $1°$ spacing in azimuth from $0°$ to $180°$ and also $1°$ spacing in elevation from $-90°$ to $90°$ for placing values of the weighting function. Fig. 16 also shows the lowest upper bound (dashed) and highest lower bound (dash) for all radial cuts of the synthesized pattern after 18 iterations. It is observed that the sidelobe peaks deviate from the ideal no more than 1.5 dB.

The sixth example is to synthesize a 2-D pattern for a nonuniform planar array of 61 elements shown in Fig. 17. The reference pattern is a circularly rotated version of Fig. 2. The desired pattern is basically the same as the reference pattern except the sidelobe value is set to be a constant of 0.06 (-24.437 dB). Also, we used $K_p = 30$ and $K_m = 300$.
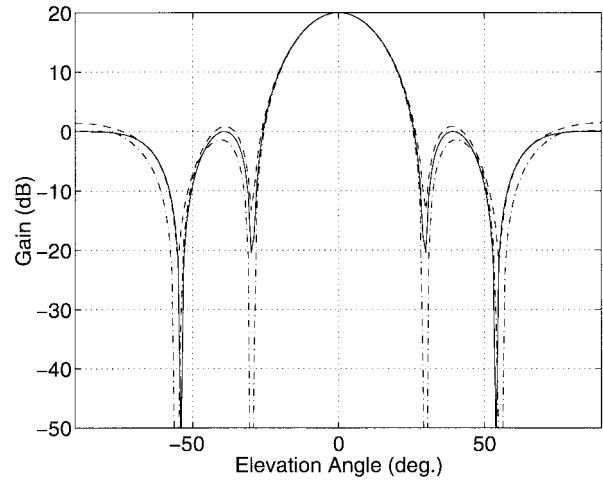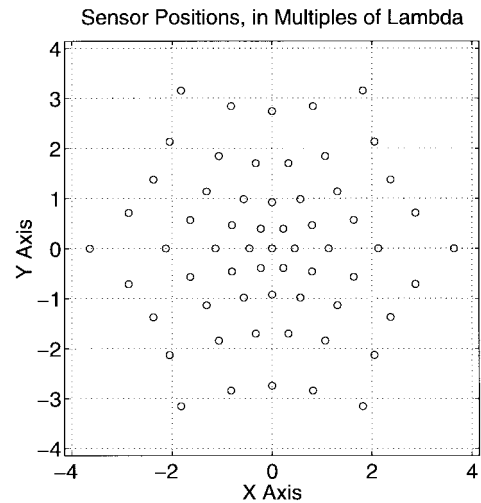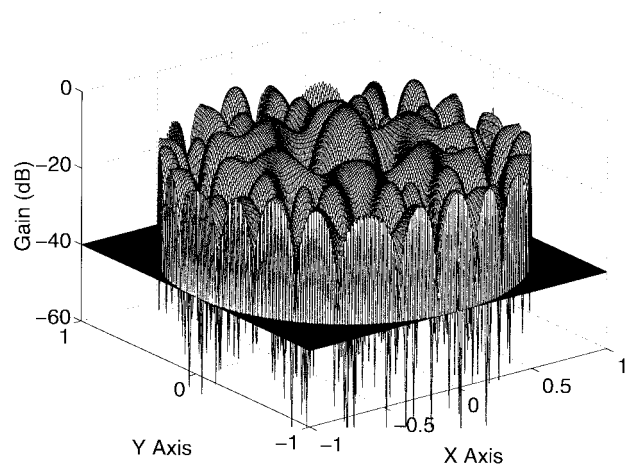

Fig. 18.   Initial pattern.

The initial 2-D pattern is plotted in Fig. 18 as a function of $x = \sin\theta\cos\phi$ and $y = \sin\theta\sin\phi$ and a side view of the initial pattern is plotted in Fig. 19. The Figs. 20 and 21 show the two views of the final synthesized pattern.
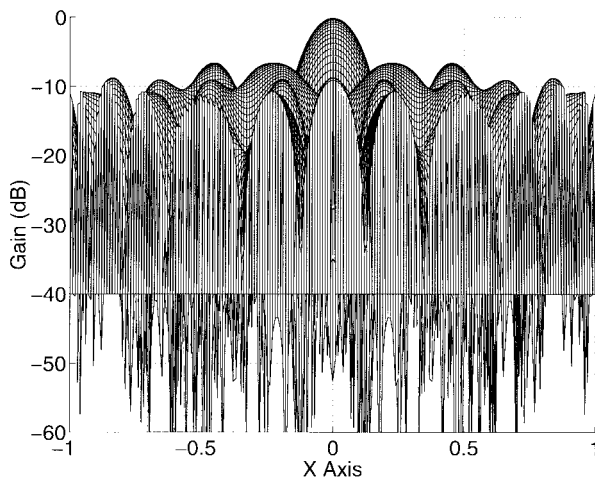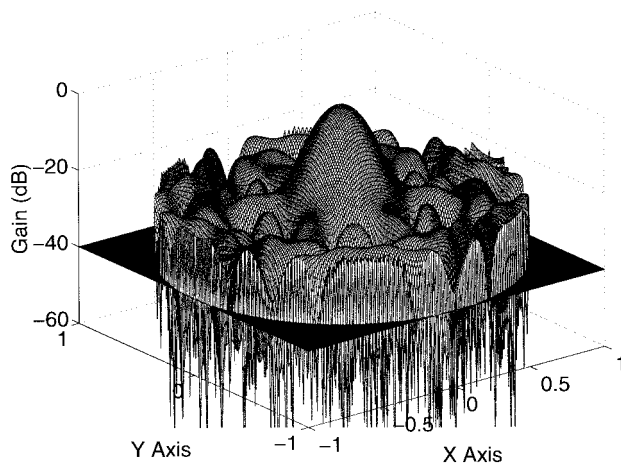
Fig. 19. A side view.
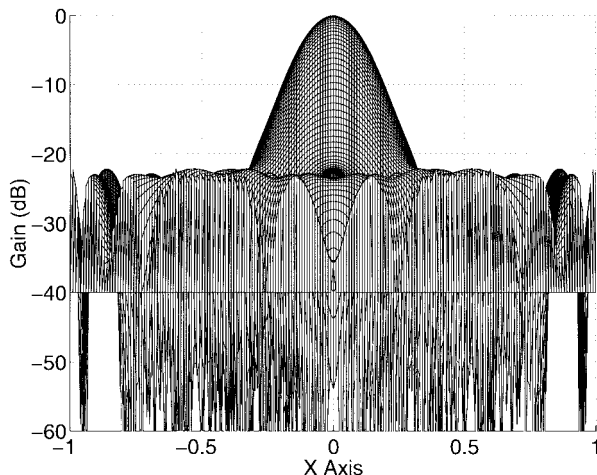

