We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,300

Open access books available

172,000

International authors and editors

190M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Designing antenna arrays using signal processing, image processing and optimization toolboxes of MATLAB

Joseph Sahaya Kulandai Raj and Joerg Schoebel
*Braunschweig University of Technology
 Germany*

1. Introduction

An antenna array is a group of identical antenna elements arranged usually in a regular fashion. In a linear array, the elements are arranged along a straight line. They are arranged on a grid of points on a plane for a planar array. Generally the currents through the elements are of different amplitudes and phases in order to obtain greater control over the radiation pattern. Antenna arrays are important components of present day wireless communication systems. They have become ubiquitous. The current wireless standards include advanced antenna array concepts such as adaptive antenna arrays and MIMO (Multiple- Input and Multiple- Output) systems to improve the performance of the communication system. Therefore understanding and designing antenna arrays have become imperative.

Basically the antenna array design involves calculating the complex currents of the individual antenna elements and selecting an appropriate antenna element. The current excitations would largely determine how sharp the resultant radiation pattern is and how small the side lobe levels are in comparison to the main lobe. After calculating the current excitations, one would want to visualize the resultant radiation pattern to verify the design. MATLAB®, one of the languages of technical computing, has many built-in functions and visualization tools that would help the design process. Here we demonstrate how toolboxes such as signal-processing, image-processing and optimization toolboxes can help in designing an array and visualize its radiation pattern.

In addition to designing antenna arrays, the toolboxes can be effectively used in the teaching and learning process. Generally, the learning process involves augmenting one's existing knowledge with new concepts. Therefore, while teaching a new concept to students, their already acquired knowledge can be exploited to help them assimilate the concept. Explaining a new concept to students is made effective if we compare the new concepts to other concepts, with which the students are already familiar. The strong similarity between antenna theory and signal-processing theory can be used in this way. Students typically learn signal-processing theory earlier than antenna theory, and antenna educators can make use of the former to illustrate the latter. Though programs can be written in MATLAB without using the signal and image-processing toolboxes to analyze and synthesize antenna arrays, students

can gain additional knowledge and insight by comparing signal and image-processing theories with antenna theory, using them. These toolboxes have many built-in functions for filter synthesis that are useful for array synthesis.

2. Use of Signal Processing Toolbox

The antenna array design process is fundamentally similar to the filter synthesis problem. This enables the use of the signal processing functions in antenna array analysis and synthesis.

2.1 Analysis

Given the currents on the radiating elements, calculating the radiation pattern and plotting it to visualize it is one of the tasks of antenna analysis. In this section, we will show how a built-in function of the signal processing toolbox is used in antenna analysis. The array factor of n isotropic point sources with different excitations placed along the x axis on the x - y plane is given by (Balanis, 2005)

$$AF = w_0 + w_1e^{j\psi} + w_2e^{j2\psi} + \dots + w_{n-1}e^{j(n-1)\psi}$$

(1)

where

$$\psi = \beta d \cos \phi$$

the w_n are the complex currents of the elements, β is the wave number, d is the inter-element spacing and ϕ is the angular variable. The frequency response of a n point FIR filter is given by

$$H(j\omega) = h_0 + h_1e^{-j\omega} + h_2e^{-j2\omega} + \dots + h_{n-1}e^{-j(n-1)\omega}$$

(2)

where h_n are the impulse-response coefficients of the FIR filter and ω is discrete angular frequency.

Equations (1) and (2) are of similar form, except for a minus sign in the power of the complex exponential. Therefore, the tool that is used for finding the frequency response can be used to find the array response as a function of direction ϕ . The built-in function `freqz` in the signal-processing toolbox of MATLAB evaluates the frequency response given the impulse-response coefficients (SP toolbox, 2010). The same function can be used to evaluate the radiation pattern of the array given the current excitations. For example, the current excitation of a four-element array is $[1, 1.2, 1.2, 1]$. Fig. 1 shows the pattern obtained by using `freqz` command. The following MATLAB code generates the radiation pattern of the four-element array of this example.

