Phase-Only Beam Broadening of Contiguous Uniform Subarrayed Arrays

BARRY K. DANIEL, Senior Member, IEEE
National Security Sciences Directorate, Oak Ridge National Laboratory, Oak Ridge, TN, USA

ADAM L. ANDERSON, Senior Member, IEEE
MRSL Real-Time Systems Lab, Monterey, CA, USA

In modern antenna systems, beam broadening of subarrayed arrays provides continuous coverage of a wide angular extent in a cost-effective manner. While many methods have been published that address beam broadening of traditional (non-subarrayed) arrays, there is a knowledge gap in the published literature with respect to efficient beam broadening of contiguous uniform subarrayed arrays. This article presents efficient methods for beam broadening of contiguous uniform subarrayed arrays where the excitation of each element is not individually controlled, but the elements of the array are grouped together as subarrays to have the same element excitations. Particularly, this article focuses on phase-only optimization to preserve maximum power output. Three modified iterative Fourier transform (IFT) methods and one genetic algorithm (GA) are presented to efficiently search the vast solution space of possible phase settings for a solution that satisfies the desired broadened pattern. These methods are evaluated on idealized $1 \times 40$ and $1 \times 80$ linear arrays with five element subarrays and $40 \times 40$ and $80 \times 80$ element rectangular arrays with $5 \times 5$ element subarrays. The proposed modified IFT methods are found to be faster than the GA approach, while the GA approach only offers a few percentage points of better effectiveness.

Manuscript received May 27, 2019; revised September 13, 2019, December 12, 2019, and March 18, 2020; released for publication March 21, 2020. Date of publication April 16, 2020; date of current version October 9, 2020.

DOI. No. 10.1109/TAES.2020.2987115

Refereeing of this contribution was handled by L. Rosenberg.

This work was supported by the US Department of Energy, Office of Science, Office of Basic Energy Sciences under Contract DE-AC05-00OR22725.

Authors’ addresses: Barry K. Daniel is with the National Security Sciences Directorate, Oak Ridge National Laboratory, Oak Ridge, TN 37831 USA E-mail: (danielbk@ornl.gov), Adam L. Anderson is with MRSL Real-Time Systems Lab, Monterey, CA 93933 USA, E-mail: (adam.anderson@mrsl.com). (Corresponding author: Barry K. Daniel.)

0018-9251 © 2020 IEEE.

1. INTRODUCTION

Many of the instrumentation radars being delivered to military test centers around the world are designed with phase-only adjustable contiguous uniform subarrayed arrays to reduce cost while giving acceptable performance. These instrumentation radars utilize beam broadening for continuous measurement of objects over wide areas to increase the probability that unexpected phenomenology produced from prototype projectiles can be captured and characterized. Therefore, the generation of efficient and effective broadened beam patterns for contiguous uniform subarrayed arrays is essential and is the focus of this paper.

The architecture of a contiguous uniform subarrayed array reduces cost by limiting the number of control elements but introduces problems with the generation of broadened beam patterns because the elements are grouped together in a manner that only allows the amplitude and phase of each subarray, not each element, to be adjusted independently. Only allowing excitation adjustments at the subarray level makes it more difficult to form a low-ripple broadened main beam and to reduce the sidelobes. Subarrayed architectures also produce grating lobes which limit the effective electronic scanning of these types of arrays. However, currently delivered subarrayed instrumentation radars utilize mechanical movement to steer the widened beams instead of electronic steering.

These delivered instrumentation radars utilize solid-state transmit amplifiers which operate most efficiently in saturation (unity gain), it is often desired to utilize phase-only approaches to beam broadening as presented in [1]–[3] to optimize power efficiency. However, these published methods of beam broadening assume that the phase of each element in the array can be independently adjusted which is not true of contiguous uniform subarrayed arrays.

There is currently a gap in the available literature regarding efficient methods to broaden the main beam of subarrayed arrays when the synthesis is constrained to be phase-only, utilize contiguous uniform subarrays, and produce low sidelobes. Papers such as [4] and [5] address synthesis with subarrayed architectures but they concentrate on optimization of combined sum and difference patterns, not broadening the main beam. The author in [6] proposes a synthesis method for subarrayed arrays but this method is amplitude only, utilizes subarrays of unequal sizes, and does not address beam broadening. The papers [7] and [8] approach our desired domain space but utilize hybrid subarrays where the amplitude is dependent on the subarray amplitude but the phase of each element in the subarray can still be adjusted independently. To the best of our knowledge, the closest published material found is [9] where a beam broadening approach for subarrayed architectures is constrained to be phase-only and utilize contiguous subarrays. Unfortunately, this method constrains the widened full array beam widths to only those beam widths that are multiples of the subarray beam width often producing unacceptable sidelobe levels.
It is well known that the Fourier transform (FT) can be used to compute the array factor (AF) of a phased array antenna with uniformly spaced antenna elements [10] where the AF is defined as the radiation pattern of the array minus the effects of the individual element patterns. A single application of a simple transform method such as FT cannot be used effectively for phase-only synthesis applications because simple transform methods assume that the amplitude and phase of the elements or subarrays can be adjusted. Therefore, iterative processes using transform methods such as FT were developed. Early works, such as [11] and [12], show the benefits of utilizing iterative processes based on the FT to synthesize beam broadened array patterns. In [13], iterative use of the FT and the inverse FT was shown to enable generation of AFs with low sidelobes and directional nulls. In [14] and [15], this iterative process was improved by detailing the usage of the more computationally efficient fast Fourier Transform (FFT) and the inverse fast Fourier transform (IFFT). However, these published IFT approaches assume that the phase of each element can be independently adjusted which is not true for subarrayed arrays.

