*Progress In Electromagnetics Research B, Vol. 49, 281–300, 2013*

# PENALTY FUNCTION SOLUTION TO PATTERN SYNTHESIS OF ANTENNA ARRAY BY A DESCENT ALGORITHM

**Tiaojun Zeng and Quanyuan Feng***

School of Information Science and Technology, Southwest Jiaotong University, Sichuan 610031, China

**Abstract**—In this paper, an algorithm based on penalty cost function for synthesizing flat-top patterns is proposed. A descent algorithm (DA) as its optimizing approach is proposed in this paper as well. Apparently, whole algorithm efficiency totally depends on the DA. Unlike traditional descent method, the DA defines step length by solving a inequality, instead of Wolf or Armijo-type search rule, stimulation results indicate that it can improve the computational efficiency. Under mild conditions, we prove that the DA has strong convergence properties. Several numerical examples are presented to illustrate the effectiveness of the proposed algorithm. The results indicate that the approach is effective in the pattern shape precisely in both mainlobe and sidelobe region for arbitrary linear arrays.

## 1. INTRODUCTION

Antenna array pattern synthesis (AAPS), the fundamental problems in array signal processing, is to find the complex excitation or physical layout of the array that produces the radiation pattern as close as possible to the desired one. It becomes a hot topic of research over the last several decades; thereof, many algorithms have been developed in the area [1–9]. Classic techniques, such as the Dolph-Chebyshev and Taylor methods, have many practical difficulties in the array design especially if there are some restricted conditions. Moreover, synthesis of antenna arrays also imposes tough challenges that mainly come from the nonlinear and non-convex dependency of the array factor to

*Received 28 December 2012, Accepted 28 January 2013, Scheduled 26 February 2013*
* Corresponding author: Quanyuan Feng (fengquanyuan@163.com).

positions, excitation phases and amplitude of elements [3]. In recent years, Evolutionary algorithms, as one important kind of algorithms, such as differential evolution (DE) [1, 2], particle swarm optimization (PSO) [3–5] and genetic algorithm (GA) [6–8], have been also used for antenna array optimization, but these global optimization techniques are in fact limited in their performances from the fact that their computational time is huge.

In general, numerical approaches are more practical for AAPS problem. Penalty function method is an important tool in solving nonlinear optimization problems. Its strategy is to convert a constrained mathematical problem into unconstrained optimizing form via a penalty parameter that penalizes any violation of the constraints. In this paper, we propose an algorithm to AAPS problem based on numerical optimization. First, convert the AAPS problem into a unconstrained cost function, then minimize the function to obtain optimal array weight vector by the DA.

The rest of the paper is organized as follows. Section 2 presents the cost function formulation. In Section 3, DA is first introduced and then proven as a well-defined algorithm with its convergence properties. Section 4 shows the experimental results and related discussions. Conclusions are drawn in Section 5.

## 2. COST FUNCTION FORMULATION

Consider a narrow band linear array with $N$ isotropic elements (actually, our method can be extended to planar array situation as well). Assume that a signal arrives at an angle of $\theta$, so the array far-field response can be represented as

$$p(W,\theta) = \sum_{i=1}^{N} w_i e^{j\phi_i(\theta)} = W^T S(\theta), \quad \theta \in [-90^\circ, 90^\circ] \tag{1}$$

where $j = \sqrt{-1}$, $S(\theta) = [1,\ e^{j\varphi_2(\theta)}\ \ldots\ e^{j\varphi_2(\theta)}]^T$ is the steering vector, $\phi_i(\theta) = 2\pi d_i \sin\theta/\lambda$ the phase delay due to propagation, $\lambda$ the wavelength of the transmitted signal, $d_i$ the position of the $i$th element of the antenna array ($d_1 = 0$), and $W = [w_1,\ w_2,\ \ldots\ w_N]^T$ the complex-weight vector. $(\cdot)^*$ is the conjugate and $(\cdot)^H$ the conjugate transpose. Considering the optimization approach is only applicable to real variable problems, a complex-to-real transform of the response $p(W,\theta)$ is necessary. Firstly, introduce a power function ($P(W,\theta)$) of the array respond, i.e., $P(W,\theta) = |p(W,\theta)|^2 = p(W,\theta)p^*(W,\theta)$, then denote $W = W_1 + jW_2$, $W_1 = [w_1^1\ \ w_2^1\ \ \ldots\ \ w_N^1]^T$, $W_2 = [w_1^2\ \ w_2^2\ \ \ldots\ \ w_N^2]^T$, $S(\theta) =$

$S_1(\theta) + jS_2(\theta)$, $S_1(\theta) = [1\ \cos(\varphi_2(\theta))\ \ldots\ \cos(\varphi_N(\theta))]^T$, $S_2(\theta) = [0\ \sin(\varphi_2(\theta))\ \ldots\ \sin(\varphi_N(\theta))]^T$, $W_1$, $W_2$, $S_1(\theta)$ and $S_2(\theta) \in \mathbb{R}^{\mathbb{N}}$. Thus,

$$P(W,\theta) = \Big[\left(W_1^T S_1\right)^2 + 2\left(W_1^T S_1\right)\left(W_2^T S_2\right) + \left(W_2^T S_2\right)^2 + \left(W_1^T S_2\right)^2 - 2\left(W_1^T S_2\right)\left(W_2^T S_1\right) + \left(W_2^T S_1\right)^2\Big] \in \mathbb{R} \quad (2)$$

In the case where phase constraint is not considered, an optimal weight vector, $W_{opt}$, is determined so that the pown (or amplitude) response $P(W,\theta)$ (or $|p(W,\theta)|$) best approximates the desired pattern. The definition of flat-top synthesis is illustrated in Figure 1. In [9], a nonlinear least square (NLS) method for synthesizing flat-top pattern was proposed, and the normalized synthesis problem was formulated by a cost function based on the least-square error criterion as

$$J(W) = \sum_{\theta\in[-90,\,\theta_1]} \eta(|p| - \alpha)^2 + \sum_{\theta\in[\theta_2,\theta_3]} (|p| - 1)^2 + \sum_{\theta\in[\theta_4,90]} \eta(|p| - \alpha)^2 \quad (3)$$

where $p = p(W,\theta)$, $0 \leqslant \alpha \ll 1$, and $\eta$ ($\eta \geqslant 1$) is a penalty of factor for weighting sidelobe errors. Nevertheless, there exist two main drawbacks in the formulation: (1) In the process of optimizing the cost function, Equation (3) punishes all iterative solutions in the same way regardless of desired or undesired one. For example, if there are two solutions $W_1$ and $W_1$ such that:
(a) $|p(W_1,\theta)| < \alpha$, $|p(W_2,\theta)| > \alpha$ and $(\alpha - |p(W_1,\theta)|) \approx (|p(W_2,\theta)| - \alpha)$, $\forall\theta \in [-90,\ \theta_1] + [\theta_4,\ 90]$.
(b) $|p(W_1,\theta)| \approx |p(W_2,\theta)| \approx 1$, $\forall\theta \in [\theta_2,\ \theta_3]$.
Thereof, $J(W_1) \approx J(W_2)$. But, in fact, vector $W_1$ is a optimal solution of the problem and $W_2$ the undesired one, so that is unreasonably very much. In other words, the peak sidelobe level (PSLL) of the antenna array is not always below threshold $\alpha$ even if its the peak main lobe level (PMLL) converges to 1, and its cost function converges to a very small positive number, because the cost function has not any penalty terms to discriminate solution's forms.

