Fast Sidelobe Calculation for Planar Phased Arrays Using an Iterative Sidelobe Seeking Method

Ke Miao, Yi Zhang *, Handan Jing, Shuoguang Wang and Houjun Sun

Beijing Key Laboratory of Millimeter Wave and Terahertz Technology, School of Integrated Circuits and Electronics, Beijing Institute of Technology, Beijing 100081, China
* Correspondence: yi.zhang@bit.edu.cn

Abstract: Sidelobe level is a key parameter for phased array design. However, the conventional method for sidelobe level calculation is time-consuming due to the high sampling number of the radiation pattern. In this paper, an iterative sidelobe-seeking method for sidelobe calculation of arbitrarily distributed planar arrays is proposed to shorten the computing time. By applying Newton’s iterative method with an approximate formula, the proposed method uses less sampling to obtain the sidelobe positions and the peak sidelobe level. Numerical experiments were carried out and verified the effectiveness of the proposed method. Compared with the conventional method, the proposed method had a similar optimization effect but saved 80% of the algorithm time in the synthesis of a sparse planar array.

Keywords: array synthesis; genetic algorithm; Newton’s iterative method; sidelobe calculation; sparse array

1. Introduction

Phased array is widely utilized in radar, communication and wireless power transfer applications [1–3]. In the design of a phased array, a lower sidelobe level (SLL) means less power loss and signal interference from undesired directions [4,5]. As a result, sidelobe calculation is applied in many problems of phased array design [6–16]. A number of optimization problems [6–11] take SLL calculation as the cost function directly, while others employ SLL calculation as the criterion of the scanning performance [12–14,17]. The SLL computing time has a direct influence on the total algorithm time in array design.

Previous studies for SLL calculation have employed two main kinds of method, the conventional method [4,6] and the fast formula method [18].

The conventional method first calculates the array pattern and then searches the sidelobe outside the main lobe region. To accurately find the sidelobe, the conventional method usually adopts a higher sampling number, which results in a longer computing time. The time complexity of the conventional method can simply be given by $O(mn)$, where $m$ is the sampling number and $n$ is the number of total elements. In some papers [13,14], sampling of the radiation pattern is simplified to $\phi = 0$ and $\phi = \frac{\pi}{2}$ planes. Although the simplification of sampling saves computing time, the optimization results are often only effective near these two planes.

The fast formula method takes advantage of an approximate formula based on estimated sidelobe positions. This method avoids the procedure of angle sampling and, thus, has a lower time complexity, which is given as $O(n)$. However, the effectiveness of the fast formula method is sensitive to the estimated sidelobe positions. Therefore, applications of this method are limited to the optimization of the amplitudes or the phases in an array with determined element positions.

To overcome the drawbacks of previous studies, we propose an iterative sidelobe-seeking method for peak sidelobe level (PSLL) fast calculation. The proposed method considers the SLL calculation problem as a zero-finding problem of the radiation pattern
derivative function. In numerical analysis, Newton’s iterative method is a fast convergent, zero-finding algorithm [19]. Combining the approximate formula in [18] with Newton’s iterative method, the proposed method adopts simple sampling of the angle domain to gradually approach the actual sidelobe positions. After comparison with the edge pattern level, the PSLL is finally obtained.

The novel features and contributions of this study can be summarized as follows,

(1) The combination of Newton’s iterative method and the approximate linear formula not only reduces the calculation pressure but also avoids the input of the estimated sidelobe positions in [18]. The linear formula facilitates the computing of second derivatives. Newton’s method removes the formula’s dependence on initial estimated sidelobe positions, which extends the application range of the formula to all kinds of array syntheses.

(2) A parallel iterative sidelobe-seeking framework is proposed to search sidelobe positions. Compared to the conventional method, the proposed framework reduces the amount of pattern sampling. Although the time complexity is still $O(mn)$, $m$ is much smaller than that of the conventional method and, thus, computing time is saved.

The rest of this paper is organized as follows: Section II describes the mathematical model and algorithm details. Section III describes three numerical experiments undertaken to verify the proposed method. Finally, conclusions are drawn in Section IV.