This article contributes to the currently published literature by presenting new efficient beam broadening methods for contiguous uniform subarrayed arrays by proposing three enhanced IFT methods and adapting the GA method for subarrayed arrays. The proposed IFT methods extend previously published IFT methods by accounting for contiguous uniform subarrayed architectures through the use of mean pooling techniques to aggregate the elements into subarrays and to ensure symmetry in the array pattern. Likewise, the presented GA method has been adapted for contiguous uniform subarrayed arrays. The GA method was selected for adaptation because it has been commonly utilized in previous non-subarrayed array synthesis literature such as [16], this is the first time to the authors’ knowledge that it has been adapted for contiguous uniform subarrayed arrays. The proposed methods are applied to linear subarrayed arrays and are then expanded for use in planar subarrayed arrays. Simulation results for the beam broadening and shaping of the main beam for linear arrays containing eight and sixteen subarrays, each containing five elements, are given to demonstrate the validity and efficiency of the proposed method. Simulation results for planar arrays containing $8 \times 8$ and $16 \times 16$ subarrays, each containing $5 \times 5$ elements, are also presented.

The mathematical foundation for synthesis is presented in Section II and is followed by Section III which presents the three enhanced IFT methods and the GA method that are modified to account for subarrayed architectures. Section IV shows the simulated results of the proposed methods for linear and planar subarrayed arrays and compares the efficiency and effectiveness of each method. Finally, Section V concludes this article.

II. MATHEMATICAL FOUNDATION

It is well known that the far-field array pattern of a linear array of $N$ elements each spaced a distance $d$ apart can be written as the product of the element factor, $EF$, and the array factor $AF$

$$ F(u) = EF(u)AF(u) $$

where $u = \sin \theta$, and $\theta$ is the angle relative to the array normal. For isotropic elements, $EF(u) = 1$; therefore, the array pattern, $F(u)$, depends solely on the array factor which can be represented as

$$ AF(u) = \sum_{n=0}^{N-1} W_n e^{jkd_nu} $$

where $N$ is the total number of elements in the linear array, $W_n$ is the complex excitation of the $n$th element, $k$ is the wavenumber $(2\pi / \lambda)$, and $\lambda$ is the wavelength. The complex excitation, $W_n$, for a normal phased array where the amplitude and phase of each element can be independently adjusted can be written as

$$ W_n = A_n e^{j\phi_n} $$

where $A_n$ is the amplitude and $\phi_n$ is the phase of the element excitation.

Let us consider a subarrayed linear array where these $N$ elements are separated into contiguous groups of $N_s$ elements to form $S$ subarrays. Each subarray has a complex excitation consisting of an amplitude and phase value that drives all elements in the subarray, as visualized in Fig. 1. For subarrayed architectures (2) can be rewritten as

$$ AF(u) = \sum_{s=0}^{S-1} \sum_{n_s=0}^{N_s-1} W_{N_s s+n_s} e^{jkd_{N_s s+n_s}u} $$

where the complex excitation variable $W_{N_s s+n_s}$ is the complex excitation applied to each subarray. For instance, in Fig. 1, the first subarray consists of the first four elements of the array. Since this first subarray is given a complex excitation of $A_0 e^{j\phi_0}$, the excitation of each of the first four elements is identical, $W_0 = W_1 = W_2 = W_3 = A_0 e^{j\phi_0}$. 

![Fig. 1. Subarrayed architecture geometry. This figure shows an example of a contiguous uniform subarrayed linear array antenna with $N = 12$ antenna elements, $N_s = 4$ elements per subarray, and $S = 3$ subarrays.](image-url)
The response of each subarray is given by

\[ \text{AF}_s(u) = \sum_{n_s=0}^{N_s-1} W_{N_s+n_s} e^{j k d n_s u} \]  

(5)

\[ \text{AF}(u) = \sum_{s=0}^{S-1} \text{AF}_s(u) e^{j k d N_s u}. \]  

(6)

As mentioned above, the excitation of each element in the subarray is identical; so the excitation variable \( W_{N_s} \) is a constant within each subarray and can be pulled outside of the summation as a variable only dependent on the subarray number as

\[ \text{AF}_s(u) = W_{N_s} \sum_{n_s=0}^{N_s-1} e^{j k d n_s u}. \]  

(7)

Now, if we recognize that the summation in (7) is the common pattern for the structure of each subarray in the uniform contiguous subarrayed array, we can reduce (7) to

\[ \text{AF}_s(u) = W_{N_s} B(u) \]  

(8)

where \( B(u) \) represents the common pattern for the structure of each subarray.

Substituting (8) into (4), the resulting array factor for the subarrayed architecture is

\[ \text{AF}(u) = B(u) \sum_{s=0}^{S-1} W_{N_s} e^{j k d N_s u}. \]  

(9)

Notice that the summation in (9) is using subarray spaced \((d N_s)\) elements to represent the substructure of the positioning of the subarrays in the full array and that \( B(u) \) is a constant outside of the summation. Using this information, (9) can be reduced further to

\[ \text{AF}_{ss}(u) = \text{AF}(u)/B(u) = \sum_{s=0}^{S-1} W_{N_s} e^{j k d N_s u}. \]  

(10)

Expanding the scope of the mathematical model for use in planar subarrayed arrays, (2) can be extended to apply to 2-D arrays with \( N \times M \) elements arranged in a rectangular grid with spacing \( d_x \) and \( d_y \) as

\[ \text{AF}(u, v) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} W_{n,m} e^{j k d x n u + j k d y m v} \]  

(11)

where \( W_{n,m} \) is the complex excitation of the element in the \( n \)th column and \( m \)th row, \( k \) is the wavenumber \((2\pi/\lambda)\), \( \lambda \) is the wavelength, \( d_x \) is the separation between the columns of elements, \( d_y \) is the separation between the rows of elements, \( u = \sin \theta \cos \phi \), and \( v = \sin \theta \sin \phi \).

Similarly, (4) can be extended to represent planar subarrayed architectures as

\[ \text{AF}(u, v) = \sum_{s=0}^{S-1} \sum_{t=0}^{T-1} \sum_{n=0}^{N_s-1} \sum_{m=0}^{M_t-1} W_{x,y} e^{j k d x u + j k d y v} \]  

(12)

where \( x = N_s s + n_s \) and \( y = M_t t + m_t \).