(2) The formulation does not consider the transition region situation, and the threshold $\alpha$ is a given value and constant in the whole optimization. They are the hindrances to the cost function converging to optimal solution. Based on above analysis, we propose a method for synthesizing flat-top pattern of antenna array, which is formulated by a discrete angle penalty function based on pown response of antenna

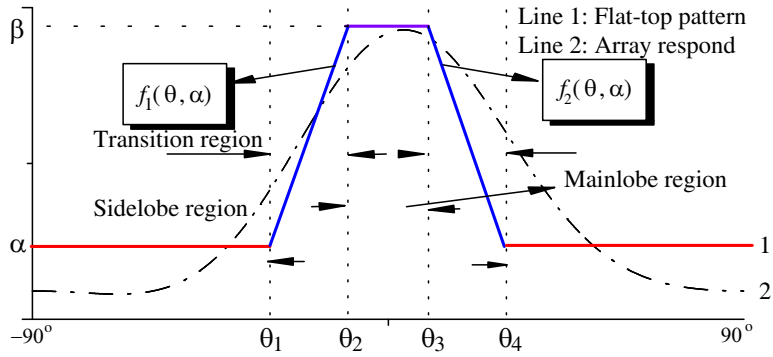


**Figure 1.** Definition of flat-top synthesis.

array as following:

$$
\begin{aligned}
& F(W,\alpha) \\
= & \sum_{i=0,\theta=-90+i\cdot\Delta\theta}^{i=N_1,\theta=\theta_1}\left[(\alpha-P)^2+\left(\frac{P}{\alpha}\right)^m\right]+\sum_{i=1,\theta=\theta_1+i\cdot\Delta\theta}^{i=N_2,\theta=\theta_2}\left[(f_1-P)^2+\left(\frac{P}{f_1}\right)^m\right] \\
& +\sum_{i=1,\theta=\theta_2+i\cdot\Delta\theta}^{i=N_3,\theta=\theta_3}(\beta-P)^2+\sum_{i=1,\theta=\theta_3+i\cdot\Delta\theta}^{i=N_4,\theta=\theta_4}\left[(f_2-P)^2+\left(\frac{P}{f_2}\right)^m\right] \\
& +\sum_{i=1,\theta=\theta_4+i\cdot\Delta\theta}^{i=N_5,\theta=90}\left[(\alpha-P)^2+\left(\frac{P}{\alpha}\right)^m\right]-\lambda\cdot\lg|\alpha| \qquad (4)
\end{aligned}
$$

where $\Delta\theta$ is the angle resolution, $P = P(W,\theta)$ defined by Equation (2), constant $m > 0$, and $\beta = r\cdot\alpha$ ($r \geqslant 100$ is a given value), $f_1 = f_1(\theta,\alpha) = \frac{(k-1)\alpha}{(\theta_2-\theta_1)}(\theta-\theta_1)+\alpha$, $f_2 = f_2(\theta,\alpha) = \frac{(k-1)\alpha}{(\theta_3-\theta_4)}(\theta-\theta_4)+\alpha$. For avoiding threshold $\alpha$ converging to zero, a penalty term $-\lambda\cdot\lg|\alpha|$ is incorporated into Equation (4), where $\lambda$ is a very small positive number. If $\alpha$ gradually converge to zero solutions, it leads to $-\lambda\cdot\lg|\alpha| \to +\infty$, because $\alpha$ is a normal value and because its penalty term is a very small positive number. As the cost function is minimized, threshold $\alpha$ never converges to zero solutions, and tends to gradually move toward a optimal solution. Similarly, penalty term $(\frac{P}{\alpha})^m$ can prevent iterative algorithm from converging these solutions which make $P(W,\theta)$ bigger than $\alpha$. In the transition region, array response does not exceed the given pattern under the penalty terms $(\frac{P}{f_1})^m$ and $(\frac{P}{f_2})^m$ limiting. When

$F(W,\theta) \to \varepsilon$ ($\varepsilon$ is a small number), we obtain

$$\forall\theta \in [-90,\theta_1]+[\theta_4,90]\,,\ \left[(\alpha - P(W,\theta))^2 + \left(\frac{P(W,\theta)}{\alpha}\right)^m\right] \to \varepsilon$$

$$\forall\omega \in [\theta_2,\theta_3]\,,\ (\beta - P(W,\omega))^2 \to \varepsilon$$

i.e., $P(W,\theta) < \alpha$ and $P(W,\omega) = \beta$, thereof, $|\text{PSLL}| = |10 * \log(\frac{P(W,\omega)}{P(W,\theta)})| > |10 * \log(\frac{\beta}{\alpha})| > 20\,\text{dB}$ ($\because \beta > 100\alpha$).

We name the proposed approach as penalty function method (PFM). Above analysis indicates that weight vector $W$ converges to an optimal solution when $F \to \varepsilon$. In practice, the converging condition is too strict for application, because $F$ is a bundle of many sub functions, and moreover, generally every sub function has convergence errors. Thereof, it is more practical to use its average denoted as ACF (average cost function, i.e., $\frac{F}{(180/\Delta\theta)}$) to index algorithm convergency. The optimal value of the cost function h is finally obtained by the optimizing method. In this paper, an algorithm based on gradient iterative method is propped in the next section; however, for applying the approach, we must first compute its gradient and Hessian matrices (see the details in Appendix A).

## 3. NEW DESCENT ALGORITHM

Consider an unconstrained optimization problem

$$\min_{x\in\Re^n} f(x). \tag{5}$$

In the problem, $f : R^n \to R$ is a continuously differentiable function. In general, numerical methods solving the problem have the following iterative formulation [10]:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{6}$$

where $x_k$, $\alpha_k$, and $d_k$ are current iterative point, positive step length, and search direction, respectively. In this paper, we regard $d_k = -\nabla f(x_k)$ as the search direction at every iteration, which is extensively used in solving large-scale minimization problems [11]. Generally, step length $\alpha_k$ can be defined by the most popular line search techniques such as Armijo's rule, Goldstein's rule, and Wolfe's rule. These line search procedures require much time to compute the objective function's gradients $\nabla f(x_k+\alpha_k d_k)$. To improve computation efficiency, we propose a new rule that merely computes inequality to define step length. Most importantly, under the new rule, there is no need to compute $\nabla f(x_k + \alpha_k d_k)$.

Throughout the paper, $\{x_k\}$ denotes the sequence of points generated by our algorithm. DA iteratively solves the optimization problem. At each iteration, the search direction is $d_k = -g_k$ ($g_k = \nabla f(x_k)$). For computing the step length $\alpha_k$, firstly, a ratio is introduced as following

$$\rho_k = \frac{f(x_k) - f(x_k + \alpha_k d_k)}{f(x_k) - \varphi(x_k + \alpha_k d_k)} \tag{7}$$

where $f(x_k) - \varphi(x_k + \alpha_k d_k) = f(x_k) - (f(x_k) + g_k^T(\alpha_k d_k) + \frac{1}{2}(\alpha_k d_k)^T H_k(\alpha_k d_k))$ and $H_k = \nabla^2 f(x_k)$. It is clear that $\rho_k$ indicates the approximate degree of $f(x_k + \alpha_k d_k)$ and $\varphi_k(x_k + \alpha_k d_k)$. Moreover, $\rho_k \to 1$ whereas $\alpha_k \to 0$. In iterative computations, $\rho_k \geqslant c_0 (3.1)$ ($c_0$ is a given positive constant) means that $\varphi(x_k + \alpha_k d_k)$ is approximate to $f(x_k + \alpha_k d_k)$, and trial step length $\alpha_k$ is accepted. On the other hand, when $\alpha_k$ is not accepted ($\rho_k < c_0$), some methods are used to resolve the inequality by reducing trial step length until an acceptable length is obtained. The procedures are as follows:

**Algorithm 3.1.** Given the points $x_k$, $g_k$, $H_k$ and the constant $0 < c_0 < 1$. Let $d_k = -g_k$.

**Step 1.** If $g_k^T H_k g_k \geqslant 0$, then solve the inequality by $f(x_k) - \varphi(x_k + \alpha_k d_k) = \alpha_k \|g_k\|^2 - \frac{1}{2}\alpha_k^2 g_k^T H_k g_k > 0$, thereby obtaining $0 < \alpha_k < \frac{2\|g_k\|^2}{g_k^T H_k g}$. Initialize $\alpha_k = \frac{1.99\|g_k\|^2}{g_k^T H_k g}$, then go to step 2.
If $g_k^T H_k g_k < 0$, then initialize $\alpha_k = C$ ($C$ is a large positive number), go to step 2.