Code 1 To generate the pattern

<code>phi=0:0.01:2*pi;</code>	<code>%0<phi<2*pi</code>
<code>shi=pi*cos(phi);</code>	<code>%For lambda/2 spacing</code>
<code>Currents=[1,1.2,1.2,1];</code>	<code>%Current excitations</code>
<code>E=freqz(Currents,1,shi);</code>	<code>%E for different shi values</code>
<code>polar2(phi,E);</code>	<code>%Generating the radiation pattern</code>

As it is desirable to plot the radiation pattern of an array in dB scale, the built-in function `polar` is modified to plot in dB scale and is called as `polar2` in the code.

Fig. 1. The pattern of a four element array with current excitations [1, 1.2, 1.2, 1]

2.2 Synthesis

2.2.1 Least square method

One of the problems of array synthesis consists of finding the current excitations of the array elements, given the desired radiation pattern or some data about the radiation pattern. This synthesis problem is equivalent to the problem of finding the impulse response of a digital FIR filter for the desired frequency response, as follows from the explanation in the previous section. The MATLAB Signal-Processing Toolbox has the `firls` function for designing an one-dimensional FIR filter that is optimal in a least-squares sense (SP toolbox, 2010). The arguments of the function are:

- N , one less than the number of impulse response coefficients;
- F , a vector of frequency points, in ascending order between 0 and 1; 1 corresponds to half the sampling frequency;
- A , a real vector of the same size as F , which specifies the desired amplitude of the frequency response of the resultant filter.

If the function is given as `firls(N, F, A)`, it returns an FIR filter with $N + 1$ impulse-response coefficients that has the best approximation to the desired frequency response described by F and A in the least-squares sense. This function can be used to design an antenna array if a proper mapping is executed between the frequency variable, F , and the variable ψ , which in turn is related to ϕ . For a broadside array, if the spacing between the elements is $\lambda/2$, then $\psi = \pi \cos \phi$. In this case, when ϕ varies from 0 to π , ψ will vary from π to $-\pi$. Thus, for the F -vector values in the `firls` function to vary from 0 to 1 (the normalized range of ω), the corresponding ψ values will range from 0 to $-\pi$. This restricts the specification of $A(\phi)$ to be specified from $\pi/2$ to π . The normalized E-field values of the desired radiation pattern at the ψ vector values is given by A . N will now be equal to the number of elements in the array minus one. The values in the vector F should be in ascending order and normalized to unity. For instance, let us design an antenna array that will give an approximately bidirectional pattern as in Fig. 2 (indicated by the blue line). The pattern is defined by:

$$A(\phi) = \begin{cases} 0.1 & \text{if } 0 < \phi < \pi/3, \\ 1 & \text{if } \pi/3 < \phi < 2\pi/3 \\ 0.1 & \text{if } 2\pi/3 < \phi < \pi. \end{cases} \quad (3)$$

Using the above approach, for a 16 element linear array, respective current excitations were obtained. The pattern of the synthesized array is indicated by a blue line in Fig. 2. The following MATLAB code synthesizes the array for a given specification.

Code 2 To synthesize a linear array in the least-squares sense

```
a=0:pi/315:pi/3; %Side lobe directions
b=pi/3:pi/315:2*pi/3; %Main lobe directions
c=2*pi/3:pi/315:pi; %Side lobe directions
T=[a b c ];L=length(b); %The direction variable
Spec=[0.1*ones(1,L),... %Specifications
      ones(1,L),...
      0.1*ones(1,L)];
shi=pi*cos(T);
F=shi((length(T))/2-1):-1:1)/pi; %Sorting shi in ascending
                                %order and normalizing to 1
A= Spec((length(T)/2-1):-1:1); %Sorting in the order of shi
Currents=firls(15,F,A); %Synthesis
E=freqz(Currents,1,shi); %Analysis
EdB=20*log10(abs(E)); %dB scale conversion
plot(T*180/pi,EdB,'LineWidth',2);
axis([0 180 -25 10])
```

The values of the excitation currents can be found by displaying the variable `Currents`. The script for plotting the specification is excluded from Code 2.

Fig. 2. Pattern of an array synthesized using least square method

2.2.2 Chebyshev and Taylor Syntheses

Another useful function regarding synthesis is `chebwin` (SP toolbox, 2010). For instance, if a six- element Chebyshev array is to be designed with a certain sidelobe level (*SLL*), the function `chebwin` available in the signal processing tool box can be used. The `chebwin(n,r)` returns a *n*-point Chebyshev window with side lobe level of *r* dB. The window values are the

current excitations of the array elements. Discrete arrays having taylor patterns can also be synthesized in the same way using `taylorwin(N,Nbar,SLL)`. This function returns a Taylor window of N-points with side lobe level of SLL dB (SP toolbox, 2010). Nbar is a parameter of the window. The radiation patterns of the arrays synthesized using the window functions are shown in Fig. 3. The number of elements and the side lobe levels are taken to be 8 and -20dB respectively.