2. Formulation and Algorithm

2.1. Formulation of Sidelobes in a Planar Phased Array

Figure 1 illustrates the geometry model of a planar phased array of $N$ elements. $\theta$ and $\phi$ are the spherical coordinate angles of a far field point $r$. When the beam of the phased array is steered to the direction of $(\theta_0, \phi_0)$, the array factor could be expressed as

$$F(u, v) = \sum_n I_n \exp[\beta(u-u_0)x_n + \beta(v-v_0)y_n + \delta_n]$$  \hspace{1cm} (1)

where $\beta$ is the wave number. $(x_n, y_n)$ is the position of the $n^{th}$ element. $I_n$ is the exciting amplitude of the $n^{th}$ element. $\delta_n$ is the phase error. $u = \sin \theta \cos \phi$, $v = \sin \theta \sin \phi$, $u_0 = \sin \theta_0 \cos \phi_0$, $v_0 = \sin \theta_0 \sin \phi_0$.

![Figure 1. A planar phased array consisting of N elements positioned on the xy-plane.](image-url)
\[ P(u, v) = F(u, v) \cdot F^*(u, v) \]
\[ = \sum_{n} \sum_{q} I_n I_q e^{i[\beta(u-u_0)(x_n-x_q) + \beta(v-v_0)(y_n-y_q) + (\delta_n - \delta_q)]} \]
\[ = \sum_{n} \sum_{q} I_n I_q \cos[\beta(u-u_0)(x_n-x_q) + \beta(v-v_0)(y_n-y_q) + \delta_n - \delta_q]. \]

where, \( F^*(u, v) \) is the complex conjugate of \( F(u, v) \).

As the sidelobes are at the extreme points of \( P(u, v) \), the problem of finding the sidelobe positions involves finding the solutions of

\[
\begin{align*}
\frac{\partial P(u, v)}{\partial u} &= 0 \\
\frac{\partial P(u, v)}{\partial v} &= 0
\end{align*}
\]

2.2. Newton’s Iterative Method

To obtain the roots of (3), we introduce Newton’s method of two variables. The \((k+1)th\) iteration is given by

\[
\begin{align*}
{u}_{(k+1)} &= u_{(k)} + \Delta u_{(k)}, \\
{v}_{(k+1)} &= v_{(k)} + \Delta v_{(k)},
\end{align*}
\]

where,

\[
\begin{bmatrix}
\Delta u_{(k)} \\
\Delta v_{(k)}
\end{bmatrix} = \left( H(u, v)^{-1} \cdot \begin{bmatrix}
-\frac{\partial^2 P(u, v)}{\partial u^2} & -\frac{\partial^2 P(u, v)}{\partial u \partial v} \\
-\frac{\partial^2 P(u, v)}{\partial u \partial v} & -\frac{\partial^2 P(u, v)}{\partial v^2}
\end{bmatrix} \right)_{u,v=u_{(k)},v_{(k)}},
\]

\( H(u, v) \) is the Hessian matrix of \( P(u, v) \) in the form of

\[
H(u, v) = \begin{bmatrix}
\frac{\partial^2 P(u, v)}{\partial u^2} & \frac{\partial^2 P(u, v)}{\partial u \partial v} \\
\frac{\partial^2 P(u, v)}{\partial u \partial v} & \frac{\partial^2 P(u, v)}{\partial v^2}
\end{bmatrix},
\]

It is costly to directly use (4) due to the second derivatives in (6) and the matrix inversion in (5). Therefore, some approximation is needed. Considering that \((u(k), v(k))\) are approaching the true extreme point \((u_m, v_m)\) after iterations, they can be used as the estimated sidelobe positions in [18] and the elements in the Hessian matrix can be approximated as,