Fig. 20. Synthesized pattern.

Fig. 21. A side view.

V. CONCLUSION

We have presented two new pattern synthesis algorithms for arbitrary arrays, where iterations occur only in the mainlobe and on sidelobe peaks. The optimal weight vector is obtained by minimizing the sum of weighted squared errors between synthesized and desired patterns. Iterations on the weighting function in both mainlobe and sidelobe regions insure a desired mainlobe shape as well as desired sidelobe levels. The weighting function iterations are driven by the difference between the synthesized pattern and desired pattern. In contrast to Olen and Compton's method, the new method provides convenient mainlobe shape control. The algorithm has been demonstrated with simulations of nonuniform linear and planar arrays.

REFERENCES

[1] C. L. Dolph, "A current distribution for broadside arrays which optimizes the relationship between beam width and sidelobe level," in *Proc. IRE*, vol. 34, pp. 335–348, June 1946.
[2] R. J. Mailloux, *Phased Array Antenna Handbook*. Norwood, MA: Artech House, 1994
[3] C.-Y. Tseng and L. J. Griffiths, "A simple algorithm to achieve desired patterns for arbitrary arrays," *IEEE Trans. Signal Processing*, vol. 40, pp. 2737–2746, Nov. 1992.
[4] C. A. Olen and R. T. Compton Jr., "A numerical pattern synthesis algorithm for arrays," *IEEE Trans. Antennas Propagat.*, vol. 38, pp. 1666–1676, Oct. 1990.
[5] B. P. Ng, M. H. Er, and C. Kot, "A flexible array synthesis method using quadratic programming," *IEEE Trans. Antennas Propagat.*, vol. 41, pp. 1541–1550, Nov. 1993.
[6] J. Perini, "Note on antenna pattern synthesis using numerical iterative methods," *IEEE Trans. Antennas Propagat.*, vol. 19, pp. 284–286, Mar. 1971.
[7] J. C. Sureau and K. J. Keeping, "Sidelobe control in cylindrical arrays," *IEEE Trans. Antennas Propagat.*, vol. AP-30, pp. 1027–1031, Sept. 1982.
[8] L. Wu and A. Zielinski, "An iterative method for array pattern synthesis," *IEEE J. Ocean. Eng.*, vol. 18, pp. 280–286, July 1993.
[9] D. H. Johnson and D. E. Dudgeon, *Array Signal Processing: Concepts and Techniques*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[10] R. T. Compton Jr., *Adaptive Antennas: Concepts and Performance*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[11] R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*. New York: Wiley, 1980.
[12] P. D. Anderson, M. A. Ingram, and P. Zhou, "Array pattern synthesis for satellite communications on-the-move," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Monterey, CA, Nov. 1996, pp. 50–54.
[13] P. Y. Zhou, M. A. Ingram, and P. D. Anderson, "Synthesis of minimax sidelobes for arbitrary arrays," *IEEE Trans. Antennas Propagat.*, vol. 46, pp. 1759–1760, Nov. 1998.

Philip Yuanping Zhou received the B.S. degree in electrical engineering from Chongqing University, China, in 1982, and the M.S. degree in electrical engineering and computer science from the University of Illinois at Chicago in 1986. He is currently working toward the Ph.D. degree at the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta.

From 1989 to 1994, he worked as an Engineer with MERET Optical Communications, Inc., Santa Monica, CA. His research interests include adaptive array processing, array pattern synthesis, and wireless communications.

Mary Ann Ingram received the B.E.E. and Ph.D. E.E. degrees from the Georgia Institute of Technology, Atlanta, in 1983 and 1989, respectively.

From 1983 to 1986, she was a Research Engineer with the Georgia Tech Research Institute, doing signals and systems analyses of advanced radar systems and radar electronic countermeasure (ECM) systems. From 1986 to 1989 she was a Research Assistant with the School of Electrical Engineering, Georgia Institute of Technology, doing modeling and estimation of linear systems that are disturbed by random point processes. In 1989 she joined the faculty of the School of Electrical and Computer Engineering, Georgia Institute of Technology, where she is now an Associate Professor. Her research has included topics in wireless communications, array pattern synthesis, array signal processing, optical transmission systems, and radar systems.