Fig. 3. Synthesized patterns of Chebyshev and Taylor arrays

The following code synthesizes a Chebyshev array and a Taylor array of eight elements with -20dB side lobe level.

Code 3 To synthesize Chebyshev and Taylor arrays

```
phi=0:0.01:2*pi; %0<phi<2*pi
shi=pi*cos(phi); %For lambda/2 spacing
Currents=chebwin(8,20); %Chebyshev Window values
E=abs(freqz(Currents,1,shi)) %E for different shi values
EdB(1,:)=20*log10(E/max(E)); %Converting into dB scale
Currents=taylorwin(8,3,-20); %Taylor Window values
E=abs(freqz(Currents,1,shi)) %E for different shi values
EdB(2,:)=20*log10(E/max(E)); %Converting into dB scale
plot(phi*180/pi,EdB,'LineWidth',2) %Generating the pattern
axis([0 180 -30 10])
```

2.2.3 Constrained least square method

In this section, we demonstrate the use of the `fircls` function in designing the antenna array. This function designs a FIR filter by minimizing the integral square error ($\|E(\omega)\|_2$) between the desired frequency response ($D(\omega)$) and the magnitude response of the filter ($A(\omega)$) subject to the constraint that the ripples of ($A(\omega)$) lie within the specified bounds. The minimization of ($\|E(\omega)\|_2$) is done until ($A(\omega)$) is within the bounds. The integral square error is:

$$\|E(\omega)\|_2 = \left(\frac{1}{\pi} \int_0^\pi (A(\omega) - D(\omega))^2 d\omega \right)^{\frac{1}{2}}$$

In order to use this built-in function to design an array, proper mapping between the digital angular frequency (ω) and the angle variable must be accomplished as discussed in section 2.2.1. The arguments of the function are:

- f is a vector of transition frequencies in the range from 0 to 1, where 1 corresponds to the Nyquist frequency. The first point of f must be 0 and the last point 1. The frequency points must be in increasing order;
- amp is a vector describing the piecewise constant desired amplitude of the frequency response. The length of amp is equal to the number of bands in the response and should be equal to $length(f)-1$;
- up and lo are vectors with the same length as amp . They define the upper and lower bounds for the frequency response in each band.

If the function is given as `fircls(n, f, amp, up, lo)`, it generates a length $n+1$ linear phase FIR filter. According to equations (1) and (2), the range 0 to π of the angular digital frequency maps onto the range 0 to $-\pi$. This means that we can specify a pattern from $\phi = \pi/2$ to π . Since the frequency response of the digital filter is periodic, the magnitude response of a real filter in the range 0 to π is equal to 0 to $-\pi$. Therefore the pattern for the range 0 to $\pi/2$ is just the mirror image of the pattern for the range $\pi/2$ to π . The mapping between the filter frequency variables and the antenna angle variables are given in the following table.

Azimuth angle	ϕ	0	$\pi/2$	π
Progressive phase	ψ	π	0	$-\pi$
Angular frequency	ω	$-\pi$	0	π
Frequency	F	-0.5	0	0.5
Frequency (normalized to 1)	f	-1	0	1

Table 1. Mapping between filter and antenna variables

Fig. 4. Pattern of an array synthesized using constrained least square method

Supposing that a flat beam pattern antenna array is to be designed. The main lobe is between $\phi = \pi/3$ to $\phi = 2\pi/3$. The sidelobe level is -40 dB. The number of elements is $n = 33$. This

would mean the main lobe is between $\pi/2$ and $-\pi/2$ in ψ domain. The corresponding filter passband ranges from $\omega = -\pi/2$ to $\omega = \pi/2$. In the normalized frequency domain, this is from -0.5 to 0.5. The built-in function requires the normalized frequency to be specified only from 0 to 1, as the magnitude response of a filter is symmetric about $f=0$. For this example, the normalized cutoff frequency is 0.5. The specification and the designed pattern are shown in Fig. 4. The script for plotting the specification is excluded from Code 4.