\[
\begin{align*}
\frac{\partial^2 P(u, v)}{\partial u^2} \bigg|_{u,v=u_{(k)},v_{(k)}} &\approx -2\beta^2 \sum_q I_q \sum_n I_n^2 x_n (x_n - \Omega_x) + \sum_q I_q \sum_n I_n^2 x_n (x_n - \Omega_x) = -2\beta^2 a \\
\frac{\partial^2 P(u, v)}{\partial u \partial v} \bigg|_{u,v=u_{(k)},v_{(k)}} &\approx -2\beta^2 \sum_q I_q \sum_n I_n y_n (y_n - \Omega_y) + \sum_q I_q \sum_n I_n y_n (y_n - \Omega_y) = -2\beta^2 b \\
\frac{\partial^2 P(u, v)}{\partial v^2} \bigg|_{u,v=u_{(k)},v_{(k)}} &\approx -2\beta^2 \sum_q I_q \sum_n I_n y_n (y_n - \Omega_y) + \sum_q I_q \sum_n I_n y_n (y_n - \Omega_y) = -2\beta^2 d
\end{align*}
\]
\[
\frac{\partial P(u, v)}{\partial u} \bigg|_{u, v = (u(k), v(k))} \approx -2\beta \{ \sum_q I_q^u \sum_n [\delta_n I_n^u (x_n - O_x^u)] + \sum_q I_q^v \sum_n [\delta_n I_n^v (x_n - O_x^v)] \} - 2\beta \sum_q I_q^u \sum_n I_n^u (x_n - O_x^u) = -2\beta e,
\]

\[
\frac{\partial P(u, v)}{\partial v} \bigg|_{u, v = (u(k), v(k))} \approx -2\beta \{ \sum_q I_q^v \sum_n [\delta_n I_n^v (y_n - O_y^v)] + \sum_q I_q^u \sum_n [\delta_n I_n^u (y_n - O_y^u)] \} - 2\beta \sum_q I_q^v \sum_n I_n^v (y_n - O_y^v) = -2\beta f.
\]

where

\[
O_x^u = \frac{\sum_q x_q I_q^u}{\sum_q I_q^u}, \quad O_y^u = \frac{\sum_q y_q I_q^u}{\sum_q I_q^u},
\]

\[
O_x^v = \frac{\sum_q x_q I_q^v}{\sum_q I_q^v}, \quad O_y^v = \frac{\sum_q y_q I_q^v}{\sum_q I_q^v},
\]

and

\[
I_n^u = I_n \cos(\beta \Delta u_n + \beta \Delta v_n),
\]

\[
I_n^v = I_n \sin(\beta \Delta u_n + \beta \Delta v_n),
\]

\[
\Delta u = u_{(k)} - u_0,
\]

\[
\Delta v = v_{(k)} - v_0
\]

The iterative relation (4) is converted to a linear structure as,

\[
\begin{bmatrix}
  u_{(k+1)} - u_{(k)} \\
  v_{(k+1)} - v_{(k)}
\end{bmatrix} \approx \frac{1}{\beta} \begin{bmatrix}
  a & b \\
  b & d
\end{bmatrix}^{-1} \begin{bmatrix}
  -e \\
  -f
\end{bmatrix},
\]

where \(a, b, d, e\) and \(f\) are defined in (7) and (8).

When the condition \(\max(|u_{(k+1)} - u_{(k)}|, |v_{(k+1)} - v_{(k)}|) < \varepsilon\) is met, the iteration is convergent. The positions of the sidelobe \(u_{SL}\) and \(v_{SL}\) are

\[
\begin{align*}
  u_{SL} &= u_{(k+1)} \\
  v_{SL} &= v_{(k+1)}
\end{align*}
\]

Through the Formula (1), we finally derive the sidelobe level,

\[
P(u_{SL}, v_{SL}) = |F(u_{SL}, v_{SL})|^2.
\]

2.3. Iterative Sidelobe-Seeking Framework

Since (11) and (12) provide access to the level of a single sidelobe, we need parallel processing to approach multi-sidelobes to determine the peak sidelobe of the full pattern. Figure 2 presents the main lobe and a sidelobe in a quarter \(u - v\) pattern to show the chief procedure of the proposed method. The specific details are as follows:
2.3.1. Initial Sampling

Initial sampling provides multiple inputs \((u(1), v(1))\) for parallel Newton iterations. The sampling methods include rectangular sampling as well as random sampling. Rectangular grid sampling is defined as,

\[
\begin{align*}
  u_{ij}^{(1)} &= -1 + \frac{2(i - 1)}{N_s - 1}, i = 1, 2, 3 \ldots, N_s \\
  v_{ij}^{(1)} &= -1 + \frac{2(j - 1)}{N_s - 1}, j = 1, 2, 3 \ldots, N_s \\
  (u_{ij}^{(1)})^2 + (v_{ij}^{(1)})^2 &\leq 1
\end{align*}
\]

where \(N_s\) is the sampling number of \(u\) or \(v\) dimension. The rectangular grid sampling method is also applicable to the conventional method of SLL calculation.