Likewise, (10) can be extended to represent planar subarray substructures as

\[ \text{AF}_{ss}(u, v) = \text{AF}(u, v)/B(u, v) = \sum_{s=0}^{S-1} \sum_{t=0}^{T-1} W_{g,h} e^{j k d x u + j k d y v} \]  

where \( g = N_s s \) and \( h = M_t t \).

A. General IFT Method

Close inspection of (2) reveals that this equation can be represented as a finite Fourier series that relates the element-level excitations, \( W_n \), of the array to the array factor of the array through a discrete inverse Fourier transform. Since AF is related to the element excitations through a discrete inverse Fourier transform, a discrete direct Fourier transform applied to AF will yield the element excitations. The general IFT method utilizes these Fourier transform relationships between AF and \( W_n \) to iteratively find \( W_n \) that will generate a desired AF, \( \text{AF}_{des} \). The general process is initiated with all \( W_n \) set to unity amplitude and zero phase. Then, the process iteratively generates the AF associated with the current \( W_n \), compares its to the \( \text{AF}_{des} \), adjusts the AF to be more like the \( \text{AF}_{des} \), and then generates new \( W_n \) from the adjusted AF. This iterative process is repeated for a maximum number of iterations, \( l_{max} \).

For computational efficiency, the IFT process makes use of the IFFT and FFT operations. When \( W_n \) is used in the IFFT operation to generate the AF, it is zero padded to become \( W_{zp} \) where the size of \( W_{zp} \) is the nearest power of two greater than \( 10 \times N \). After the FFT process is used on the AF, \( W_{zp} \) is truncated to find \( W_n \). A detailed step-by-step process of the general IFT approach can be found in [15].

B. Cost Function

In this article, a cost function determines how close the generated array factor, \( \text{AF}_{gen} \), is to the desired array factor, \( \text{AF}_{des} \). The desired array factor is defined as a normalized flat-top function with a main beam and sidelobes. The main beam consists of a desired angular width with an amplitude of 0 dB. The sidelobes are the points in \( \text{AF}_{des} \) outside this main beam with a desired sidelobe level defined in negative dB. \( \text{AF}_{gen} \) is computed by taking the IFFT of the element excitations (2), and is normalized before comparing it to \( \text{AF}_{des} \). The cost function, \( C \), is defined as

\[ C = \log(E_{mb} + E_{sl}) \]  

(14)

where \( E_{mb} \) is the error calculated for the points that are in the main beam and below 0 dB, \( \theta_{mb} \), and is defined as

\[ E_{mb} = \sum_{\theta_{mb}} ((|\text{AF}_{des}(\theta_{mb})| - |\text{AF}_{gen}(\theta_{mb})|)^2 \]  

(15)

and \( E_{sl} \) is the error calculated for the points that are outside the main beam and above the desired sidelobe level, \( \theta_{sl} \), and is defined as

\[ E_{sl} = \sum_{\theta_{sl}} ((|\text{AF}_{gen}(\theta_{sl})| - |\text{AF}_{des}(\theta_{sl})|)^2 \]  

(16)
The logarithm is used in (14) to reduce the values returned to a more condensed range so that the convergence rate of the cost function can be more easily visualized. For a more tangible performance measure, (14) can be converted to a percentage pattern effectiveness metric as

$$P_{eff} = 10^{-\sqrt{\exp(C)/(\beta \times 100)}} \times 100$$

where $\beta$ represents the number of angular points used to evaluate $AF_{gen}$ vs. $AF_{des}$.

### III. PROPOSED METHODS

This article presents new efficient beam broadening methods for contiguous uniform subarrayed arrays by proposing three speed enhanced IFT methods that modify the general IFT approach [15] to account for contiguous uniform subarrayed architectures and to broaden the main beam. This article also presents a GA method that has been altered for use with contiguous uniform subarrayed architectures. The previously presented (2) assumes that the amplitude and phase of each individual array element can be independently adjusted. This assumption does not hold true for subarrayed array architectures where the amplitude and phase adjustments are applied to groups of array elements.

#### A. Integrated IFT (iIFT)

The first proposed modified IFT algorithm, herein referred to as the integrated IFT, or iIFT algorithm, enhances the general IFT method by using a “mean pooling” strategy to group the elements into subarrays and assign each element in the subarray the average phase value of the elements in the subarray. The average phase value is calculated using the complex number form of the element excitations to account for the phase being a circular function. These averaged values are used as the new element excitations and the next AF in the iterative process is generated using these element excitations. The iIFT process is shown in block diagram form in Fig. 2. The detailed process for the iIFT method is shown in Algorithm 1.

#### B. Decoupled IFT (dIFT)

The second proposed modified IFT algorithm, herein referred to as the decoupled IFT, or dIFT algorithm, creates an additional set of element excitations, $V_n$, to decouple the grouping of the elements into subarrays from the iterative process. After the element excitations, $W_n$, are found via the IFT process for the next iteration, they are grouped into subarrays to form a new set of element excitations, $V_n$. $V_n$ is then used to generate a subarray influenced array factor, $AF_{gen}$, which is compared to $AF_{des}$ using (14). The dIFT process is shown in block diagram form in Fig. 3. The detailed process for the dIFT method is shown in Algorithm 2.