Code 4 To synthesize an array using constrained least square method

```
f=[0 0.5 1];amp=[1 0];           %Specification
up=[1.02 0.01];lo =[0.98 -0.01]; %Bounds
phi=0:0.01:2*pi;                 %0<phi<2*pi
shi=pi*cos(phi);                 %For lambda/2 spacing
b = fircls(32,f,amp,up,lo);       %Sythesis
E=freqz(b, 1, shi);              %E for different shi values
EdB=20*log10(abs(E)/max(abs(E))); %Converting into dB scale
plot(phi*180/pi,EdB,'LineWidth',2) %Generating the pattern
axis([0 180 -50 10])
```

3. Schelkunoff polynomial method

The array factor of a linear array (see (1)) can be rewritten as

$$AF = w_0 + w_1z + w_2z^2 + \dots + w_{n-1}z^{n-1}$$

where

$$z = e^{j\beta d \cos \phi}$$

For an array with a real beam pattern, the array factor is given by

$$AF = w_0 + w_1z + \dots + w_1z^{n-2} + w_0z^{n-1} \tag{4}$$

For a real beam pattern, the excitations coefficients of an array are symmetric and the roots of the array factor occur in complex conjugate pairs. The Z transform of a symmetric discrete time sequence is given by

$$X(Z) = a_0 + a_1z^{-1} + \dots + a_1z^{-(n-2)} + a_0z^{-(n-1)} \tag{5}$$

The equations (4) and (5) are of similar form except for the minus sign. We can rewrite the Z transform equation as below to compare the array factor.

$$X(Z) = \frac{a_0z^{n-1} + a_1z^{n-2} + \dots + a_1 + a_0}{z^{n-1}} \tag{6}$$

Exploiting this similarity, we may use a built-in function called `zplane` of the signal processing toolbox which is used to visualize the poles and zeros of a discrete filter to see the locations of the roots of a real beam pattern. Because of negative powers of the z terms in the Z transform, we would get $(n - 1)$ poles at $z = 0$. For instance, we would like to see the roots of an array factor on the z plane. The array factor is given by

$$AF = z^3 - z^2 + z - 1$$

If we treat the above polynomial as the numerator of a Z transform of a sequence, then the function `zplane([-1 1 -1 1], 1)` will give the locations of the roots of the array factor on the z plane in addition to three poles at the origin, which we can ignore (See Fig. 5). Although one can write a separate MATLAB code to find the roots of the array factor and then plot it on a rectangular graph using simple commands like `poly`, `roots` and `plot`, comparing the Schelkunoff polynomial with the Z transform would give more insight into the array factor dependence on the roots. One may also be interested in the Pole/Zero editor of the `fdatool` of the signal processing toolbox. This interactive editor allows the user to adjust the locations of the poles and zeros on the z plane and see the effect on the frequency response which is analogous to the array factor.

Fig. 5. Locations of the roots of the beam pattern (indicated in red)

4. Use of Image Processing Toolbox

4.1 Analysis

The array factor for an $M \times N$ rectangular array on the x-y plane (Ioannides & Balanis, 2005) is given by

$$AF(\theta, \phi) = \sum_{m=1}^M \sum_{n=1}^N w_{mn} e^{j[(m-1)\psi_x + (n-1)\psi_y]} \quad (7)$$

where

$$\psi_x = kd_x \sin \theta \cos \phi$$

$$\psi_y = kd_y \sin \theta \sin \phi$$

where d_x and d_y are the spacings between the adjacent elements along the x axis and the y axis, respectively, and the w_{mn} are the current excitations. The frequency response of a two-dimensional filter is given by

$$H(j\omega_1, j\omega_2) = \sum_{m=1}^M \sum_{n=1}^N h_{mn} e^{-j[(m-1)\omega_1 + (n-1)\omega_2]} \quad (8)$$

Equations (7) and (8) are of similar format, except for a minus sign in the power of the complex exponentials. Therefore, the tool that is used for finding the frequency response can be used to find the array response as a function of variables θ and ϕ . The image-processing toolbox of MATLAB has a function `freqz2` for finding the frequency response of two-dimensional filters. The same could be effectively used to evaluate the radiation pattern of the two-dimensional arrays (IP toolbox, 2010). The radiation pattern of the array on the x-y plane is shown in Fig. 6. For example, the code given below generates the radiation pattern of a 5 x 5 uniform rectangular array. The following MATLAB function generates the three dimensional radiation pattern of a rectangular array. This requires three arguments:

- `Currents`, a matrix whose elements define the current distribution of the array;
- `theta`, an array having the θ directions in which the pattern is evaluated;
- `phi`, an array having the ϕ directions in which the pattern is evaluated .