**Step 2.** If the inequality (3.1) holds, the computation is stopped, and output $\alpha_k$ is obtained. Otherwise, let $\alpha_k \leftarrow c_1 \alpha_k$ ($0 < c < 1$)and loop with step 2.
For the comparison utilizing Wolfe's rule, we first describe the parameter.

**Wolfe's rule** defines step length $\alpha_k$ such that [10]

$$\begin{aligned} &f(x_k + \alpha_k d_k) \leqslant f(x_k) + \sigma_1 \alpha_k g_k^T d_k \\ &g(x_k + \alpha_k d_k)^T d_k \geqslant \sigma_2 g_k^T d_k. \end{aligned} \tag{8}$$

Apparently, the inequality in the group must be resolved, and $g(x_k + \alpha_k d_k)$ must be computed under this rule, which wastes more time than our method. Likewise, step length $\alpha_k$ is determined directly by following inequality:

$$\begin{aligned} f(x_k) - f(x_k + \alpha_k d_k) = f(x_k) - (f(x_k) + g_k^T(\alpha_k d_k) \\ + \frac{1}{2}(\alpha_k d_k)^T H_k(\alpha_k d_k) + o\left(\|\alpha_k d_k\|^2\right) \geqslant 0. \end{aligned}$$

The method is certainly feasible and simple. Unfortunately, its step length $\alpha_k$ (direction vector $d_k$ is given here) is a smaller value because $f(x_k+\alpha_k d_k) = f(x_k) + g_k^T(\alpha_k d_k) + \frac{1}{2}(\alpha_k d_k)^T H_k(\alpha_k d_k) + o(\|\alpha_k d_k\|^2)$ holds only in the condition that $\|\alpha_k d_k\|$ is a very small value. Such a condition ultimately debases computational efficiency. In our method, we obtain a bigger step length by adjusting the positive constant $c_0$. Based on Algorithm 3.1, we calculate descent algorithm as follows:

**Algorithm 3.2.**

**Step 0:** Choose an initial point $x_0 \in R^n$. Set $d_0 = -g_0 = -\nabla f(x_0)$ and $k = 0$.

**Step 1:** If $\|g_k\| \leqslant \varepsilon$ ($\varepsilon$ is a very small positive number), then the calculation is stopped; otherwise, go to step 2.

**Step 2:** Set $d_k = -g_k = -\nabla f(x_k)$, then compute step length $\alpha_k$ by Algorithm 3.1 and let $x_{k+1} = x_k + \alpha_k d_k$.

**Step 3:** Let $k \leftarrow k+1$, and go back to step 1.

The following assumptions, commonly used in the convergence analysis of most optimization algorithms, are proposed to analyze the convergence properties of Algorithm 3.2.

**Assumption 1.** The objective function $f(x)$ is continuously differential and has a low bound on $R^n$.

**Assumption 2.** There exists a positive constant $M$ such that $\left\|\nabla^2 f(x)\right\| \leqslant M$, $\forall x \in \Omega$.

The following lemma shows that Algorithm 3.1 is well defined:

**Lemma 0.0.1.** *If Assumption 1 holds, then the Algorithm 3.1 is well defined.*

**Proof.** By Assumption 1, we obtain

$$\begin{aligned} f(x_k+\alpha_k d_k) &= f(x_k)+g_k^T(\alpha_k d_k)+\frac{1}{2}(\alpha_k d_k)^T H_k(\alpha_k d_k)+o(\|\alpha_k d_k\|^2) \\ &= \varphi_k(x_k+\alpha_k d_k)+o(\|\alpha_k d_k\|^2). \end{aligned}$$

So,

$$\lim_{\alpha_k \to 0} \frac{f(x_k) - f(x_k+\alpha_k d_k)}{f(x_k) - \varphi_k(x_k+\alpha_k d_k)} = 1 > c_0$$

Therefore, we can obtain trial step length $\alpha_k$ after finite computation, i.e., Algorithm 3.1 is well defined.

**Lemma 0.0.2.** *Suppose that Assumption 1 and 2 hold, and is the sequence generated by Algorithm 3.2, then* $\lim\limits_{k\to\infty} \|g_k\| = 0$.

**Proof.** $\{x_k,\, k = 1,\, 2,\, \ldots\}$ is generated by Algorithm 3.2, therefore, $x_k$, $d_k$ and $\alpha_k$ such that

$\rho_k = \frac{f(x_k)-f(x_k+\alpha_k d_k)}{f(x_k)-\phi(x_k+\alpha_k d_k)} \geq c_0 > 0$ and $f(x_k) - \phi(x_k + \alpha_k d_k) > 0$

So, $f(x_k) > f(x_{k+1})(x_{k+1} = x_k + \alpha_k d_k)$. It means that

$$f(x_0) > f(x_1) > f(x_2) > \ldots > f(x_k) > f(x_k + 1) \ldots .$$

According to Assumption 1, $f$ has bottom limits. Thus, $\lim_{k\to\infty} x_k = x^*$ and $\lim_{k\to\infty} f(x_k) = f(x^*)$. From Assumptions 1 and 2, two positive numbers $M_1$, $M_2$ exist, as well as a neighborhood $N(x^*, \delta)(\delta > 0)$ of $x^*$, such that $M_1 \leqslant \|\nabla^2 f(x)\| \leqslant M_2$ and $\forall x \in N(x^*, \delta)$. Then, using Mean-value theorem [11], we obtain

$$\|g_k\| = \left\| \int_0^1 \nabla^2 f(x^* + \theta(x_k - x^*))(x_k - x^*) d\theta \right\| \leqslant M \|x_k - x^*\|$$

$$\begin{aligned}
\|g_k\| &= \left\| \int_0^1 \nabla^2 f(x^* + \theta(x_k - x^*))(x_k - x^*) d\theta \right\| \\
&= \frac{\|x_k - x^*\| \left\| \int_0^1 \nabla^2 f(x^* + \theta(x_k - x^*))(x_k - x^*) d\theta \right\|}{\|x_k - x^*\|} \\
&\geqslant \frac{\|x_k - x^*\| \left\| (x_k - x^*) \int_0^1 \nabla^2 f(x^* + \theta(x_k - x^*))(x_k - x^*) d\theta \right\|}{\|x_k - x^*\|^2} \\
&\geqslant M_1 \|x_k - x^*\| .
\end{aligned}$$

Given that $\lim_{k\to\infty} x_k = x^*$, i.e., $\lim_{k\to\infty} \|x_k - x^*\| = 0$, then $\lim_{k\to\infty} \|g_k\| = 0$.

Finally, the general algorithm is derived by minimizing the cost function, obtaining optimal weight vector $W_o$, and outputting amplitude pattern $|p(W_o, \theta)|$. The algorithm is described as follows.

**Step 1.** The cost function $F(W, \alpha)$ is constructed as Equation (4).

**Step 2.** Let

$$X = \left(W_1^T, W_2^T, \alpha\right) \in \mathbb{R}^{2N+1}$$

then, computing $\nabla_{W_1} F$, $\nabla_{W_2} F$ and $F'_\alpha$ by Equations (A3)–(A5) respectively. Using Equations (A4), (A9) and (A10) (see the details in Appendix A), we obtain $\nabla^2_{W_1} F$, $\nabla^2_{W_2} F$ and $F''_\alpha$, whereafter constructing $\nabla_X F$ and $\nabla^2_X F$ as following:

$$\nabla_X F = \left( (\nabla_{W_1} F)^T, (\nabla_{W_2} F)^T, F'_\alpha \right)$$

$$\nabla^2_X F = \begin{pmatrix} \left(\nabla^2_{W_1} F\right)_{N\times N} & (0)_{N\times N} & 0 \\ (0)_{N\times N} & \left(\nabla^2_{W_2} F\right)_{N\times N} & 0 \\ 0 & 0 & F''_\alpha \end{pmatrix}_{(2N+1)\times(2N+1)}$$

The two formulas are essential to Algorithm 3.1. Finally, **Algorithm 3.2** is used to minimize the cost function $F(X)$.