Random sampling adopts a uniform probability distribution for the sampling points in the \(u - v\) visible area. Both sampling methods are effective here. A discussion of the methods is provided in the next section.

2.3.2. Newton’s Iteration

This step involves one iteration period using (11) for all the sampling points in parallel. We introduce (13) and calculate the power of these iterated points. For the \(k^{th}\) iteration, we have

\[
\begin{align*}
  (u_1^{(k)}, v_1^{(k)}, P(u_1^{(k)}, v_1^{(k)})), \\
  (u_2^{(k)}, v_2^{(k)}, P(u_2^{(k)}, v_2^{(k)})), \\
  \vdots \\
  (u_n^{(k)}, v_n^{(k)}, P(u_n^{(k)}, v_n^{(k)}))
\end{align*}
\]

2.3.3. Data Processing

After the iteration, the output points move closer to their nearest extreme points compared to their input. However, they cannot immediately be put into the next iteration before three steps of processing have occurred. The next three steps are employed to reduce the subsequent calculation.

1. Data exclusion. The iterated points may move out of the visible area or into the main lobe area. These points should be excluded due to their meaningless for subsequent solution. The visible area conforms to \(u^2 + v^2 \leq 1\).
The main lobe area requires an evaluation of beam width. We first calculate the pattern of the array in \( u_0 \)-cut and \( v_0 \)-cut as,

\[
\begin{align*}
P_{u_0}(u) &= |F(u, v_0)|^2, \\
P_{v_0}(v) &= |F(u_0, v)|^2.
\end{align*}
\] (16)

Secondly, the \( \gamma \) dB beam width can be determined from each cut pattern. \( \gamma \) is set according to the size of the array. The two beam widths are defined as \( \Gamma_u \) and \( \Gamma_v \). As the main lobe is approximately elliptical in the \( u-v \) system, the area is marked as,

\[
\left(\frac{u-u_0}{2\Gamma_u}\right)^2 + \left(\frac{v-v_0}{2\Gamma_v}\right)^2 \leq 1
\] (17)

2. Data interception. This step cuts off the iterated points with low power due to their low possibility of iterating to the extreme points. The criteria for the interception are a certain proportion of the total points or a certain power value.

3. Data merging. This step merges the close points into one point. If two points \((u_a, v_a)\) and \((u_b, v_b)\) meet the condition \((u_a - u_b)^2 + (v_a - v_b)^2 < L\), they are to be merged. \( L \) is a preset parameter.

2.3.4. Additional Sampling

The initial sampling may not cover all the sidelobes. Therefore, this step adds extra sampling around the points of the higher power. In our programming, the extra samplings \((u_e, v_e)\) for a single sampling point \((u_{(k)}, v_{(k)})\) can be expressed as,

\[
\begin{align*}
u_e^{(i)} &= u_{(k)} + R_e \cos[a(i)] \\
v_e^{(i)} &= v_{(k)} + R_e \sin[a(i)]
\end{align*}
\] (18)

where, \( a(i) = (i-1)2\pi/N_e \), and \( N_e \) is a user-defined parameter. \( R_e \) reflects the precision of the radiation pattern and is usually evaluated by \( \Gamma_u \) and \( \Gamma_v \) as,

\[
R_e = \max(\Gamma_u, \Gamma_v).
\] (19)

2.3.5. Iteration and the Iteration End

Repeat the steps from (2) to (6) until the convergent condition \( \max(|u_{(k+1)}^n - u_{(k)}^n|, |v_{(k+1)}^n - v_{(k)}^n|) < \varepsilon \) is met. After the iterations are ended, we acquire a series of points sorted by their powers. The maximum power \( P_i \) is a candidate for the final result, which can be expressed as,

\[
P_i = \max[P(u_{(k+1)}^1, v_{(k+1)}^1), P(u_{(k+1)}^2, v_{(k+1)}^2), \ldots, P(u_{(k+1)}^n, v_{(k+1)}^n)].
\] (20)