The function `freqz2` requires the frequency variables to be normalized frequencies in the range -1.0 to 1.0, where 1.0 corresponds to half the sampling frequency, or π radians. Due to this, the progressive phase shifts ψ_x and ψ_y are normalized so that their limits are equal to the limits of the normalized frequencies of a two dimensional filter. The normalized values of the progressive phase shifts are defined as follows:

$$\begin{aligned}\psi_{nx} &= \sin \theta \cos \phi \\ \psi_{ny} &= \sin \theta \sin \phi\end{aligned}\tag{9}$$

Finally $E(\theta, \phi)$ is converted into $E(x, y)$ so that the three dimensional plotting function `mesh` can be used.

Code 5 To generate a three dimensional radiation pattern

<code>function analyze2D(Currents,theta,phi)</code>	<code>%Function definition</code>
<code>[the,ph]=ndgrid(theta,phi);</code>	
<code>shinx=cos(ph).*sin(the);</code>	<code>%Normalized shinx</code>
<code>shiny=sin(ph).*sin(the);</code>	<code>%Normalized shiny</code>
<code>E=freqz2(Currents,shinx,shiny);</code>	<code>%Analysis</code>
<code>x=abs(E).*sin(the).*cos(ph);</code>	<code>%Spherical to</code>
<code>y=abs(E).*sin(the).*sin(ph);</code>	<code>%rectangular</code>
<code>z=abs(E).*cos(the);</code>	<code>%coordinate system</code>
<code>mesh(x,y,z)</code>	<code>%3D plot</code>

4.2 Synthesis

Similarly, two-dimensional rectangular arrays can be synthesized using tools available in the image-processing toolbox. The tools are `fwind1`, `fsamp2`, `ftrans2`, and `fwind2` (IP toolbox, 2010). For example, `H=fwind2(FR,WIN)` designs a two-dimensional FIR filter `H` with frequency response `FR`. `fwind2` uses the two-dimensional window `WIN` to truncate the infinitely large impulse response required to meet the given frequency response, `FR`. `WIN` can be the `boxcar(rectangular)`, `hamming`, `hanning`, `bartlett`, `blackman`, `kaiser`, or `chebwin` window functions of the signal processing toolbox. To synthesize the rectangular array, the `fwind2` function’s argument `FR` is given by the desired radiation pattern samples in the θ and

Fig. 6. The radiation pattern of a 5×5 uniform rectangular array

ϕ directions, and the dimension of WIN is given by $N \times N$, the dimension of the array. The output, H, is the array excitations. Fig. 7(a) shows the desired pattern, and Fig. 7(b) shows the synthesized pattern, of the two-dimensional array.

Fig. 7. (a) The desired and (b) the synthesized pattern for the 10×10 two-dimensional array

The following MATLAB code synthesizes the array whose radiation pattern approximates a conical pattern (see Fig. 7) The code uses a built-in function called `freqspace`, which creates the two-dimensional normalized frequency vectors and returns a grid of values for each normalized frequency vector as if `meshgrid` is applied to the vectors. An intermediate variable is introduced to define the conic beam. Let the intermediate variable be r and is equal to

$$r = \sqrt{\psi_{nx}^2 + \psi_{ny}^2}$$

Using (9), $r = \sin \theta$ for all ϕ . The specification of the conic beam is

$$E_{spec}(\theta) = \begin{cases} 1 & \text{if } 0 < \theta < \pi/6 \\ 0 & \text{if } \pi/6 < \theta < \pi/2. \end{cases}$$