**Step 3.** Based on the previous analysis, vector $W$ converges a optimal solution, if $\frac{F}{(180/\Delta\theta)} \rightarrow 0$, then array amplitude pattern is outputted.

## 4. SIMULATION RESULTS

In this section, several simulations are performed to test the penalty function method for synthesizing flat-top patterns. We assume that all array elements are isotropic and no mutual coupling.

**Simulation 1:** Comparison with the line search (LS) [10]

Consider a synthesis problem using a 17-element half-wave-length spacing linear array with the following configurations: $\theta_1 = -4.5$, $\theta_2 = -3.0$, $\theta_3 = 3.0$, and $\theta_4 = 4.5$. The simulations based on the DA and LS algorithms were executed 10 times independently, and the results are listed in Table 1.

The two iterative algorithms optimize the same cost functions in every stimulation, but the results show their distinct performance. Under LS, average wasted time, minimum PSLL and average PSLL are 40.54 second, $-22.73$ dB and $-21.25$ dB, and under DA, they are

**Table 1.** Comparison of the DA and LS algorithms.

| NO. | Algorithm | $r$ | $m$ | $\Delta\theta$ | ACF | PSLL(dB) | Time (s) |
|---|---|---|---|---|---|---|---|
| 1 | **DA** | **110** | **0.4** | **0.2** | **0.330050** | **-27.18** | **24.20** |
| | LS | 110 | 0.4 | 0.2 | 0.333661 | -21.16 | 54.69 |
| 2 | **DA** | **120** | **0.4** | **0.2** | **0.475723** | **-26.21** | **32.18** |
| | LS | 120 | 0.4 | 0.2 | 0.559412 | -20.41 | 35.43 |
| 3 | **DA** | **120** | **0.4** | **0.2** | **0.512754** | **-24.27** | **22.32** |
| | LS | 120 | 0.4 | 0.2 | 0.593485 | -19.80 | 29.01 |
| 4 | **DA** | **110** | **0.8** | **0.2** | **0.718242** | **-24.63** | **21.72** |
| | LS | 110 | 0.8 | 0.2 | 0.599144 | -21.50 | 40.03 |
| 5 | **DA** | **110** | **0.4** | **0.2** | **0.272849** | **-29.69** | **25.72** |
| | LS | 110 | 0.4 | 0.2 | 0.558577 | -21.42 | 39.37 |
| 6 | **DA** | **120** | **0.4** | **0.2** | **0.309771** | **-24.03** | **26.79** |
| | LS | 120 | 0.4 | 0.2 | 0.652165 | -22.33 | 50.69 |
| 7 | **DA** | **130** | **0.6** | **0.6** | **0.269191** | **-23.83** | **23.91** |
| | LS | 130 | 0.6 | 0.6 | 0.378267 | -19.01 | 34.45 |
| 8 | **DA** | **108** | **0.5** | **0.6** | **0.265707** | **-24.67** | **26.71** |
| | LS | 108 | 0.5 | 0.6 | 0.617897 | -21.99 | 40.12 |
| 9 | **DA** | **110** | **0.5** | **0.6** | **0.277525** | **-23.68** | **24.54** |
| | LS | 110 | 0.5 | 0.6 | 0.509221 | -22.73 | 40.75 |
| 10 | **DA** | **120** | **0.4** | **0.2** | **0.276542** | **-23.63** | **26.75** |
| | LS | 120 | 0.4 | 0.2 | 0.511626 | -22.22 | 40.92 |

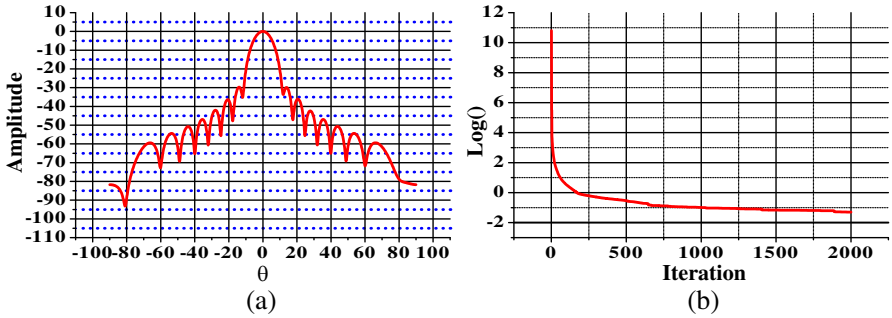


**Figure 2.** (a) Amplitude pattern and (b) cost function variety of 17-element line antenna array under DA algorithm.

**Table 2.** Optimal solution of 17-element line antenna array under DA algorithm.

| Element | $W_1$ | $W_2$ | Element | $W_1$ | $W_2$ | Element | $W_1$ | $W_2$ |
|---|---|---|---|---|---|---|---|---|
| **1** | –0.006852 | –0.010127 | **2** | –0.020647 | –0.030564 | **3** | –0.036663 | –0.054285 |
| **4** | –0.053729 | –0.079793 | **5** | –0.069787 | –0.103909 | **6** | –0.083744 | –0.124851 |
| **7** | –0.093917 | –0.139986 | **8** | –0.099285 | –0.148060 | **9** | –0.099111 | –0.147892 |
| **10** | –0.093792 | –0.139897 | **11** | –0.084071 | –0.124943 | **12** | –0.071362 | –0.105343 |
| **13** | –0.056918 | –0.083282 | **14** | –0.042212 | –0.061230 | **15** | –0.028050 | –0.040264 |
| **16** | –0.015600 | –0.022166 | **17** | –0.005144 | –0.007227 | α | 0.027799 | |

25.48 second, −29.69 dB and −25.18 dB, respectively. Undoubtedly, compared with LS, the DA algorithm exhibits stronger search ability in these simulations. We show the optimal amplitude pattern, ACF variety and optimal solution in Figures 2(a), (b) and Table 2, respectively.

**Simulation 2:** Comparison with NLS proposed in [9]

Consider a synthesis problem using a 37-element half-wave-length spacing linear array with the following configurations: $\theta_1 = -4.0$, $\theta_2 = -1.5$, $\theta_3 = 1.5$, and $\theta_4 = 4.0$. Executing the two algorithms 10 times independently and listing results in Table 3 as following.

Under NLS, average iterative number, minimum PSLL and average PSLL are 10000, −18.20 dB and −16.05 dB, and under PEM, they are 2135, −30.11 dB and −26.43 dB, respectively. The reported minimum PSLLs for synthesis of 37-element array are −20.56 dB in [8] and −24.11 dB in [2], respectively. Obviously, our result is much better than the best solutions of references [2] and [8]. The result indicates that the PEM has stronger ability to suppress sidelobe level because of

the penalty terms of cost function. These penalty terms ($\frac{P}{\alpha}$, $\frac{P}{f_1}$ and $\frac{P}{f_2}$) converge to big values when iterative algorithm converges to undesired solutions such that $P > \alpha$, $P > f_1$ and $P > f_2$. Contrarily, the penalty

**Table 3.** Comparison of the PEM and NLS algorithms.

| NO. | Algorithm | $r$ | $m$ | $\Delta\theta$ | ACF | PSLL(dB) | Iterative number |
|---|---|---|---|---|---|---|---|
| 1 | **PEM** | **140.0** | **0.6** | **0.6** | **0.081197** | **–30.11** | **8500** |
| | NLS | | | 0.6 | 0.002175 | –15.01 | 10000 |
| 2 | **PEM** | **140.0** | **0.6** | **0.4** | **0.108233** | **–28.46** | **8000** |
| | NLS | | | 0.6 | 0.008687 | –15.30 | 10000 |
| 3 | **PEM** | **120.0** | **0.8** | **0.4** | **0.069333** | **–25.05** | **2000** |
| | NLS | | | 0.6 | 0.019542 | –16.73 | 10000 |
| 4 | **PEM** | **130.0** | **1.5** | **0.2** | **0.008909** | **–24.65** | **1200** |
| | NLS | | | 0.6 | 0.034728 | –17.28 | 10000 |
| 5 | **PEM** | **130.0** | **1.5** | **0.6** | **0.010744** | **–25.16** | **1100** |
| | NLS | | | 0.6 | 0.078126 | –16.09 | 10000 |
| 6 | **PEM** | **130.0** | **1.4** | **0.6** | **0.002911** | **–25.10** | **1400** |
| | NLS | | | 0.4 | 0.001393 | –15.18 | 10000 |
| 7 | **PEM** | **150.0** | **0.8** | **0.6** | **0.074565** | **–25.83** | **1800** |
| | NLS | | | 0.4 | 0.000788 | –17.00 | 10000 |
| 8 | **PEM** | **150.0** | **1.3** | **0.4** | **0.003712** | **–25.96** | **2000** |
| | NLS | | | 0.4 | 0.001394 | –18.20 | 10000 |
| 9 | **PEM** | **130.0** | **0.8** | **0.4** | **0.075532** | **–26.83** | **2000** |
| | NLS | | | 0.4 | 0.001068 | –16.20 | 10000 |
| 10 | **PEM** | **130.0** | **0.8** | **0.2** | **0.104004** | **–27.18** | **550** |
| | NLS | | | 0.4 | 0.006767 | –13.56 | 10000 |

**Table 4.** Optimal solution of 37-element line antenna array under PEM.

| Element | $W_1$ | $W_2$ | Element | $W_1$ | $W_2$ | Element | $W_1$ | $W_2$ |
|---|---|---|---|---|---|---|---|---|
| **1** | –0.001164 | 0.002451 | **2** | –0.002811 | 0.006139 | **3** | –0.004483 | 0.010120 |
| **4** | –0.006476 | 0.014968 | **5** | –0.008660 | 0.020351 | **6** | –0.011135 | 0.026409 |
| **7** | –0.013746 | 0.032831 | **8** | –0.016510 | 0.039629 | **9** | –0.019322 | 0.046570 |
| **10** | –0.022175 | 0.053588 | **11** | –0.024981 | 0.060422 | **12** | –0.027696 | 0.066948 |
| **13** | –0.030237 | 0.072919 | **14** | –0.032646 | 0.078295 | **15** | –0.034821 | 0.082875 |
| **16** | –0.036713 | 0.086613 | **17** | –0.038244 | 0.089324 | **18** | –0.039417 | 0.091011 |
| **19** | –0.040168 | 0.091601 | **20** | –0.040461 | 0.091115 | **21** | –0.040244 | 0.089542 |
| **22** | –0.039506 | 0.086963 | **23** | –0.038238 | 0.083377 | **24** | –0.036543 | 0.078984 |
| **25** | –0.034399 | 0.073815 | **26** | –0.031858 | 0.068058 | **27** | –0.028973 | 0.061749 |
| **28** | –0.025868 | 0.055116 | **29** | –0.022605 | 0.048246 | **30** | –0.019346 | 0.041417 |
| **31** | –0.016128 | 0.034664 | **32** | –0.013046 | 0.028231 | **33** | –0.010101 | 0.022079 |
| **34** | –0.007446 | 0.016544 | **35** | –0.005028 | 0.011431 | **36** | –0.003034 | 0.007139 |
| **37** | –0.001204 | 0.002947 | $\alpha$ | 0.025167 | | | | |

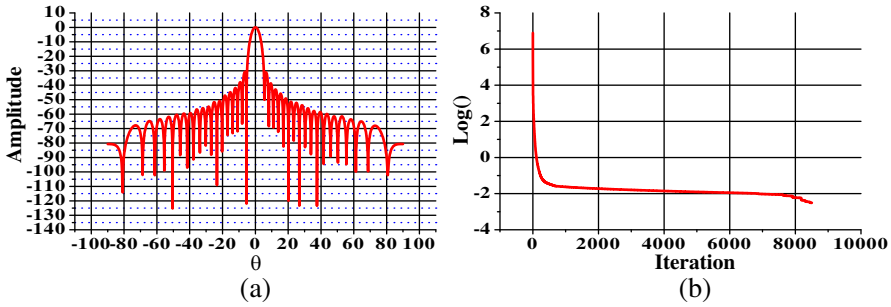


**Figure 3.** (a) Amplitude pattern and (b) cost function variety of 37-element line antenna array under PEM.

terms converge to small values when iterative algorithm converges to desired solutions such that $P < \alpha$, $P < f_1$ and $P < f_2$. Thereof, in the process of minimizing the cost function, the iterative algorithm tends to gradually move toward a optimal solution that makes the sidelobe level below to threshold. Finally, we show the optimal amplitude pattern, ACF variety and optimal solution in Figures 3(a), (b) and Table 4, respectively.

## 5. CONCLUSION

In this paper, we present a penalty function method for antenna array pattern synthesis (AAPS) problem, and an iterative optimization algorithm named DA to minimize the function. Based on flat-top model, the AAPS is transformed into a penalty function optimizing problem. The additional penalty terms to the cost function can identify desired or undesired solution by their values variety. Thereof, our method can converge to optimal solution more precisely than tradition flat-top method. The DA algorithm, unlike the traditional descent method, defines step length by solving inequality instead of merely using the Wolfe- or Armijo-type search rule, and can improve computational efficiency as indicated by the theoretical analysis and stimulation results. Upon application of the general algorithm to the AAPS problem, computer simulations illustrate its good performance.

## APPENDIX A. THE DERIVATION OF GRADIENT AND HESSIAN MATRIX OF COST FUNCTION

Let $tr\{\cdot\}$, $d\{\cdot\}$ and $vec(\cdot)$ denote the matrix trace, differential and the vector operator, respectively. The derivation of cost function's gradient

and Hessian matrix is based on the following properties [12]:

$$P_1 : d(XY) = (dX)Y + X(dY) \quad P_2 : d(tr(X)) = tr(dX)$$
$$P_3 : tr(X+Y) = tr(X) + tr(Y)$$

$P_4$ : Let $F$ be a differentiable real-valued function of an $m \times n$ matrix $X$. The following relationship then holds:

$$dF(X) = AdX \Leftrightarrow \nabla_X F = \frac{\partial F(X)}{\partial X} = A^T.$$

$P_5$ : Let $F$ be a twice-differentiable, real-valued function of an $m \times n$ matrix $X$. The following two relationships between the second differential and the Hessian matrix of $F$ at $X$ then hold:

$$d^2F(X) = tr(B(dX)^T CdX) \Leftrightarrow H(F(X)) = \frac{1}{2}(B^T \otimes C + B \otimes C^T)$$
$$d^2F(X) = tr(B(dX)CdX) \Leftrightarrow H(F(X)) = \frac{1}{2}K_{mn}(B^T \otimes C + C^T \otimes B)$$

where $K_{mn}$ is the switch matrix such that $K_{mn}vec(X) = vec(X^T)$.

The gradient matrix of the cost function (Equation (3)) is then calculated. Using $P_1$ and $P_2$, we obtain function $P$'s differential (versus vector $W_1$),

$$\begin{aligned} dP &= d\left((W_1^T S_1)^2 + 2(W_1^T S_1)(W_2^T S_2) + (W_2^T S_2)^2 + (W_1^T S_2)^2\right. \\ &\quad \left. -2(W_1^T S_2)(W_2^T S_1) + (W_2^T S_1)^2\right)\{P_2\} \\ &= 2(S_1^T W_1)S_1^T dW_1 + 2(W_2^T S_2)S_1^T dW_1 + 2(S_2^T W_1)S_2^T dW_1 \\ &\quad -2(W_2^T S_1)S_2^T dW_1\{P_1\} \\ &= 2[\left((S_1^T W_1)S_1^T + (W_2^T S_2)S_1^T + (S_2^T W_1)S_2^T - (W_2^T S_1)S_2^T\right) dW_1] \\ &= 2[\left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right) dW_1] \\ &\quad (t_1 = S_1^T W_1,\ t_2 = W_2^T S_2,\ t_3 = S_2^T W_1 \text{ and } t_4 = W_2^T S_1 \in \mathbb{R}) \end{aligned} \quad \text{(A1)}$$

Therefore, using Equation (A1) and $P_4$, we obtain

$$\nabla_{W_1} P = 2\left((t_1+t_2)S_1 + (t_3+t_4)S_2\right) \quad \text{(A2)}$$

For convenience, let

$$F_1 = (\alpha - P)^2 + \left(\frac{P}{\alpha}\right)^m,\ F_2 = (f_1 - P)^2 + \left(\frac{P}{f_1}\right)^m$$
$$F_3 = (\beta - P)^2 \quad \text{and} \quad F_4 = (f_2 - P)^2 + \left(\frac{P}{f_2}\right)^m$$

Thereof, $\nabla_{W_1} F_1 = \nabla_{W_1}\left((\alpha-P)^2+\left(\frac{P}{\alpha}\right)^m\right) = 2(P-\alpha)\nabla_{W_1}P + \frac{m}{\alpha}\left(\frac{P}{\alpha}\right)^{m-1}\nabla_{W_1}P$. Similarly,

$$\begin{aligned}
\nabla_{W_1} F_2 &= \left(2(P-f_1)+\frac{m}{f_1}\left(\frac{P}{f_1}\right)^{m-1}\right)\nabla_{W_1}P,\\
\nabla_{W_1} F_3 &= 2(P-\beta)\nabla_{W_1}P,\\
\nabla_{W_1} F_4 &= \left(2(P-f_2)+\frac{m}{f_2}\left(\frac{P}{f_2}\right)^{m-1}\right)\nabla_{W_1}P\\
&\quad(\because \nabla_{W_1}\alpha = \nabla_{W_1} f_1 = \nabla_{W_1} f_2 = 0)
\end{aligned}$$

Finally, we obtain,

$$\begin{aligned}
\nabla_{W_1} F(W,\alpha) &= \sum_1 \nabla_{W_1} F_1 + \sum_2 \nabla_{W_1} F_2 + \sum_3 \nabla_{W_1} F_3\\
&\quad + \sum_4 \nabla_{W_1} F_4 + \sum_5 \nabla_{W_1} F_1
\end{aligned}$$

i.e.,

$$\begin{aligned}
\nabla_{W_1} F = &\sum_1\left(2(P-\alpha)+\frac{m}{\alpha}\left(\frac{P}{\alpha}\right)^{m-1}\right)\nabla_{W_1}P\\
&+\sum_2\left(2(P-f_1)+\frac{m}{f_1}\left(\frac{P}{f_1}\right)^{m-1}\right)\nabla_{W_1}P_2\\
&+\sum_3 2(P-\beta)\nabla_{W_1}P+\sum_4\left(2(P-f_2)+\frac{m}{f_2}\left(\frac{P}{f_2}\right)^{m-1}\right)\nabla_{W_1}P\\
&+\sum_5\left(2(P-\alpha)+\frac{m}{\alpha}\left(\frac{P}{\alpha}\right)^{m-1}\right)\nabla_{W_1}P \qquad \text{(A3)}
\end{aligned}$$

Similarly,

$$\begin{aligned}
&\nabla_{W_2}P = 2\left((t_4-t_3)S_1+(t_1+t_2)S_2\right),\\
&\nabla_{W_2}F_1 = \left(2(P-\alpha)+\frac{m}{\alpha}\left(\frac{P}{\alpha}\right)^{m-1}\right)\nabla_{W_2}P,\\
&\nabla_{W_2}F_2 = \left(2(P-f_1)+\frac{m}{f_1}\left(\frac{P}{f_1}\right)^{m-1}\right)\nabla_{W_2}P,\ \nabla_{W_2}F_3 = 2(P-\beta)\nabla_{W_2}P,\\
&\nabla_{W_2}F_4 = \left(2(P-f_2)+\frac{m}{f_2}\left(\frac{P}{f_2}\right)^{m-1}\right)\nabla_{W_2}P\\
&\qquad(\because \nabla_{W_2}\alpha = \nabla_{W_2} f_1 = \nabla_{W_2} f_2 = 0)
\end{aligned}$$

$$\nabla_{W_2}F(W,\alpha) = \sum_1 \nabla_{W_2}F_1 + \sum_2 \nabla_{W_2}F_2 + \sum_3 \nabla_{W_2}F_3 + \sum_4 \nabla_{W_2}F_4 + \sum_5 \nabla_{W_2}F_1$$

i.e.,

$$\begin{aligned}\nabla_{W_2}F = &\sum_1 \left(2\left(P-\alpha\right)+\frac{m}{\alpha}\left(\frac{P}{\alpha}\right)^{m-1}\right)\nabla_{W_2}P \\ &+\sum_2 \left(2\left(P-f_1\right)+\frac{m}{f_1}\left(\frac{P}{f_1}\right)^{m-1}\right)\nabla_{W_2}P \\ &+\sum_3 2\left(P-\beta\right)\nabla_{W_2}P+\sum_4 \left(2\left(P-f_2\right)+\frac{m}{f_2}\left(\frac{P}{f_2}\right)^{m-1}\right)\nabla_{W_2}P \\ &+\sum_5 \left(2\left(P-\alpha\right)+\frac{m}{\alpha}\left(\frac{P}{\alpha}\right)^{m-1}\right)\nabla_{W_2}P \end{aligned} \tag{A4}$$

$$\begin{aligned}&(F_1)'_\alpha = 2\left(\alpha - P\right) - mP^m\alpha^{-m-1}, \\ &(F_2)'_\alpha = 2\left(f_1 - P\right)(f_1)'_\alpha - mP^m f_1^{-m-1}(f_1)'_\alpha \\ &(F_3)'_\alpha = 2r\left(\beta - P\right) \quad (\beta = r\alpha,\ r > 100), \\ &(F_4)'_\alpha = 2\left(f_2 - P\right)(f_2)'_\alpha - mP^m f_2^{-m-1}(f_2)'_\alpha \\ &(\lg|\alpha|)'_\alpha = \frac{1}{\alpha},\ (f_1)'_\alpha = \frac{(k-1)}{(\theta_2-\theta_1)}(\theta-\theta_1)+1,\ (f_2)'_\alpha = \frac{(k-1)}{(\theta_3-\theta_4)}(\theta-\theta_4)+1 \\ &F'_\alpha(W,\alpha) = \sum_1 (F_1)'_\alpha + \sum_2 (F_2)'_\alpha + \sum_3 (F_3)'_\alpha + \sum_4 (F_4)'_\alpha + \sum_5 (F_1)'_\alpha - \lambda\alpha^{-1}\end{aligned}$$

i.e.,

$$\begin{aligned}F'_\alpha = &\sum_1 \left(2\left(\alpha-P\right)-mP^m\alpha^{-m-1}\right)+\sum_2 \left(2\left(f_1-P\right)(f_1)'_\alpha\right. \\ &\left.-mP^m f_1^{-m-1}(f_1)'_\alpha\right)+\sum_3 \left(2r\left(\beta-P\right)\right)+\sum_4 \left(2\left(f_2-P\right)(f_2)'_\alpha\right. \\ &\left.-mP^m f_2^{-m-1}(f_2)'_\alpha\right)+\sum_5 \left(2\left(\alpha-P\right)-mP^m\alpha^{-m-1}\right)-\lambda\alpha^{-1}\end{aligned} \tag{A5}$$