---

**Algorithm 1: Integrated Iterative Fourier Transform (iIFT).**

1:  $i \leftarrow 0$  \hspace{1cm} $\triangleright$ Initialize element excitations  
2:  $W(i) \leftarrow 1$  \hspace{1cm} $\triangleright$ Initialize the cost value  
3:  $C \leftarrow 1 \times 10^{30}$ \hspace{1cm} $\triangleright$ Normalized desired pattern  
4:  Define $AF_{des}$ \hspace{1cm} $\triangleright$ Generate the pattern  
5:  while $i < l_{max}$ do  
6:    $W_{zp} \leftarrow W(i) +$ zero padding  
7:    $AF_{gen}(i) \leftarrow$ IFFT($W_{zp}$) \hspace{1cm} $\triangleright$ Generate new excitations  
8:    Compute $C_{new}$ using (14)  
9:    if $C_{new} < C$ then  
10:      $C \leftarrow C_{new}$  
11:      $i_{best} \leftarrow i$  \hspace{1cm} $\triangleright$ Save index of best pattern  
12:    end if  
13:    $|AF_{gen}(i, \theta_{sl})| \leftarrow |AF_{des}(\theta_{sl})|$  
14:    $|AF_{gen}(i, \theta_{sl})| \leftarrow |AF_{des}(\theta_{sl})|$  
15:    $W_{zp} \leftarrow$ FFT($AF_{gen}(i)$) \hspace{1cm} $\triangleright$ Generate new excitations  
16:  end while  
17:  for $s = 0 : S$ do \hspace{1cm} $\triangleright$ Adjust subarray excitations  
18:    Calculate element $\phi_{mean}$ of equidistant subarrays.  
19:    for $n_s = 0 : N_s$ do \hspace{1cm} $\triangleright$ Set element excitations  
20:      $|W(N_s,N_s+N_s(i+1))| \leftarrow 1$  
21:      $\angle W(N_s,N_s+N_s(i+1)) \leftarrow \phi_{mean}$  
22:    end for  
23:  end for  
24:  $i \leftarrow i + 1$  
25:  end while  
26:  $W_{best} \leftarrow W(i_{best})$  
27:  $AF_{best} \leftarrow AF_{gen}(i_{best})$
Algorithm 2: Decoupled Iterative Fourier Transform (dIFT).

1: \( i \leftarrow 0 \)  \( \triangleright \) Initialize element excitations
2: \( W(i) \leftarrow 1 \)  \( \triangleright \) Initialize the cost value
3: \( C \leftarrow 1 \times 10^{30} \)  \( \triangleright \) Initialize the cost value
4: Define \( AF_{\text{des}} \)  \( \triangleright \) Normalized desired pattern
5: while \( i < i_{\text{max}} \) do
6: \( W_{zp} \leftarrow W(i)+ \) zero padding
7: \( AF_{\text{gen}}(i) \leftarrow \text{IFFT}[W_{zp}] \)  \( \triangleright \) Generate the pattern
8: \( |AF_{\text{gen}}(i, \theta_{\text{sub}})| \leftarrow |AF_{\text{des}}(\theta_{\text{sub}})| \)
9: \( |AF_{\text{gen}}(i, \theta_{s})| \leftarrow |AF_{\text{des}}(\theta_{s})| \)
10: \( W_{zp} \leftarrow \text{FFT}[AF_{\text{gen}}(i)] \)
11: \( W(i+1) \leftarrow W_{zp}(1:N) \)  \( \triangleright \) Generate new excitations
12: for \( s = 0 : S \) do  \( \triangleright \) Adjust subarray excitations
13: Calculate element \( \phi_{\text{mean}} \) of equidistant subarrays.
14: for \( n_s = 0 : N_s \) do  \( \triangleright \) Set element excitations
15: \( |V_{s}(N_s + n_s)(i+1)| \leftarrow 1 \)
16: \( \angle V_{s}(N_s + n_s)(i+1) \leftarrow \phi_{\text{mean}} \)
17: end for
18: end for
19: \( V_{zp} \leftarrow V(i)+ \) zero padding
20: \( AF_{\text{gen}}(i) \leftarrow \text{IFFT}[V_{zp}] \)
21: Compute \( C_{\text{new}} \) using (14)
22: if \( C_{\text{new}} < C \) then
23: \( C \leftarrow C_{\text{new}} \)
24: \( i_{\text{best}} \leftarrow i \)  \( \triangleright \) Save index of best pattern
25: end if
26: \( i \leftarrow i+1 \)
27: end while
28: \( V_{\text{best}} \leftarrow V(i_{\text{best}}) \)
29: \( AF_{\text{best}} \leftarrow AF_{\text{gen}}(i_{\text{best}}) \)

C. Subarray Spaced IFT (ssIFT)

The third proposed modified IFT algorithm, herein referred to as the subarray spaced IFT, or ssIFT algorithm, enhances the general IFT method by reducing the problem to focus on the subarray substructure pattern instead of focusing on the full array pattern. With the ssIFT algorithm there is some additional complexity added with the transition from the full pattern domain to the subarray substructure domain. From (10), given a desired overall array pattern, the desired overall array pattern can be divided by the subarray pattern which results in the pattern of the phase weights at the center of each subarray referred to herein as the desired subarray structure pattern \( AF_{\text{gen}(\text{sub})} \). This subarray substructure pattern then needs to be converted from element-wise FFT bins to subarray-wise FFT bins. A new number of FFT bins \( M \) is calculated as the nearest power of two greater than \( 10 \times S \). The angular extent of the pattern \( (\theta_{\text{sub}}) \) also needs to be limited to \( \text{arcsin}[M/2, M/2 - 1]/(\lambda/(M d_{\text{sub}})) \) due to the subarray spacing. Then, interpolation can be used to arrive at the desired subarray spaced subarray substructure pattern \( AF_{\text{des}(\text{sub})} \). While there is some added single-use complexity added with this method, the complexity inside the iterative loop can be reduced using this algorithm by reducing the size of the FFT and using a simplified “mean pooling” strategy to ensure that subarrays that are equidistant from the center of the array receive the same excitation to force symmetry in the array pattern. These averaged values are used as the new excitations and the next generated subarray substructure pattern, \( AF_{\text{gen}(\text{sub})} \), in the iterative process is generated using these subarray excitations. The ssIFT process is shown in block diagram form in Fig. 4.

The detailed process for the ssIFT method is shown in Algorithm 3.