Code 6 To synthesize a rectangular array

<pre>phi=2*pi*((0:1:100)/100); theta=(pi/2)*((0:1:100)/100); [shix,shiy]=freqspace(101,'meshgrid'); r=sqrt(shix.*shix+shiy.*shiy); Espec=ones(101); Espec(r>sin(pi/6))=0; win=boxcar(10)*boxcar(10)'; Currents=fwind2(Espec,win); analyze2D(Currents,theta,phi)</pre>	<pre>%0<phi<2*pi %0<theta<pi/2 %Frequency points %Specifications %Window %Synthesis</pre>
--	---

5. Spacing and Sampling Rate

The effect on the array pattern of changing the spacing between the adjacent elements of the array is same as the effect on the frequency response of changing the sampling period of the FIR filter. To illustrate this point, the frequency responses of a 15-tap FIR filter with uniform impulse response coefficients for different sampling rates are compared with the array responses of a 15-element uniform linear array with different spacing. The frequency responses are plotted from -1 Hz to 1 Hz for the Nyquist rate (F_s), one-half the Nyquist rate, and one-quarter of the Nyquist rate in Fig. 8 (a), (b) and (c) respectively. It can be observed that the frequency response shrinks as the sampling frequency decreases (the sampling period increases). As a result, more main lobes creep into the fundamental range, which otherwise has only one main lobe.

In Fig. 9 (a), (b) and (c), the array responses for a 15-element uniform linear array are plotted to show the analogy. As the spacing between the elements is increased, more main lobes occur within the range of ϕ , just as in the frequency response of the FIR filter. This analogy can be used to explain the occurrence of grating lobes to the students, who have already come across the phenomenon of aliasing. Aliasing occurs when the signal is sparsely sampled. Aliasing is reflected in the frequency response of the filter, because it should respond in the same way for two different frequencies having the same identity. Similarly, aliasing of the pattern in the spatial domain occurs when the array elements are sparsely arranged. If the sampling rate is sufficiently high, the frequency response expands, and only the main lobe may occupy the entire fundamental range of -1 Hz to 1 Hz. In the same way, if the array elements are arranged densely enough, the extra grating lobes which would otherwise occur will disappear from the range of ϕ .

6. Convolution and Pattern Multiplication

Pattern multiplication is used to find the array pattern from the patterns of individual elements. This is analogous to multiplication of two frequency responses, corresponding to two

Fig. 8. (Top to bottom: a, b, and c) The frequency responses of a 15-tap FIR filter for different sampling rates.

Fig. 9. (Top to bottom: a, b, and c) The array responses of a 15-point uniform linear array for different spacings between the elements.

discrete time sequences. The resultant frequency response obtained by multiplication is actually the frequency response of the resultant sequence obtained by convolving the two discrete time sequences. In the same way, an array for which the pattern is needed can be thought of as a result of convolution of two arrays, one comprising m individual elements of the array and the other comprising n elements. Each of the n elements is a combination of m elements.

For instance, to demonstrate the pattern-multiplication concept, a four-element isotropic array, shown in Fig. 10 (a) is considered here. The four element array is treated as a combination of two arrays of two elements each. Each two-element isotropic array can be regarded as a directional antenna. Then, the four-element array is equivalent to an array of two directional antennas separated by a distance $2d$. Now, by multiplying the pattern of the d -spaced two-element array with the pattern of the d -spaced two-element array, the pattern of the four-element array can be obtained.

The four-element array is comparable to a discrete time sequence having four unit samples with sampling period d . The two-element array separated by distance $2d$ is comparable to a sequence having two unit samples with sampling period $2d$, and so on. The discrete time sequence with $2d$ sampling period (Sequence 1) is $[1, 0, 1]$ and with d sampling period (Sequence 2) is $[1, 1]$. The resultant sequence (Sequence 3) is $[1, 1, 1, 1]$ which is obtained by the script `conv([1, 0, 1], [1, 1])`. Just as the time sequence 3 can be considered as a result of a convolution of time sequences 1 and 2, as shown in Fig. 10 (b), the four-element array can be regarded as the result of a convolution of two arrays of two elements with spacing $2d$ and d distances. Since the convolution of two sequences in the time domain corresponds to multiplication of frequency responses in the frequency domain, the convolution of two arrays corresponds to multiplication of the radiation patterns of the respective two-element arrays (Raj & Kabilan, 2007).

Fig. 10. (a) The convolution of two arrays of two elements with spacings of $2d$ and d , respectively. (b) The convolution of two sequences with sampling rates of spacings of $2d$ and d , respectively.