Next, we calculate the Hessian matrices of the cost function. First, the function $P$'s twice differential (versus vector $W_1$) is written as

(vector $dW_1$ is considered constant)

$$
\begin{aligned}
&d^2(P) = d(d(P)) \\
&= d(2[\left((S_1^T W_1)S_1^T + (W_2^T S_2)S_1^T + (S_2^T W_1)S_2^T - (W_2^T S_1)S_2^T\right) dW_1]) \\
&= 2(S_1^T dW_1 S_1^T dW_1 + S_2^T dW_1 S_2^T dW_1).
\end{aligned}
$$

Since $F_1 \in \mathbb{R}$, thereof, $F_1 = tr(F_1)$,

$$
\begin{aligned}
dF_1 &= tr\left(d\left((\alpha - P)^2 + \left(\frac{P}{\alpha}\right)^m\right)\right) \\
&= tr\left(2(P-\alpha)dP + \frac{m}{\alpha}\left(\frac{P}{\alpha}\right)^{m-1} dP\right) \\
d^2F_1 &= tr\left(d\left(2(P-\alpha)dP + \frac{m}{\alpha}\left(\frac{P}{\alpha}\right)^{m-1} dP\right)\right) \\
&= tr\Big(2dP \cdot dP + 2(P-\alpha)d^2P + m(m-1)P^{(m-2)}\alpha^{-m} dP \cdot dP \\
&\quad + mP^{(m-1)}\alpha^{-m} d^2P\Big) \\
&= tr\left(\left(2 + m(m-1)P^{(m-2)}\alpha^{-m}\right) dP \cdot dP\right) \\
&\quad + tr\left(\left(2(P-\alpha) + mP^{(m-1)}\alpha^{-m}\right) d^2P\right)\{P_3\}
\end{aligned}
$$

Let

$$
\begin{aligned}
d^2F_{11} &= tr\left(\left(2 + m(m-1)P^{(m-2)}\alpha^{-m}\right) dP \cdot dP\right), \\
d^2F_{12} &= tr\left(\left(2(P-\alpha) + mP^{(m-1)}\alpha^{-m}\right) d^2P\right),
\end{aligned}
$$

so,

$$
\nabla^2 F_1 = \nabla^2 F_{11} + \nabla^2 F_{12}. \tag{A6}
$$

On the other hand,

$$
\begin{aligned}
d^2F_{11} &= tr\left(\left(2 + m(m-1)P^{(m-2)}\alpha^{-m}\right) dP \cdot dP\right) \\
&= tr\left(4\left(2 + m(m-1)P^{(m-2)}\alpha^{-m}\right)\right. \\
&\left.\left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right) dW \cdot \left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right) dW\right) \\
&\therefore \nabla^2 F_{11} = 2K_N\left(B_{11}^T \otimes C_{11} + B_{11} \otimes C_{11}^T\right)\{\ \mathrm{P}_5\}
\end{aligned} \tag{A7}
$$

where

$$
\begin{aligned}
B_{11} &= \left(2 + m(m-1)P^{(m-2)}\alpha^{-m}\right)\left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right) \\
C_{11} &= \left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right) \\
K_N &= I_N\left(\because I_N vec\,(W_1) = vec\left(W_1^T\right)\right)
\end{aligned} \tag{A8}
$$

Similarly,

$$\begin{aligned}\nabla^2 F_{12} =& K_N\left(B_{121}^T \otimes C_{121} + B_{121} \otimes C_{121}^T\right)\\ &+ K_N\left(B_{122}^T \otimes C_{122} + B_{122} \otimes C_{122}^T\right)\\ B_{121} =& \left(2(P-\alpha) + mP^{(m-1)}\alpha^{-m}\right) S_1^T,\ C_{121} = S_1^T\\ B_{122} =& \left(2(P-\alpha) + mP^{(m-1)}\alpha^{-m}\right) S_2^T,\ C_{122} = S_2^T\end{aligned} \tag{A9}$$

From Equations (A6)–(A9), we obtain,

$$\begin{aligned}&\nabla_{W_1}^2 F_1 = 2\left(2 + m(m-1)P^{(m-2)}\alpha^{-m}\right)\\ &\left(\begin{array}{c}((t_1+t_2)S_1 + (t_3+t_4)S_2)\otimes\left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right) +\\ \left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right)\otimes((t_1+t_2)S_1 + (t_3+t_4)S_2)\end{array}\right)\\ &+\left(2(P-\alpha)+mP^{(m-1)}\alpha^{-m}\right)\left(S_1\otimes S_1^T + S_1^T\otimes S_1 + S_2\otimes S_2^T + S_2^T\otimes S_2\right)\end{aligned} \tag{A10}$$

By comparing the functions $F_1$, $F_2$, $F_3$ and $F_4$ above, it is found that functions $\alpha$, $\beta$, $f_1$ and $f_2$ are equal entirely in computing $\nabla_{W_1}^2 F(W,\theta)$ because they are all independent of vector $W_1$. Thereof, substituting $f_1$, $\beta$ and $f_2$ for $\alpha$ in above equations correspondingly, we obtain the formulation of $\nabla_{W_1}^2 F_2$, $\nabla_{W_1}^2 F_3$ (additionally let $m=0$) and $\nabla_{W_1}^2 F_4$,

$$\begin{aligned}&\nabla_{W_1}^2 F_2 = 2\left(2 + m(m-1)P^{(m-2)}f_1^{-m}\right)\\ &\left(\begin{array}{c}((t_1+t_2)S_1 + (t_3+t_4)S_2)\otimes\left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right) +\\ \left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right)\otimes((t_1+t_2)S_1 + (t_3+t_4)S_2)\end{array}\right)\\ &+\left(2(P-f_1)+mP^{(m-1)}f_1^{-m}\right)\left(S_1\otimes S_1^T + S_1^T\otimes S_1 + S_2\otimes S_2^T + S_2^T\otimes S_2\right)\end{aligned} \tag{A11}$$

$$\begin{aligned}\nabla_{W_1}^2 F_3 =& 4\left(\begin{array}{c}((t_1+t_2)S_1+(t_3+t_4)S_2)\otimes\left((t_1+t_2)S_1^T+(t_3+t_4)S_2^T\right)+\\ \left((t_1+t_2)S_1^T+(t_3+t_4)S_2^T\right)\otimes((t_1+t_2)S_1+(t_3+t_4)S_2)\end{array}\right)\\ &+2(P-\beta)\left(S_1\otimes S_1^T + S_1^T\otimes S_1 + S_2\otimes S_2^T + S_2^T\otimes S_2\right)\end{aligned} \tag{A12}$$

$$\begin{aligned}&\nabla_{W_1}^2 F_4 = 2\left(2+m(m-1)P^{(m-2)}f_2^{-m}\right)\\ &\left(\begin{array}{c}((t_1+t_2)S_1 + (t_3+t_4)S_2)\otimes\left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right) +\\ \left((t_1+t_2)S_1^T + (t_3+t_4)S_2^T\right)\otimes((t_1+t_2)S_1 + (t_3+t_4)S_2)\end{array}\right)\\ &+\left(2(P-f_2)+mP^{(m-1)}f_2^{-m}\right)\left(S_1\otimes S_1^T + S_1^T\otimes S_1 + S_2\otimes S_2^T + S_2^T\otimes S_2\right)\end{aligned} \tag{A13}$$

Finally, using Equations (A10)–(A13), we obtain the Hessian matrix $\nabla^2_{W_1}F(W,\theta)$ as following.

$$\begin{aligned}\nabla^2_{W_1}F \;=\; &\sum_1 \nabla^2_{W_1}F_1 + \sum_2 \nabla^2_{W_1}F_2 + \sum_3 \nabla^2_{W_1}F_3 \\ &+ \sum_4 \nabla^2_{W_1}F_4 + \sum_5 \nabla^2_{W_1}F_1 \end{aligned} \tag{A14}$$

Similarly, it is not difficult to compute the Hessian matrix $\nabla^2_{W_2}F(W,\alpha)$.

$$\begin{aligned}&\nabla^2_{W_2}F_1 = 2\left(2+m(m-1)P^{(m-2)}\alpha^{-m}\right) \\ &\begin{pmatrix}((t_4-t_3)S_1+(t_1+t_2)S_2)\otimes\big((t_4-t_3)S_1^T+(t_1+t_2)S_2^T\big)+\\ \big((t_4-t_3)S_1^T+(t_1+t_2)S_2^T\big)\otimes((t_4-t_3)S_1+(t_1+t_2)S_2)\end{pmatrix} \\ &+\Big(2(P-\alpha)+mP^{(m-1)}\alpha^{-m}\Big)\big(S_1\otimes S_1^T+S_1^T\otimes S_1+S_2\otimes S_2^T+S_2^T\otimes S_2\big)\end{aligned} \tag{A15}$$

$$\begin{aligned}&\nabla^2_{W_2}F_2 = 2\left(2+m(m-1)P^{(m-2)}f_1^{-m}\right) \\ &\begin{pmatrix}((t_4-t_3)S_1+(t_1+t_2)S_2)\otimes\big((t_4-t_3)S_1^T+(t_1+t_2)S_2^T\big)+\\ \big((t_4-t_3)S_1^T+(t_1+t_2)S_2^T\big)\otimes((t_4-t_3)S_1+(t_1+t_2)S_2)\end{pmatrix} \\ &+\Big(2(P-f_1)+mP^{(m-1)}f_1^{-m}\Big)\big(S_1\otimes S_1^T+S_1^T\otimes S_1+S_2\otimes S_2^T+S_2^T\otimes S_2\big)\end{aligned} \tag{A16}$$

$$\begin{aligned}\nabla^2_{W_2}F_3=4&\begin{pmatrix}((t_4-t_3)S_1+(t_1+t_2)S_2)\otimes\big((t_4-t_3)S_1^T+(t_1+t_2)S_2^T\big)+\\ \big((t_4-t_3)S_1^T+(t_1+t_2)S_2^T\big)\otimes((t_4-t_3)S_1+(t_1+t_2)S_2)\end{pmatrix} \\ &+2(P-f_1)\big(S_1\otimes S_1^T+S_1^T\otimes S_1+S_2\otimes S_2^T+S_2^T\otimes S_2\big)\end{aligned} \tag{A17}$$

$$\begin{aligned}&\nabla^2_{W_2}F_4 = 2\left(2+m(m-1)P^{(m-2)}f_2^{-m}\right) \\ &\begin{pmatrix}((t_4-t_3)S_1+(t_1+t_2)S_2)\otimes\big((t_4-t_3)S_1^T+(t_1+t_2)S_2^T\big)+\\ \big((t_4-t_3)S_1^T+(t_1+t_2)S_2^T\big)\otimes((t_4-t_3)S_1+(t_1+t_2)S_2)\end{pmatrix} \\ &+\Big(2(P-f_2)+mP^{(m-1)}f_2^{-m}\Big)\big(S_1\otimes S_1^T+S_1^T\otimes S_1+S_2\otimes S_2^T+S_2^T\otimes S_2\big)\end{aligned} \tag{A18}$$

using Equations (A15)–(A18), we obtain the Hessian matrix $\nabla^2_{W_2}F(W,\alpha)$ as following.

$$\nabla^2_{W_2}F=\sum_1\nabla^2_{W_2}F_1+\sum_2\nabla^2_{W_2}F_2+\sum_3\nabla^2_{W_2}F_3+\sum_4\nabla^2_{W_2}F_4+\sum_5\nabla^2_{W_2}F_1 \tag{A19}$$

At last, computing the second derivation of the cost function $F(W, \alpha)$ (versus variety $\alpha$) below.

$$\begin{aligned}
&F_1'' = 2 + m(m+1)P^m\alpha^{-m-2},\\
&F_2'' = 2\left(f_1'\right)^2 + m(m+1)P^m f_1^{-m-2}\left(f_1'\right)^2\\
&F_3'' = 2r^2,\ F_4'' = 2\left(f_2'\right)^2 + m(m+1)P^m f_2^{-m-2}\left(f_2'\right)^2\\
&(\lg|\alpha|)'' = -\alpha^{-2},\ f_1' = \frac{(k-1)}{(\theta_2-\theta_1)}(\theta-\theta_1)+1,\\
&f_2' = \frac{(k-1)}{(\theta_3-\theta_4)}(\theta-\theta_4)+1\\
&F''(W,\theta) = \sum_1 F_1'' + \sum_2 F_2'' + \sum_3 F_3'' + \sum_4 F_4'' + \sum_5 F_1'' + \lambda\alpha^{-2}
\end{aligned} \tag{A20}$$

## ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (NNSF) under Grants 60990320, 60990323 and 61271090, the National 863 Project of China under Grant 2012AA012305, Sichuan Provincial Science and technology support Project under Grant 2012GZ0101, and Chengdu Science and technology support Project under Grant 12DXYB347JH-002.

## REFERENCES

1. Li, R., L. Xu, X.-W. Shi, N. Zhang, and Z.-Q. Lv, "Improved differential evolution strategy for antenna array pattern synthesis problems," *Progress In Electromagnetics Research*, Vol. 113, 429–441, 2011.
2. Lin, C., A.-Y. Qing, and Q.-Y. Feng, "Synthesis of unequally spaced antenna arrays by using differential evolution," *IEEE Transactions on Antennas and Propagation*, Vol. 58, 2553–2561, 2010.
3. Wang, W.-B., Q.-Y. Feng, and D. Liu, "Application of chaotic particle swarm optimization algorithm to pattern synthesis of antenna arrays," *Progress In Electromagnetics Research*, Vol. 115,173–189, 2011.
4. Liu, D., Q.-Y. Feng, W.-B. Wang, and X. Yu, "Synthesis of unequally spaced antenna arrays by using inheritance learning particle swarm optimization," *Progress In Electromagnetics Research*, Vol. 118, 205–221, 2011.

5. Li, W.-T., Y.-Q. Hei, and X.-W. Shi, "Pattern synthesis of conformal arrays by a modified particle swarm optimization," *Progress In Electromagnetics Research*, Vol. 117, 237–252, 2011.
6. Mahanti, G. K., N. Pathak, and P. Mahanti, "Synthesis of thinned linear antenna arrays with fixed sidelob elevel using real-coded genetic algorithm," *Progress In Electromagnetics Research*, Vol. 75, 319–328, 2007.
7. Xu, Z.,et al., "Pattern synthesis of conformal antenna array by the hybrid genetic algorithm," *Progress In Electromagnetics Research*, Vol. 79, 75–90, 2008.
8. Chen, K., Z. He, and C. Han, "A modified real GA for the sparse linear array synthesis with multiple constraints," *IEEE Trans. Antennas Propag.*, Vol. 54, No. 7, 2169–2173, 2006.
9. Yuan, W., et al., "Nonlinear least-square solution to flat-top pattern synthesis using arbitrary linear array," *Signal Processing*, Vol. 85, 1869–1874, Sept. 2005.
10. Li, M. and H. Feng, "A sufficient descent LS conjugate gradient method for unconstrained optimization problems," *Applied Mathematics and Computation*, Vol. 218, 1577–1586, Nov. 1, 2011.
11. Nocedal, J. and S. J. Wright, *Numerical optimization*, Springer Verlag, 1999.
12. Zhang, X.-D., *Matrix Analysis and Application*, Springer, 2004.