2.3.6. Edge Pattern

The sidelobes may appear at the extreme points of \( P(u, v) \) as well as at the edge of the visible area. The side lobes on the edge of the visible area may not be caught by the iterations. To solve this problem, we need to calculate the array pattern of the edge area \((u_{edge}^2 + v_{edge}^2 = 1)\) and obtain \( P_e \) as,

\[
P_e = \max[P(u_{edge}^1, v_{edge}^1), P(u_{edge}^2, v_{edge}^2), \ldots, P(u_{edge}^n, v_{edge}^n)].
\] (21)

2.3.7. Final Result

The peak sidelobe level can be expressed as \( \text{PSLL} = \max(P_y, P_e) \).
2.4. Analysis of Computational Complexity

Due to the linear structure of (11), the time computational complexity of the iterative sidelobe-seeking method is $O(Nn_{s1}^2)$, where $N$ is the total element number and $n_{s1}$ is the sampling number in (14).

The conventional method of SLL calculation usually contains three steps:
1. Sampling the $u-v$ area. Usually, the sampling method for the conventional method is rectangular sampling, which can take the form of (14).
2. Computing the array factor of the sampling points using (1) and forming the array pattern.
3. Searching and determining the peak sidelobe among the array pattern.

When the sampling number is $n_{s2}$, the conventional time computational complexity is $O(Nn_{s2}^2)$. For larger arrays, $n_{s2}$ is at least 200, while $n_{s1}$ only needs to be 50 or less. Therefore, the proposed method can systematically reduce the SLL calculation time.

3. Numeric Experiments and Results

This section describes numeric experiments performed to verify the effectiveness of the iterative sidelobe-seeking method in the PSLL calculation. The computing procedure was mainly performed in MATLAB on a computer equipped with a CPU of 3.0 GHz and RAM of 8 GB.

3.1. Sidelobe Calculation for Square Aperture Arrays

This example involved checking the performance of the iterative sidelobe-seeking method in the case of large planar arrays with elements of 400, 900, 1600, and 2500. The array element positions were generated based on $20 \times 20$, $30 \times 30$, $40 \times 40$ and $50 \times 50$ uniform arrays with an element distance of 0.5 wavelength ($\lambda$) (the circles in a light color in Figure 3a). The example was conducted for 100 runs. To increase the complexity of our example, the array element positions in both $x$ and $y$ directions involved random variation of $\pm 0.3 \lambda$ in each run (the circles in a dark color in Figure 3a). Figure 3b is a sample of the array element distribution. The amplitude of each element conformed to $U(0, 1)$. The phase shifter bit value was 3. The errors of the phase were $\delta_n \sim U(0, \frac{\pi}{4})$. In addition, the beam point angles were also random, where $\theta_0 \sim U(0, \frac{\pi}{4})$ and $\phi_0 \sim U(0, 2\pi)$.

![Figure 3](image)

Figure 3. Generation and example of array distribution in the numeric experiments. (a) The generation of the random positions of the array. (b) A sample distribution of an array of 1600 elements.

For the proposed method, we used rectangular sampling and set $N_s$ as 44, 46, 48, and 50 to study the effect on performance. $\gamma$ was set to 17. $L$ was set to 0.1 (we tried a series of $L$ values and 0.1 showed the best performance). When the number of iterations was
four, the iterations were considered to be convergent. As a comparison, we also applied the
conventional method using the sampling strategy of (14). Three kinds of sampling numbers,
200, 400, and 1000 were adopted. We considered the result of the sampling number of 1000
to be the accurate value. We also applied the fast method in [18] for comparison.

The results are provided in Tables 1 and 2. The sampling numbers for the proposed
method had a direct influence on both the accuracy and the computing time. As the
sampling numbers \(N_s\) increased, the average errors decreased and the computing time
increased. For a desired accuracy, the array with more elements required a higher sam-
pling number. In addition, the computing time increased along with the number of array
elements.