D. Genetic Algorithm (GA)

A genetic algorithm (GA) is inspired by the process of natural selection and relies on bio-inspired operators such as mutation, crossover and selection to evolve an initial population of candidate solutions toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered. The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual’s genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. The algorithm terminates when either a maximum number
Algorithm 3: Subarray Spaced Iterative Fourier Transform (ssIFT).

1: Define $AF_{des(ss)}$ $\triangleright$ Full desired pattern (element-spaced)
2: $W(1 : N_c) \leftarrow 1$ $\triangleright$ Initialize subarray element excitations
3: $W_{zp} \leftarrow W(1 : N_c) + \text{zero padding}$
4: $AF_{subarray} \leftarrow \text{IFFT}(W_{zp})$ $\triangleright$ Generate subarray pattern
5: $AF_{des(ss)\backslash elem} \leftarrow AF_{des(ss)}/AF_{subarray}$
6: $\theta_{x} \leftarrow \arcsin[-M/2, M/2 - 1](\lambda/(Md_{sub}))$
7: $AF_{des(ss)\backslash sub} \leftarrow \text{interp}(AF_{des(ss)\backslash elem}, \theta_{x})$
8: $i \leftarrow 0$ $\triangleright$ Set iteration count variable
9: $W_{sub}(i) \leftarrow 1$ $\triangleright$ Initialize element excitations
10: $C \leftarrow 1 \times 10^{30}$ $\triangleright$ Initialize the cost value
11: while $i < i_{\max}$ do
12: $W_{zp} \leftarrow W_{sub}(i) + \text{zero padding}$
13: $AF_{gen(ss)\backslash sub}(i) \leftarrow \text{IFFT}(W_{zp})$ $\triangleright$ Generate pattern
14: Compute $C_{new}$ using (14)
15: if $C_{new} < C$ then
16: $C \leftarrow C_{new}$ $\triangleright$ Update cost value
17: $i_{\text{best}} \leftarrow i$ $\triangleright$ Save index of best pattern
18: end if
19: $|AF_{gen(ss)\backslash sub}(i, \theta_{mb})| \leftarrow |AF_{des(ss)\backslash sub}(\theta_{mb})|$ $\triangleright$ Compute $|AF_{gen(ss)\backslash sub}(i, \theta_{mb})|$ for each $\theta_{mb}$
20: $|AF_{gen(ss)\backslash sub}(i, \theta_{d})| \leftarrow |AF_{des(ss)\backslash sub}(\theta_{d})|$ $\triangleright$ Compute $|AF_{gen(ss)\backslash sub}(i, \theta_{d})|$ for each $\theta_{d}$
21: $W_{zp} \leftarrow \text{FFT}(AF_{gen(ss)\backslash sub}(i))$
22: $W_{sub}(i + 1) \leftarrow W_{zp}(1 : S)$ $\triangleright$ Adjust subarray excitations
23: for $s = 0 : S$ do $\triangleright$ Adjust subarray excitations
24: Calculate element $\phi_{\text{mean}}$ of equidistant subarrays.
25: $\angle W_{sub}(i + 1) \leftarrow 1$
26: $\angle W_{sub}(i + 1) \leftarrow \phi_{\text{mean}}$
27: end for
28: $i \leftarrow i + 1$
29: end while
30: $W_{sub(best)} \leftarrow W_{sub}(i_{\text{best}})$
31: $AF_{gen(ss)\backslash sub\backslash best} \leftarrow AF_{gen(ss)\backslash sub}(i_{\text{best}})$
32: Zero pad subarray excitations to elem spacing ($W_{elem_{best}}$)
33: $AF_{gen(ss)\backslash elem} \leftarrow \text{IFFT}(W_{elem_{best}})$
34: $AF_{gen(ss)\backslash sub} \leftarrow AF_{gen(ss)\backslash elem} \times AF_{subarray}$

of generations has been produced, or a satisfactory fitness level has been reached for the population.

For our beam broadening array pattern optimization problem the population includes candidate sets of element excitations. Therefore, in genetic algorithm terminology, a set of element excitations for each element in the array is the chromosome and each individual element citation is the gene.

The GA process used in this article is derived from the implementation described in [17] but minimizes a cost function instead of maximizing a fitness function. In addition, elite members of the population are introduced due to analysis in [18]. The variables description of the GA process are defined as:

- $U$: A candidate set of element excitations.
- $P$: The population of candidate sets of element excitations.
- $P_{new}$: The new population of candidate sets of element excitations derived from the previous population.
- $C$: A function that assigns an evaluation score to a given candidate set of element excitations.
- $G$: The maximum number of generations.
- $Z$: The number of candidate sets of element excitations to be included in the population.
- $r$: The fraction of the population to be replaced via the crossover technique at each step.
- $m$: The mutation rate.

The implemented GA process can be described as:

1) randomly generate a population, $P$, of size $Z$ candidate sets of subarray excitations, $U$;
2) for each $U$, synthesize the array pattern and evaluate the generated pattern against the desired pattern via a cost function, $C$;
3) for $G$ generations of solutions
   a) create a set of parents by selecting $rZ$ members of $P$ in a probabilistic manner. The probability $Pr(U_{i})$ of selecting candidate set $U_{i}$ from $P$ is given by
   
   $$Pr(U_{i}) = \frac{C_{\text{max}} - C(U_{i})}{\sum_{j=1}^{Z}(C_{\text{max}} - C(U_{j}))}$$
   
   where $C_{\text{max}} = \max(C(U_{1}), C(U_{2}), \ldots, C(U_{Z}))$. If all the values of $C(U_{i})$ are the same, $Pr(U_{i}) = \frac{1}{Z}$.
   b) for each pair of candidate sets from $P$, $(U_{1}, U_{2})$, produce two offspring by applying the crossover operator. Add all offspring to $P_{new}$.
   c) for each of the members of $P_{new}$, perform the following mutation operator from [18]: for each gene, if $\text{rand}() < m$, mutate it.
   d) add the $(1 - r)Z$ members of $P$ with the lowest cost into $P_{new}$ as elites.
   e) update the population: $P \leftarrow P_{new}$.
   f) for each candidate set of subarray excitations, synthesize the array pattern and evaluate the generated pattern against the desired pattern via a cost function, $C$.
4) return the candidate set from $P$ that has the lowest cost function value.