7. Use of Optimization Toolbox

Optimization is one of the main techniques that enable the designer to design an antenna array having asymmetric sidelobes and/or arbitrary main lobe, for example (Schoebel et al., 2005). In this section, we demonstrate how the optimization toolbox of MATLAB can be used to do antenna array optimization. Here we use the built-in function `fmincon` of the toolbox, as this function performs nonlinear constrained optimization (Optimization toolbox, 2010). This is suitable for the design of antenna array whose pattern is a nonlinear function of the direction variable and the pattern is constrained at various directions to meet the sidelobe requirement (Lebret & Boyd, 1997).

The problem that we consider for our demonstration is formulated as follows.

$$\begin{aligned} \min_{w_i} \quad & \left(\max_{m=1,\dots,P} |AF(\phi_m)| \right) \\ \text{subject to} \quad & |AF(\phi_n)| \leq U_n, \quad n = 1, \dots, Q \\ & |AF(\phi_s)| = 1. \end{aligned} \quad (10)$$

- ϕ_m , directions in which the array factor is evaluated for the minimax problem;
- ϕ_n , directions in which the array factor is constrained to be equal to U_n ;
- ϕ_s , angle at which the normalized gain is equal to 1.

The angles ϕ_m and ϕ_n will be within the two sidelobe regions. The original problem which involves complex numbers is cast into a formulation involving only real numbers. The weights w_i and the steering vector terms s_i are expressed as below.

$$\begin{aligned} w_i &= \Re(w_i) + i\Im(w_i) \\ e^{i\beta d_i \cos(\phi)} &= \Re(s_i) + i\Im(s_i). \end{aligned}$$

where d_i is the position of the i^{th} element from the origin. Now we can write the array factor as

$$AF(w_i) = a_m^T x + i b_m^T x \quad (11)$$

where

$$\begin{aligned} x &= (\Re(w_1), \dots, \Re(w_n), \Im(w_1), \dots, \Im(w_n)); \\ RS &= (\Re(s_1), \dots, \Re(s_n)); \\ IS &= (\Im(s_1), \dots, \Im(s_n)); \\ a_m &= (RS, -IS); \\ b_m &= (IS, RS) \end{aligned}$$

Following (Lasdon et al., 1987), the problem in (10) is

$$\begin{aligned} \min_{w_i} \quad & z \\ \text{subject to} \quad & |AF(\phi_m)|^2 < z, \quad m = 1, \dots, P \\ & |AF(\phi_n)|^2 = U_n, \quad n = 1, \dots, Q \\ & |AF(\phi_s)|^2 = 1. \end{aligned} \quad (12)$$

The remaining section will show how to use `fmincon` function to the above optimization problem. The specification for the beam pattern is as follows:

```
phim=[0,44,57,72,106,122,140,180]*pi/180;
phin=[30,55,67,90,112,128,150]*pi/180;
goal=[0.01,0.01,0.01,1,0.02,0.02,0.02];
```

In this piece of code, `phin` vector has the values of the angles in which the magnitude square of the array factor is constrained to be equal to the values in the vector `goal`. The goal is specified for the power pattern. The goal is equal to -20dB for the first three `phin` and is equal to -17 dB for the last three `phin` angles. When `phin`=90°, it is 0 dB. A separate function

Fig. 11. Asymmetric pattern of an array of 8 elements with the specification indicated

must be written for the objective. This function returns the function value which is to be minimized. For our example, the argument is equal to the function value. The code for the objective function is given below.

```
function z=objfun(a0,goal,phim,phin)
%a0=[R(w1),...,R(wN),I(wN),...,I(w1),z]
z=a0(end);
%Since objective function is equal to the input argument.
```

Only the input `a0` is used in the objective function. The remaining arguments are not used but they must be included because the arguments of the objective function and the constraint function (described below) are the same. These arguments are passed by the optimization tool `fmincon` when it invokes the objective and constraint functions by their handles. Similar to the objective function, the constraints are also stored as a separate MATLAB function. We show below the code for the constraint function.

```
function [c,ceq]=confun(a0,goal,phim,phin)
N=length(a0)-1;x=a0(1:1:N);phi=[phim,phin];
for n=1:length(phi)
AR=exp(-j*pi*(0:N/2-1)*cos(phi(n)));
RS=real(AR);IS=imag(AR);
a=[RS,-IS];b=[IS RS];
P(n)=(a*x').*(a*x')+(b*x').*(b*x');
end
Pnorm=P/max(P);
z=a0(end)*ones(1,length(phim));
c=Pnorm(1:length(phim))'-z';
ceq=goal'-Pnorm(length(phim)+1:end)';
```