Table 1. Comparison of sidelobe absolute errors (dB) with different methods.

| Array Size | 400   | 900   | 1600  | 2500  |
|------------|-------|-------|-------|-------|
| Proposed (\(N_s\) 44) | 0.013 | 0.022 | 0.064 | 0.052 |
| Proposed (\(N_s\) 46) | 0.003 | 0.022 | 0.035 | 0.046 |
| Proposed (\(N_s\) 48) | 0.002 | 0.016 | 0.023 | 0.049 |
| Proposed (\(N_s\) 50) | 0.001 | 0.012 | 0.010 | 0.035 |
| Conventional (\(N_s\) 400) | 0.009 | 0.023 | 0.040 | 0.061 |
| Conventional (\(N_s\) 200) | 0.045 | 0.089 | 0.167 | 0.189 |
| Fast method in [18] | 0.003 | 0.033 | 0.039 | 0.061 |

Table 2. Comparison of computing time (s) with different methods.

| Array Size | 400 | 900 | 1600 | 2500 |
|------------|-----|-----|------|------|
| Proposed (\(N_s\) 44) | 0.124 | 0.164 | 0.257 | 0.368 |
| Proposed (\(N_s\) 46) | 0.130 | 0.189 | 0.271 | 0.385 |
| Proposed (\(N_s\) 48) | 0.144 | 0.202 | 0.285 | 0.405 |
| Proposed (\(N_s\) 50) | 0.147 | 0.216 | 0.307 | 0.457 |
| Conventional (\(N_s\) 400) | 1.106 | 1.824 | 2.850 | 4.184 |
| Conventional (\(N_s\) 200) | 0.277 | 0.459 | 0.718 | 1.050 |
| Fast method in [18] | 0.283 | 0.462 | 0.709 | 1.100 |

The iterative sidelobe-seeking method had an advantage over the conventional method.
The proposed method with \(N_s\) of 50 obtained better accuracy than the conventional method
of 400-sampling, but only consumed 11% computing time in the array of 2500 elements.
When compared to the conventional method of 200-sampling, the proposed method re-
resulted in an 18% average error and consumed 42% computing time.

The proposed method performed better than the method described in [18]. Although
the method in [18] exhibited good accuracy, the computing time was similar to that of
the conventional method of 200-sampling. This was because the input estimated sidelobe
positions were calculated by the conventional method of 200-sampling.

Finally, we also compared the performance between rectangular sampling and random
sampling in the array of 1600 elements. As shown in Table 3, the rectangular sampling of
1961 produced the best result among all the experiments. In addition, the experiments in
which there was a small number of samples consumed less computing time, which means
that random sampling provided no advantage for improving efficiency.
| Sampling Type | Sampling Number | Mean Error (dB) | Time (s) |
|---------------|-----------------|-----------------|----------|
| Random        | 2049            | 0.0188          | 0.3094   |
| Random        | 1984            | 0.0214          | 0.3006   |
| Random        | 1746            | 0.0311          | 0.2725   |
| Random        | 1596            | 0.0397          | 0.2610   |
| Random        | 1442            | 0.0483          | 0.2516   |
| Rectangular   | 1961 (N<sub>s</sub> 50) | 0.0095          | 0.3061   |
| Rectangular   | 1793 (N<sub>s</sub> 48) | 0.0231          | 0.2835   |
| Rectangular   | 1576 (N<sub>s</sub> 46) | 0.0265          | 0.2707   |

### 3.2. Sidelobe Calculation for Circular Array with Taylor Weighting

To verify the effectiveness of the iterative sidelobe-seeking method with a larger amplitude of variation, this example considered a Taylor-weighted circular array with an aperture size of 12 $\lambda$. The array adopted a rectangular grid with 0.5 $\lambda$. The total element number was 448. The example was also conducted for 100 runs. Complying with the Taylor distribution of $-30$ dB, the amplitude of each element had a random error of 30%. Except for the amplitude, the array element positions, the phase errors, and the beam point angles all exhibited random variations similar to the first example. Figure 4a shows a random sample of the array distribution.

In this example, $\gamma$ was set to 30 to meet the more drastic sidelobe level change. Similarly, the conventional method of $N_s$-1000 sampling was responsible for the accurate value.

![Figure 4a](image1)

![Figure 4b](image2)

**Figure 4.** (a) A sample distribution of array elements for Section 3.2. (b) Computing time comparison of example Section 3.2.

Figures 4b and 5 are the results of the proposed method comparing the $N_s$-200 and $N_s$-400 samplings. The iterative sidelobe-seeking method is represented as ‘Proposed’. Figure 5a shows the absolute errors of each method compared to the accurate value. To better analyze the error distribution, Figure 5b shows the boxplot graph for the three methods. The statistical results indicate that our formula was more accurate than $N_s$-200 and close to the $N_s$-400. As shown in Figure 4b, the proposed method saved 54% and 89% algorithm time compared to the $N_s$-200 and $N_s$-400 samplings.
Based on the above two examples, a comparison between previous studies and the proposed method is presented in Table 4.

Table 4. The comparison of the proposed method with other methods.

|                  | Conventional Method | Fast Formula Method | Proposed Method |
|------------------|---------------------|---------------------|-----------------|
| Time Complexity   | O (mn)              | O (n)               | O (mn)          |
| Computing Time    | slow                | fast                | middle          |
| Scope of Application | any planar arrays with given positions | any planar arrays |
| Accuracy          | high                | middle              | high            |

3.3. Synthesis of Sparse Planar Arrays

In this experiment, we applied the iterative sidelobe-seeking method in the synthesis of sparse planar arrays using the modified real genetic algorithm [6]. The array aperture was symmetrical in a rectangle of $9.5\lambda \times 4.5\lambda$. The minimum element spacing was $0.5\lambda$. The number of sparse elements was 100. The fitness function was the PSLL. We used the proposed method and the conventional method in the calculations of fitness for comparison. $\gamma$ was 15 in the proposed method. $N_s$ was 400 in the conventional method. Each method was conducted for 100 runs. In the genetic algorithm, both the generation number and the population size were 100.

Figure 6 presents the results of the two methods. As shown in Figure 6a, the average SLL of the proposed method was $-18.31$ dB, while the average SLL of the conventional method was $-18.50$ dB. The two methods produced similar results. As shown in Figure 6b, our method consumed only 20.1% of the computing time of the conventional method. In general, by replacing the conventional method with the proposed method, we were able to achieve similar results with less time in practical applications.
Figure 6. Comparison of the two methods in the synthesis of sparse arrays. (a) Comparison of optimized SLLs. (b) Comparison of optimization time.

4. Conclusions

In this paper, an iterative sidelobe-seeking method for PSLL calculation was proposed for planar phased arrays involving converting SLL calculation to the zero-point problem of a radiation pattern derivative function. Applying Newton’s iterations, the proposed method approximates the simple sampling positions to the actual sidelobe positions, which reduces the sampling number of the pattern and overcomes the need for initial estimation. The peak sidelobe position and level are obtained after the processing of an iterative sidelobe-seeking framework. The proposed method was verified using numerical examples and possessed better accuracy and efficiency in comparison with conventional methods and the fast method described in [18]. In addition, we tested the performance of the proposed method in practical applications. With the effect of optimization, the proposed method was able to save 80% algorithm time in the synthesis of a sparse planar array. The proposed method is applicable not only for phased arrays but also for all kinds of planar arrays, and, thus, could improve the efficiency of array design for radar and communication.

Author Contributions: Conceptualization, K.M. and H.S.; methodology, K.M. and Y.Z.; formal analysis, S.W. and H.J.; investigation, S.W.; resources, H.J.; writing—original draft preparation, K.M.; writing—review and editing, K.M. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 62001025.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Brookner, E. Advances and breakthroughs in radars and phased-arrays. In Proceedings of the 2016 CIE International Conference on Radar (RADAR), Guangzhou, China, 10–13 October 2016; pp. 1–9. https://doi.org/10.1109/RADAR.2016.8059284.
2. Warnick, K.F.; Maaskant, R.; Ivashina, M.V.; Davidson, D.B.; Jeffs, B.D. Phased Arrays for Radio Astronomy, Remote Sensing, and Satellite Communications; Cambridge University Press: Cambridge, UK, 2018.
3. Lopezf, J.; Tsay, J.; Guzman, B.; Mayeda, J.; Lie, D. Phased arrays in wireless power transfer. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS); IEEE: Boston, MA, USA, 2017; pp. 5–8. https://doi.org/10.1109/MWSCAS.2017.8052846.
4. Mailloux, R.J. Phased Array Antenna Handbook, 3rd ed.; Artech House: Norwood, MA, USA, 2018.
5. Balanis, C.A. Antenna Theory: Analysis and Design, 4th ed.; WILEY: Hoboken, NJ, USA, 2015.
6. Chen, K.; Yun, X.; He, Z.; Han, C. Synthesis of Sparse Planar Arrays Using Modified Real Genetic Algorithm. *IEEE Trans. Antennas Propagat.* 2007, 55, 1067–1073. https://doi.org/10.1109/TAP.2007.893375.

7. Rocha-Alicano, C.; Covarrubias-Rosales, D.; Brizuela-Rodriguez, C.; Panduro-Mendoza, M. Differential evolution algorithm applied to sidelobe level reduction on a planar array. *AEU-Int. J. Electron. Commun.* 2007, 61, 286–290.

8. Chen, L.; Yu, Z.L.; Ser, W.; Cen, W. Linear Aperiodic Array Synthesis Using an Improved Genetic Algorithm. *IEEE Trans. Antennas Propagat.* 2012, 60, 895–902. https://doi.org/10.1109/TAP.2011.2173111.

9. Pappula, L.; Ghosh, D. Linear Antenna Array Synthesis using Cat Swarm Optimization. *AEU-Int. J. Electron. Commun.* 2014, 68, 540–549. https://doi.org/10.1016/j.aeue.2013.12.012.

10. Liu, H.; Zhao, H.; Li, W.; Liu, B. Synthesis of Sparse Planar Arrays Using Matrix Mapping and Differential Evolution. *IEEE Antennas Wireless Propagat. Lett.* 2016, 15, 1905–1908. https://doi.org/10.1109/LAWP.2016.2542882.

11. Pinchera, D.; Migliore, M.D.; Panariello, G. Synthesis of Large Sparse Arrays Using IDEA (Inflating-Deflating Exploration Algorithm). *IEEE Trans. Antennas Propagat.* 2018, 66, 4658–4668. https://doi.org/10.1109/TAP.2018.2846777.

12. Anselmi, N.; Rocca, P.; Salucci, M.; Massa, A. Irregular Phased Array Tiling by Means of Analytic Schemata-Driven Optimization. *IEEE Trans. Antennas Propagat.* 2017, 65, 4495–4510. https://doi.org/10.1109/TAP.2017.2722539.

13. Dong, W.; Xu, Z.H.; Liu, X.; Wang, L.; Xiao, S.P. Modular Subarrayed Phased Array Design by Means of Iterative Convex Relaxation Optimization. *IEEE Antennas Wireless Propagat. Lett.* 2019, 18, 1–1. https://doi.org/10.1109/LAWP.2019.2893946.

14. Yang, K.; Wang, Y.; Tang, H. A Subarray Design Method for Low Sidelobe Levels. *Prog. Electromagn. Res. Lett.* 2020, 89, 45–51. https://doi.org/10.2528/PIERL19110301.

15. Hwang, Y.; Han, C. The use of GTD in the design and analysis of low sidelobe reflector antennas. *Proc. Antennas Propag. Soc. Int. Symp.* 1981, 19, 484–487. https://doi.org/10.1109/APS.1981.1148470.

16. Odendaal, J.; Pistorius, C. Horn Antenna with Suppressed Sidelobe Level. *Microw. Opt. Technol. Lett.* 1991, 4, 238–239. https://doi.org/10.1002/mop.4650040610.

17. Rocca, P.; Mailloux, R.J.; Tos, G. GA-Based Optimization of Irregular Subarray Layouts for Wideband Phased Arrays Design. *IEEE Antennas Wireless Propagat. Lett.* 2015, 14, 131–134. https://doi.org/10.1109/LAWP.2014.2356855.

18. Miao, K.; Zhang, Y.; Sun, H. Fast Sidelobe Calculation and Suppression in Arbitrary Two-Dimensional Phased Arrays. *IEEE Antennas Wireless Propagat. Lett.* 2021, 20, 928–932. https://doi.org/10.1109/LAWP.2021.3067380.

19. Sauer, T. *Numerical Analysis*; Pearson Education: Boston, MA, USA, 2011.