IV. RESULTS

The results presented in this article were generated using a computer simulation that synthesized an approximation of the array pattern using (1) where each element was assumed to be an isotropic radiator thus making $EF$ equal to 1. The $AF$ portion of the array pattern approximation was implemented.
with an FFT-based approach that utilized (2) as its foundation. This simulation contained adjustable parameters that allowed arbitrary arrays to be modeled based on the number of elements in each subarray, the number of subarrays in the array, the spacing between elements, and the number of bits for each phase shifter. The desired antenna pattern was described using parameters for width of the broadened main beam and maximum sidelobe level. The simulation also constrained the generated element/subarray excitations to be symmetric based on their distance from the center of the array to ensure that the synthesized array pattern was symmetric about the center of the array. The iIFT, dIFT, ssIFT, and GA methods are iteratively executed until the maximum number of iterations is achieved.

The simulation was configured to model X-band (10 GHz transmit frequency) linear arrays of 40 and 80 elements and rectangular arrays of $40 \times 40$ and $80 \times 80$ elements. The linear arrays were grouped into contiguous subarrays of five elements while the rectangular arrays were grouped into subarrays of $5 \times 5$ elements. The simulation was configured to synthesize an array pattern that closely approximates a flat-top pattern with a beamwidth of 12 degree and maximum sidelobes of $-13$ dB. To better understand the impact of subarrayed architectures, array patterns were synthesized for arrays with and without subarrays. Since it is the relative phase of the elements and subarrays to each other that is important, the innermost subarrays were fixed to zero phase to reduce the solution space and thus decrease the convergence time.

Since the GA method is not a deterministic process, the GA method was performed 100 times with the same settings to obtain statistical significance. The constants used in the GA algorithm are given in Table I. Using a quarter of the population as elites in the parent selection was determined to improve convergence. Due to the use of elitism preserving genes in the GA, $r$ and $m$ were both set higher to increase the rate of testing new solutions. The selection of population size was based on the rule-of-thumb that population sizes between 20–50 are usually sufficient. For the smaller $40 \times 40$ array with a smaller solution space, a population of size 25 was used. However, in the case of the $80 \times 80$ radar, due to the increased solution space size, a larger population size of 50 was used. The generation size was set to ensure that 20 000 array patterns were generated.

A. A Note on Complexity

The iIFT, dIFT, ssIFT, and GA approaches are compared based on their computational efficiency and their effectiveness. The efficiency of each approach is compared based upon the number of array patterns and/or sets of element excitations that need to be generated per iteration times the number of iterations that were performed. Each iteration of the proposed iIFT and ssIFT methods calculate one array pattern and one set of element excitations for iIFT is twice the number of iterations of the algorithm. The dIFT method creates an additional set of element excitations and generates an array pattern with this additional set for each iteration. The number of generated array patterns and sets of element excitations for dIFT is three times the number of iterations of the algorithm. The GA method calculates an array pattern for each new candidate set of the population at each generation.

To summarize, the computational efficiency is measured by counting the number of array patterns and sets of element excitations that were generated to achieve a desired minimum cost value. The effectiveness of each approach is measured by its ability to achieve the lowest cost value.

After the IFT and GA methods calculate their best solutions, the calculated phase values were constrained to values that a 6-bit phase shifter can utilize. Constraining the phase values during the IFT process was found to overly constrain the solution space and hinder the IFT process. In other words, there is no advantage to enforcing the resolution constraint before running the algorithms.

B. $1 \times 40$ Results

The convergence curves for the synthesis of a 40 element linear array (subarrayed into 8 subarrays of 5 element each) are shown in Fig. 5. The iIFT approach requires 50 generated array patterns/excitations to find its best solution ($C = 6.38$) and 90 generated array patterns/excitations to find its steady-state solution ($C = 6.41$). The dIFT approach requires 78 generated array patterns/excitations to find its best solution ($C = 6.47$) and 114 generated array patterns/excitations to find its steady-state solution ($C = 6.54$). The ssIFT approach requires 30 generated array patterns/excitations to find its best solution ($C = 6.36$) and
Fig. 6. Full (a) and zoomed (b) amplitude patterns for the synthesized 40 element linear array. Note that the patterns are very similar in the main beam region, and the effectiveness is comparable among the iIFT, dIFT, and ssIFT, and GA methods. Also notice that the sidelobes in the visible region are below the desired level.

70 generated array patterns/excitations to find its steady-state solution ($C = 6.39$). In comparison the GA approach on average needs to generate greater than 20,000 array patterns to reach the same best cost function value as the ssIFT approach (3000 array patterns to reach the ssIFT steady-state value), 14,000 array patterns to reach the same best cost function value as the iIFT approach (1500 array patterns to reach the iIFT steady-state value), and 600 array patterns to reach the same best cost function value as the dIFT approach (400 array patterns to reach the dIFT steady-state value). The best (lowest cost, most effective) synthesized amplitude patterns are shown in Fig. 6. The phase excitations for each element in the 40 element array that are associated with the best generated patterns are presented in Fig. 7.

C. $1 \times 80$ Results

The convergence curves for the synthesis of an 80 element linear array (subarrayed into 16 subarrays of 5 element each) are shown in Fig. 8. The iIFT approach requires 120 generated array patterns/excitations to find its best and also steady state solution ($C = 7.21$). The dIFT approach requires 315 generated array patterns/excitations to find its best solution ($C = 7.28$) and 381 generated array patterns/excitations to find its steady state solution ($C = 7.36$). The ssIFT approach requires 100 generated array patterns/excitations to find its best and steady state solution ($C = 7.19$). In comparison the GA approach on average needs to generate 1240 array patterns to reach the same best cost function value as the ssIFT approach, 1195 array patterns to reach the same best cost function value as the iIFT approach, and 1160 array patterns to reach the same best cost function value as the dIFT approach. The best synthesized amplitude patterns are shown in Fig. 9. The phase excitations for each element in the 80 element array that are associated with the best generated patterns are presented in Fig. 10.

D. $40 \times 40$ Results

The convergence curves for the synthesis of a $40 \times 40$ element rectangular array are shown in Fig. 11. The iIFT method requires 50 generated array patterns/excitations to provide its best and steady state solution ($C = 11.52$). The dIFT method requires 60 generated array patterns/excitations to provide its best and steady state solution ($C = 11.59$). The ssIFT method requires 70 generated array patterns/excitations to provide its best and steady state solution ($C = 11.39$). In comparison the GA approach on
average needs to generate 1039 array patterns to reach the same best cost function value as the ssIFT approach, 440 array patterns to reach the same best cost function value as the iIFT approach, and 270 array patterns to reach the same best cost function value as the dIFT approach. The best synthesized amplitude patterns are shown in Figs. 12 and 13. The phase excitations for each element in the 40 × 40 element array that are associated with the best generated patterns are presented in Fig. 14.

E. 80 × 80 Results

The convergence curves for the synthesis of a 80 × 80 element rectangular array are shown in Fig. 15. The iIFT method requires 195 generated array patterns/excitations to provide its best and steady state solution ($C = 12.85$). The dIFT method requires 480 generated array patterns/excitations to provide its best and steady state solution ($C = 12.94$). The ssIFT method requires 360 generated
Fig. 13. Amplitude patterns including the full visible region for the synthesized $40 \times 40$ array. Note that the extended sidelobe patterns are below the desired sidelobe level. (a) iIFT. (b) dIFT. (c) ssIFT. (d) GA. (e) No subarrays IFT.

Fig. 14. Element phase excitations for the synthesized $40 \times 40$ array. (a) iIFT. (b) dIFT. (c) ssIFT. (d) GA. (e) No subarrays IFT.

Fig. 15. Cost convergence curves for the synthesized $80 \times 80$ array. The convergence curve for the non-subarrayed array ($80 \times 80$ IFT Reference) is given as a reference. Note that the iIFT, dIFT, and ssIFT methods converge significantly faster than the GA method. The full synthesis was executed for 20 000 generated array patterns/coefficients.

Fig. 16. Amplitude patterns for the synthesized $80 \times 80$ array. Note that the iIFT, dIFT, and no subarrays IFT methods produce very similar antenna patterns. The extended sidelobe patterns, not shown in this figure, are all below the desired sidelobe level. (a) iIFT. (b) dIFT. (c) ssIFT. (d) GA. (e) No subarrays IFT.

array patterns/excitations to provide its best solution ($C = 11.64$) and 700 generated array patterns/excitations for its steady state solution ($C = 12.94$). In comparison the GA approach needs to generate 5000 array patterns to reach the same best cost function value as the dIFT approach, 6600 array patterns to reach the same best cost function value as the iIFT approach, and greater than 20 000 array patterns to reach the same best cost function value as the ssIFT approach. The best synthesized amplitude patterns are
Fig. 17. Amplitude patterns including the full visible region for the synthesized 80 × 80 array. Note that the extended sidelobe patterns are below the desired sidelobe level. (a) iIFT. (b) dIFT. (c) ssIFT. (d) GA. (e) No subarrays IFT.

Fig. 18. Element phase excitations for the synthesized 80 × 80 array. (a) iIFT. (b) dIFT. (c) ssIFT. (d) GA. (e) No subarrays IFT.

Table II summarizes the efficiency comparisons of the three proposed IFT methods with the GA method and identifies the improvement factor ($F_{imp}$) of each IFT method versus the GA method. Table III compares the best effectiveness of each method for 20,000 generated array patterns/excitations.

The provided efficiency results summarized in Table II show that the three proposed modified IFT methods are between four and greater than 667 times more efficient in synthesizing array patterns than GA which is a more common approach in the published literature. The GA approach can be more effective in generating a lower cost pattern if given enough computational resources and time. However, Table III shows that even after 20,000 iterations the effectiveness of the GA method is only a few percentage points higher than the iIFT, dIFT, and ssIFT methods which take substantially less computational time. The pattern efficiency value, $P_{eff}$, in Table III describes how close the generated pattern is to the desired pattern.

Table III

| Method | $iIFT$ | $dIFT$ | ssIFT |
|--------|--------|--------|-------|
| $F_{imp}$ | 6.38 | 6.47 | 6.36 |
| $P_{eff}$ | 78.10 | 77.22 | 78.29 |

F. Summary Results

Table II summarizes the efficiency comparisons of the three proposed IFT methods with the GA method and identifies the improvement factor ($F_{imp}$) of each IFT method versus the GA method. Table III compares the best effectiveness of each method for 20,000 generated array patterns/excitations.

The provided efficiency results summarized in Table II show that the three proposed modified IFT methods are between four and greater than 667 times more efficient in synthesizing array patterns than GA which is a more common approach in the published literature. The GA approach can be more effective in generating a lower cost pattern if given enough computational resources and time. However, Table III shows that even after 20,000 iterations the effectiveness of the GA method is only a few percentage points higher than the iIFT, dIFT, and ssIFT methods which take substantially less computational time. The pattern efficiency value, $P_{eff}$, in Table III describes how close the generated pattern is to the desired pattern.

A comparison of the proposed methods reveals that the ssIFT approach does limit the effectiveness of the synthesis by inhibiting the iterative process from jumping out of local optima to find a more effective global optima, but this approach is the most efficient approach overall. Among the IFT methods, the performance of the ssIFT is followed by the iIFT and then the dIFT. Although the ssIFT adds some initial complexity in its transition between the element spaced domain and the subarray spaced domain, it is able to utilize smaller FFT sizes and a simpler mean pooling technique than the other two presented IFT methods inside the iterative loop. This makes the ssIFT more computationally efficient than the other presented IFT methods.
V. CONCLUSION

This article has presented four methods (three modified IFT methods and one GA method) that have been found to be efficient and effective in generating array patterns for contiguous uniform subarrayed arrays. The modified IFT methods are significantly (up to 667 times) more efficient than the average performance of the GA method and are only slightly outperformed (a few percentage points in power efficiency in the main beam) in effectiveness by the GA.

This article has extended the impact of the IFT and GA methods and has shown that they can be efficiently and effectively utilized for synthesis of subarrayed architectures. Future work in this area includes an extension of this article that utilizes the iterative projection method (IPM) to give more freedom to the transition regions between the main beam and the sidelobe region.

REFERENCES

[1] J. C. Kerce, G. C. Brown, and M. A. Mitchell
Phase-only transmit beam broadening for improved radar search performance
In Proc. IEEE Radar Conf., 2007, pp. 451–456.
[2] A. F. Morabito, A. Massa, P. Rocca, and T. Isernia
An effective approach to the synthesis of phase-only reconfigurable linear arrays
IEEE Trans. Antennas Propag., vol. 60, no. 8, pp. 3622–3631, Aug. 2012.
[3] A. Chatterjee and G. K. Mahanti
Combination of fast fourier transform and self-adaptive differential evolution algorithm for synthesis of phase-only reconfigurable rectangular array antenna
Ann. Telecommun., vol. 69, no. 9/10, pp. 515–527, 2014.
[4] T.-S. Lee and T.-K. Tseng
Subarray-synthesized low-side-lobe sum and difference patterns with partial common weights
IEEE Trans. Antennas Propag., vol. 41, no. 6, pp. 791–800, Jun. 1993.
[5] P. Rocca, L. Manica, R. Azaro, and A. Massa
A hybrid approach to the synthesis of subarrayed monopulse linear arrays
IEEE Trans. Antennas Propag., vol. 57, no. 1, pp. 280–283, Jan. 2009.
[6] R. L. Haupt
Optimized weighting of uniform subarrays of unequal sizes
IEEE Trans. Antennas Propag., vol. 55, no. 4, pp. 1207–1210, Apr. 2007.
[7] P. Rocca, R. L. Haupt, and A. Massa
Sidelobe reduction through element phase control in uniform subarrayed array antennas
IEEE Antennas Wireless Propag. Lett., vol. 8, pp. 437–440, Feb. 2009.
[8] L. Manica, P. Rocca, G. Oliveri, and A. Massa
Synthesis of multi-beam sub-arrayed antennas through an excitation matching strategy
IEEE Trans. Antennas Propag., vol. 59, no. 2, pp. 482–492, Feb. 2011.
[9] S. Rajagopal
Beam broadening for phased antenna arrays using multi-beam subarrays
In Proc. IEEE Int. Conf. Commun., 2012, pp. 3637–3642.
[10] C. Balanis
Antenna Theory: Analysis and Design. Hoboken, NJ, USA: Wiley, 2016.
[11] O. Bucci, G. Franceschetti, G. Mazzarella, and G. Panariello
Intersection approach to array pattern synthesis
In Proc. IEE H (Microw., Antennas Propag.), 1990, vol. 137, pp. 349–357.
[12] G. Franceschetti, G. Mazzarella, and G. Panariello
Array synthesis with excitation constraints
In Proc. IEE H (Microw., Antennas Propag.), 1988, vol. 135, pp. 400–407.
[13] C. Carroll and B. V. Kumar
Iterative fourier transform phased array radar pattern synthesis
In Proc. SPIE, 1987, pp. 73–84.
[14] W. P. Keizer
Fast low-sidelobe synthesis for large planar array antennas utilizing successive fast fourier transforms of the array factor
IEEE Trans. Antennas Propag., vol. 55, no. 3, pp. 715–722, Mar. 2007.
[15] W. P. Keizer
Low-sidelobe pattern synthesis using iterative fourier techniques coded in MATLAB [EM programmer’s notebook]
IEEE Antennas Propag. Mag., vol. 51, no. 2, pp. 137–150, Apr. 2009.
[16] Y.-Q. Wen, B.-Z. Wang, and X. Ding
A wide-angle scanning and low sidelobe level microstrip phased array based on genetic algorithm optimization
IEEE Trans. Antennas Propag., vol. 64, no. 2, pp. 805–810, Feb. 2016.
[17] T. M. Mitchell
Machine Learning. Boston, MA, USA: McGraw-Hill, 1997.
[18] G. Rudolph
Convergence analysis of canonical genetic algorithms
IEEE Trans. Neural Netw., vol. 5, no. 1, pp. 96–101, Jan. 1994.

Barry K. Daniel (Senior Member, IEEE) received the B.S. degree in electrical engineering, and the B.S. degree in computer science from Tennessee Technological University, TN, USA, both in 1995, and the M.S. degree in electrical engineering from the University of Alabama, Huntsville, USA, in 2000. He is currently working toward the Ph.D. degree in electrical engineering with Tennessee Technological University.

For 10 years, he worked in Huntsville, Alabama, where he developed simulations of radar and optical portions of missile threat systems and performing signal processing analysis. Since 2005, he has been working for Oak Ridge National Laboratory, in the areas of antenna array synthesis, efficient digital processing algorithms, track/data fusion, machine learning, real-time simulation test beds to analyze new waveforms, evaluation of prototype radar designs, and environment models that propagate RF and EO/IR energy.
Adam L. Anderson (Senior Member, IEEE) received the B.S. and M.S. degrees from Brigham Young University, Utah, USA, in 2002 and 2004, respectively, and the Ph.D. degree from the University of California at San Diego, California, USA, in 2008. He is currently a Scientist at MRSL Real-Time Systems Laboratory, Monterey, CA, USA.

Dr. Anderson was the winner of the 2014 DARPA Spectrum Challenge. He was the recipient of the 2014 Leighton E. Sissom Award for Creativity and Innovation, and was recently awarded as a Finalist in the 2019 DARPA Spectrum Collaboration Challenge.