In order to understand the variables of the above code, one may refer (11). The optimization tool is invoked as:

```
options = optimset('Algorithm','interior-point');
Wopt= fmincon('objfun',a0,[],[],[],[],...
lb,ub,'confun',options,goal,phim,phin);
```

As the problem does not have any linear constraints, the corresponding arguments must be empty. The arguments `lb` and `ub` define the bounds on the real and imaginary parts of the weight vector and the objective function input. Fig. 11 shows the normalized pattern of the array with sidelobe specifications.

8. Conclusion

Exploiting the similarity between signal-processing theory and antenna-array theory, it has been demonstrated that with little effort analysis of antenna arrays can be done using signal- and image-processing tools of MATLAB. The analogy between FIR filters and arrays can be effectively used to explain the concepts of the arrays to students. Also, synthesis of arrays, to a first approximation, can be carried out using signal-processing tools in the case of one-dimensional arrays, and image-processing tools in the case of two-dimensional arrays, ignoring the mutual coupling between the elements. It has been shown that the optimization toolbox of MATLAB can be used to design antenna arrays with asymmetric sidelobes.

9. References

- Balanis, C. (2005). *Antenna Theory: Analysis and Design*, 3rd ed., Wiley and Sons, 047166782X, Hoboken, New Jersey, USA.
- Ioannides, P. & Balanis, C. (2005). Uniform circular and rectangular arrays for adaptive beam-forming applications, *IEEE Antennas Wireless Propag. Lett.*, vol.4, no., pp. 351- 354.
- Image Processing Toolbox User's Guide. (2010), The MathWorks Inc.
- Lasdon L. S. ; Plummer, J. ; Buehler, B. & Warren A. D. (1987). Optimal design of efficient acoustic antenna arrays, *Math. Programming*, vol.39, no. 2, pp. 131-155.
- Lebret, H. & Boyd, S. (1997). Antenna array pattern synthesis via convex optimization, *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 526-532.
- Optimization Toolbox User's Guide. (2010), The MathWorks Inc.
- Raj, J. S. K & Kabilan, A.P. (2007). Teaching and Designing Antenna Arrays Using Signal and Image-Processing. Toolboxes of MATLAB, *IEEE Antennas Propag. Mag.*, vol. 49, issue 4, pp. 184-190.
- Schoebel, J.; Buck, T.; Reimann, M.; Ulm, M.; Hansen, T. & Schneider, M. (2005) Planar Phased Array Antenna Systems for W-Band Automotive Radar Sensors with Beam Steering Based on RF MEMS, *Frequenz*, vol. 59, No. 1-2, pp. 27-34.
- Signal Processing Toolbox User's Guide. (2010), The MathWorks Inc.

Matlab - Modelling, Programming and Simulations

Edited by Emilson Pereira Leite

ISBN 978-953-307-125-1

Hard cover, 426 pages

Publisher Sciyo

Published online 05, October, 2010

Published in print edition October, 2010

This book is a collection of 19 excellent works presenting different applications of several MATLAB tools that can be used for educational, scientific and engineering purposes. Chapters include tips and tricks for programming and developing Graphical User Interfaces (GUIs), power system analysis, control systems design, system modelling and simulations, parallel processing, optimization, signal and image processing, finite different solutions, geosciences and portfolio insurance. Thus, readers from a range of professional fields will benefit from its content.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Sahaya Kulandai Raj Joseph and Joerg Schoebel (2010). Designing Antenna Arrays Using Signal Processing, Image Processing and Optimization Toolboxes of MATLAB, Matlab - Modelling, Programming and Simulations, Emilson Pereira Leite (Ed.), ISBN: 978-953-307-125-1, InTech, Available from:
<http://www.intechopen.com/books/matlab-modelling-programming-and-simulations/designing-antenna-arrays-using-signal-processing-image-processing-and-optimization-toolboxes-of-matl>

INTECH
open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen