AN OLD TOOL AND A NEW CHALLENGE FOR DEPICTING
ANTENNA ARRAY RADIATION PATTERNS
being also
A CLASSROOM DEMO FOR THE PRINCIPLE OF WAVE INTERFERENCE UNDERLYING
ANTENNA ARRAY THEORY

Juan.R.Mosig, LEMA-EPFL

INTRODUCTION

The basic facts allowing the intuitive understanding of how antenna array patterns arise from basic interference phenomena are frequently obscured by careless expositions and technical treatments which rely too much on the mathematical properties of complex exponential functions and discourage students who do not anymore possess the trigonometric skills of their elders. However, many basic features of radiation patterns (position and number of secondary lobes, null angles, beam steering...) can be always successfully explained with the same graphical tools similar to those that were usual in old fashioned courses on optics, acoustics and vibration&waves. They deserve a modest revival.

In this paper, we describe one of these graphical devices, that has been successfully used by the author as a classroom demonstration in many undergraduate antenna courses. In addition, this traditional demonstration is also discussed in the context of a modern computer software, like MatLab. Although surely not the best language to obtain impressive graphical representations, MatLab can be easily used for a proof-of-concept trial and to pave the way for more sophisticated approaches.

A final word must be said about the lack of bibliography and references in this paper. The following developments weren't not directly inspired by any previously existing published material and arose as the result of the author's inner thoughts, continuously improved by exposure to young students and careful evaluation of their reactions. But like almost everything that we naïvely believe original and the sole fruit of our brilliant mind, these ideas were probably inspired in a sub-conscientious way by some readings and pictures encountered during a lifetime of interest on wave phenomena, like the Fresnel-Young double-slit experiment, which is a basic staple of classic and modern Physics textbooks.

1. BASIC CONCEPTS

Array theory is based in that elusive concept called "point isotropic time-harmonic source". Let's accept it without further discussion and start with the classical expression for a scalar harmonic wave function generated by such a source:

\[ f(r,t) = A \cos(\omega t - kr) \]

Here we will disregard the amplitude variation to concentrate only in phase properties which are the most relevant in the interference phenomena. Also, we will start with a snapshot of the phenomena taken at a given time \( t_0 \)(the possibility of a "movie" including the time variation will be considered later). Therefore, we will assume for the sake of simplicity an unit amplitude, \( A = 1 \), and we will reformulate our basic expression as:

\[ f(r) = \cos[2\pi(r / \lambda - \tau / T)] \]
where we have introduced the normalized distance in terms of wavelengths \( r / \lambda = kr / 2\pi \) and a normalized timeshift or phaseshift in terms of periods \( \tau / T = \omega t_0 / 2\pi \). We will discuss in this paper only one-dimensional (1D) and two-dimensional (2D) cases where the source-observer distance \( r \) is given, respectively, by:

\[
\begin{align*}
 r & = |x - x_0| \quad \text{and} \quad r = \sqrt{(x - x_0)^2 + (y - y_0)^2}
\end{align*}
\]

As usual, the index "0" denotes a source coordinate.

Finally, it has been found that for graphical purposes it is often convenient to use square waveforms instead of cosine waveforms. This is roughly equivalent to an analog-digital conversion, which is formally expressed as:

\[
\begin{cases}
 f < 0 & \Rightarrow f = -1 \\
 f > 0 & \Rightarrow f = +1
\end{cases}
\]

Obviously, the case \( f = 0 \) corresponds to a mathematical discontinuity where the square wave hasn't a defined value. If disregarded, this abrupt transition, which is treated differently by every software, could introduce some problems during the graphical implementations.

To illustrate these formulas, the graphs in figure 1 depict the 1D case with a source at the origin of coordinates. Then \( x_0 = 0 \), \( r = |x| \) and the waveform can be written as:

\[
f(x) = \cos[2\pi(|x| / \lambda - \tau / T)]
\]

Three "normalized timeshifts" \( \tau / T = 0, 0.25 \) and 0.50 (equivalent to phaseshifts of 0, 90 and 180 degrees, lines A, B, C in figure 1) have been depicted in both, the cosine and the square waveform cases.
If now we superpose a waveform of A-type ($\tau/T = 0.00$) with a waveform of B-type ($\tau/T = 0.25$) shifted of $0.25\lambda$ towards the left ($x_0/\lambda = -0.25$), we get twice the original waveform (constructive interference) for $x/\lambda > -0.25$ and a null waveform (destructive interference) for $x/\lambda < -0.25$. This simple situation, which is shown in figure 1D, corresponds, when generalized to two dimensions, to a classic cardioid pattern.
2. A PRE-COMPUTER ERA DEVICE FOR THE GRAPHICAL IMPLEMENTATION OF THE 2D CASE.

The 2D generalization of the square waveforms in the column II of figure 1 is simply a series of \( N \) concentric circles with radius \( R = R_0 + n\lambda / 2 \), \( n = 0,1,2...N \). The rings between circles correspond to, alternatively, positive and negative values of the square waveform. If your classroom hasn't yet a beamer but still boasts an overhead transparency projector, the best realization is to plot these rings in a transparent sheet and to use, for instance, a bright color for the positive rings and no color (transparent) for the negative ones.

The radius \( R_0 \) and the condition (colored/uncolored) of the inner circle depends on the normalized timeshifts, as clearly shown in figure 1.

For a starter's kit, we need three versions of the same source (let's use a nice primary color like red) with, respectively, phaseshifts of 0, 90 and 180 degrees, and a second source with another primary color (for instance green) just in the 0 degrees phaseshift version. The following table summarizes the properties of the sheets representing these four cases.

<table>
<thead>
<tr>
<th>SHEET'S CODE</th>
<th>TIMESHIFT</th>
<th>INNER CIRCLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \tau / T = 0.00 )</td>
<td>( R_0 = 0.25\lambda )</td>
</tr>
<tr>
<td>R-90 (red&amp;transp.)</td>
<td>( \tau / T = 0.25 )</td>
<td>( R_0 = 0.50\lambda )</td>
</tr>
<tr>
<td>R-180 (red&amp;transp.)</td>
<td>( \tau / T = 0.50 )</td>
<td>( R_0 = 0.25\lambda )</td>
</tr>
<tr>
<td>G-0 (green&amp;transp.)</td>
<td>( \tau / T = 0.00 )</td>
<td>( R_0 = 0.25\lambda )</td>
</tr>
</tbody>
</table>

So, we start with three red sheets R-0, R-90 and R-180 that will represent our first source (isotropic antenna) fed by three currents with relative phases 0, 90 and 180 degrees. Our graphical displays will be based on the optical properties of light and on the additive color rules. Therefore a second antenna source, whose interferences with the first one are to be depicted, will be represented by a fourth sheet G-0, which is a copy of R-0 but with green replacing red.

The figure 2 shows the four transparencies you need. A black dot in the center is added for alignment purposes.
The superposition of a red and a green transparent sheets under the backlight of an overhead projector will result in the following obvious result, pictorially represented in figure 3:

\[ \text{RED WAVE} \]

\[
\begin{array}{ccc}
\text{RED} & \text{WHITE} \\
(+ & (-) \\
\text{GREEN} & \text{BLACK} & \text{GREEN} \\
(+ & (+ & (zero) \\
\text{WHITE} & \text{RED} & \text{WHITE} \\
(- & (zero) & (-)
\end{array}
\]

\text{Fig. 3 Color optical combination and associated waveform values}

Therefore, red-green areas in the superposed sheets are non-radiation zones whereas black&white areas correspond to strong radiation.

Exact superposition of R-0 and G-0 gives a black&white concentric pattern, which means that both sources are adding their strengths in a constructive interference pattern. Exact superposition of R-180 and G-0 gives a red&green concentric pattern, i.e. null fields everywhere.
Now let start the fun. If we consider two in-phase isotropic sources we know well the radiation patterns that we must get for different normalized distances between sources $d/\lambda$. Theoretical results are graphically depicted in the figure 4, with two sources located in the horizontal axis.

![Figure 4](image-url)

- a) $d=0$
- b) $d=0.5\lambda$
- c) $d=1\lambda$
- d) $d=2\lambda$
Similarly, when the phaseshift between the two sources is 90 degrees we obtain the patterns of figure 5.

<table>
<thead>
<tr>
<th></th>
<th>Figure 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a)</td>
<td><img src="d=0" alt="Pattern" /> <img src="d=0.25%5Clambda" alt="Pattern" /></td>
</tr>
<tr>
<td>b)</td>
<td><img src="d=0.5%5Clambda" alt="Pattern" /> <img src="d=1%5Clambda" alt="Pattern" /></td>
</tr>
</tbody>
</table>

Similarly, when the phaseshift between the two sources is 90 degrees we obtain the patterns of figure 5.
Our transparent sheets easily reproduces all the patterns in these figures in a very eye-telling way by just slightly sliding one sheet above the other. For instance, for obtaining results equivalents to figure 4 (sources in phase) we use the sheets R-0 and G-0. Figure 6 is the graphical result you see in your overview projector when you slide R-0 over G-0 for a distance between sources of one wavelength. Similarly, we can mimic all the theoretical results of figure 5 by sliding R-90 over G-0. For instance, figure 7 correspond to the classical cardioid pattern (distance between sources of one quarter of wavelength).

3. APPRAISAL AND CRITIC OF THE OLD WAYS

The graphical power of this humble set of concentric rings is certainly amazing and most teachers are surprised by how easily they evoke forgotten physical concepts among the students and how decisively they will help them to understand array theory. Particularly impressive is the "movie" capacity of these sheets. By slowly sliding one sheet above the other, starting from an "aligned" (sources-at-the-same-point) situation, the spectator experiences a "beamforming live show", where new sidelobes arise and zeros are steered continuously.

However, there are some rotten apples in the basket. Firstly, we are considering square waves of identical amplitudes rather than more realistic sine waves with arbitrary amplitudes. It is hard to imagine how you can use natural additive color combinations to study two sources having different amplitudes. And secondly, it also hard to find color combinations giving good results for more than two sources. We must sadly conclude that, for all its charm, the transparency sheets are limited to an equiamplitude array of two elements.

Now the obvious questions arise: can we replace the sheets by a computer screen? and use algebra or boolean true&false color rules instead of exploiting the natural behavior of light when crossing colored screens?
If the answers are positive, we should be able to overcome the above mentioned limitations of the classical system. But before exploring some promising paths, it must be pointed out that the system of two superposed sheet is a natural parallel computer: the operation symbolized in the table of figure 3 is simultaneously done for all the couple of points in the sheets crossed by the same light ray. This fact is both a warning about the difficulty of mimicking Mother Nature and a hint about the way of doing it. Every source could be considered as a layer in a 2D graphical software where intensities (colors) would be assigned to each pixel in the screen. Combination of two sources would correspond then to the basic operation of mixing two layers by performing an artificial color rule combination, if possible simultaneously on all the screen pixels. This means that if we accept that a screen of 1000 x 1000 pixels can accurately represent our figures, the computer must perform $10^6$ elementary operations fast enough to allow convincing representations of the "gliding/movie" effect, so easily obtained with our transparent sheets.

4. SOME BASIC COMPUTER TRIALS

It is an easy exercise to write a MatLab file illustrating the principles that could be followed for reproducing the superposition of transparencies in a computer screen. Essentially we define x- and y-coordinates of a set of points in a rectangular grid (pixels) They are represented, as usual in MatLab by 2D-arrays $x,y$. Then we compute the polar coordinate $\rho=\sqrt{x^2+y^2}$, we define the normalized time- or phase-shift by the parameter $\text{time\_shift}$ and, finally, the 2D square waveform is simply given by $\text{waveform}=\text{sign}(\cos(2\pi(\rho-\text{time\_shift}))$.

Plotting the variable $\text{waveform}$ against $x,y$ with a standard MatLab graphical routine like $\text{contourf(x,y,waveform)}$ will result in a reasonable depiction of our concentric rings.

There are at least two different ways of introducing spatial phase shifts. If the wave origin is located at $(x_0;y_0)$ the analytical strategy is to define the radial coordinate as $\rho=\sqrt{(x-x_0)^2+(y-y_0)^2}$ and then the 2D array waveform is fully recalculated. But perhaps it is more efficient in a MatLab framework to introduce the space shifts a posteriori by acting directly on the 2D-array waveform. The snag is that only discrete space-shifts (one pixel) can be considered but this drawback could be compensated in some implementations by the easiness of just shifting row and columns in an existing 2D-array waveform. This shift of rows and columns is performed by the MatLab M-function $\text{MAT\_SHIFT}$.

Now, the combination of two waveforms can be easily obtained in a very straightforward way. For instance, the MatLab listing in the Appendix defines two waveforms with parameters:

```
time\_shift=0.00, space\_shift\_x=0.00, space\_shift\_y=0.00; for the 1st waveform \text{(waveform1)} and 
time\_shift=0.25 , space\_shift\_x=0.00, space\_shift\_y=-0.25; for the 2nd waveform \text{(waveform2)}.
```

Then the total waveform is simply: $\text{waveform}=0.5*(\text{waveform1}+\text{waveform2});$.
A direct drawing using `contourf(x,y,waveform)` gives the result of figure 8, which is the MatLab counterpart of figure 7 and convincingly shows the cardioid-like pattern (but here due to the MatLab colormap used (`hot`), constructive interferences appears as white (++) or black (−−), while negative interferences of both types (+−), (−+) appear as red, which is the color assigned to the zero value in this colormap.

**Fig.8 A Matlab generated cardioid pattern (two sources with 90° phaseshift separated by 0.25λ).**
In principle, there shouldn't be any problem to obtain results for more than two sources and, moreover, with arbitrary amplitudes and continuous cosine-like behavior. Also, power diagrams are just obtained by squaring the normalized waveform values before plotting them. Obtaining a similar result with our transparency sheets is almost unthinkable. For instance, if we define the two waveforms:

```
waveform1: time_shift=0.00 , space_shift_x=0.00, space_shift_y=0.00;
waveform2: time_shift=0.00 , space_shift_x=0.00, space_shift_y=1.00;
```

and then plot the squared sum
```
waveform=0.50*(waveform1+waveform2);
waveform=waveform.^2;
```

we obtain the power pattern of figure 9 equivalent to the case $d=1\lambda$ in figure 4. In figure 9, since we are considering square waves, the squared amplitude (power) can only take the binary zero/one values that the MatLab color map "hot" translates as black&white, thus generating very nice figures, that in addition of their scientific merit could be perceived by many as of artistic interest.

![Fig.9 A Matlab generated power pattern (two sources in phase separated by 1.00\(\lambda\))](image)
5. CONCLUSIONS

A simple set of transparencies allow a direct graphical demonstration of the fact that antenna array patterns are the fruit of wave interferences. The most impressive use of these transparencies is obtained when they are slowly glided one upon another and the overhead projector shows a movie-like effect, where radiation pattern nulls are steered and new lobes are generated. But it is practically impossible to reproduce the combination of more than two sources and both must have equal amplitude. These restrictions can be eliminated by mimicking the superposition of these transparencies with a software code. Using convenient false color rules to combine the waves, the computer screen can become a virtual version of the overhead projector, where there is no limitation in the number of sources, their relative amplitudes and phases, and in the nature of the depicted quantity (amplitude, power or other). This has been convincingly demonstrated by writing a simple MatLab code given in the appendix. Of course, this simple code can be greatly improved, especially by using more sophisticated drawing tools, either available as standard Matlab routines or specially tailored for this kind of representation. The reader is encouraged to do so, the goal here was at the proof-of-concept level.

The true challenge is to develop a software code with a movie capacity that convincingly mimics in real time the transparency sliding effect. Obviously, the sought-after code must act directly on the computer screen pixels and therefore should be written using Open-GL or any other equivalent programming language which can act directly at the graphical card level.

The author humbly acknowledges that he lacks the needed experience and know-how. But he hopes that this challenge either has been already met (please let me know it), or will push more knowledgeable colleagues and computer graphics mavericks to provide a convincing answer.

The availability of such a code will allow the replacement of transparency sheets, overhead projectors and human hands by applets, beamers and non-animal mice, but without losing the intuitive appeal and convincing appearance of the classical demonstration.

5. APPENDIX: THE MATLAB CODE

```matlab
% Rough simulation of the interference patterns obtained by physical superposition
% of two transparencies where sets of concentric rings are drawn.
% Juan Mosig, EPFL Lausanne, Switzerland, December 2003
% juan.mosig@epfl.ch

clear all

% Defining a linear cartesian grid of points (x,y)
% Coordinates (x,y) are normalized respect to wavelength
% The number of points per wavelength is PPW and the grid spans
% the values [-MW;MW] along x and [-NW;NW] along y.
% Hence the number of points is 2*MW*PPW+1 along x and 2*NW*PPW+1 along y
mw=5;
w=5;
```

ppw=20;
 m=mw*ppw;
 n=nw*ppw;
 x=(-m:1:m)/ppw;
 y=(-n:1:n)/ppw;
 [x,y]=meshgrid(x,y);

% Defining the polar coordinate RHO
rho=sqrt(x.*x+y.*y);

% Defining a set of radial RHO-dependent waveforms. Every waveform is defined as
% a matrix of dimensions (2*MW*PPW+1;2*NW*PPW+1) and the matrix elements depend
% on three parameters:
% 1) the phase or time shift: TIME_SHIFT
% 2) the space shift along x: SPACE_SHIFT_X
% 3) the space shift along y: SPACE_SHIFT_Y

% The time shift (in fractions of period) is directly introduced in the
% analytical expression of the waveform:

time_shift=0.00
waveform1=sign(cos(2*pi*(rho-time_shift))); % This is a square waveform
% waveform1=cos(2*pi*(rho-time_shift)); % This is a cosine waveform

% On the other hand, the space shifts are applied by shifting adequately
% the rows and columns of the matrix representing the waveform.
% This mimics the physical operation of gliding a transparency sheet above another one

% Matlab has probably a built-in function for this shifting operation, but if you, like me,
% don't know it, just create your own. Here it is the M-function MAT_SHIFT
space_shift_x=0.00;
space_shift_y=0.00;
ishift=space_shift_x*ppw
jshift=space_shift_y*ppw
waveform1=mat_shift(waveform1,ishift,jshift);

% Now repeat the process for the waveform #2:

time_shift=0.25
waveform2=sign(cos(2*pi*(rho-time_shift))); % This is a square waveform
space_shift_x=0.00;
space_shift_y=-0.25;
ishift=space_shift_x*ppw
jshift=space_shift_y*ppw
waveform2=mat_shift(waveform2,ishift,jshift);
% Finally, combine additively the waveforms to obtain their superposition

waveform=.5*(waveform1+waveform2);

% And plot the results

contourf(x,y,waveform)
axis equal
colorbar
colormap(hot)

function shiftmat=mat_shift(matrix,krow,kcol)

% MAT_SHIFT shifts a matrix MATRIX(M,N)of KROW along rows
% and KCOL along columns. The empty positions are filled with zeros.

% Row #I becomes row #I+KROW. Column #J becomes column J+KCOL
% Negative values of KROW and KCOL are possible but
% KROW <= M and KCOL<=N. Also null values of KROW and KCOL
% simply leave the matrix unchanged.

% I wonder if there is a Matlab library function for this
% JR Mosig, vs1.0, 180102

%************************************EXAMPLE**************************************

%>> a=rand(4,5)
%a =
% 0.5028  0.1897  0.5417  0.8600  0.8998
% 0.7095  0.1934  0.1509  0.8537  0.8216
% 0.4289  0.6822  0.6979  0.5936  0.6449
% 0.3046  0.3028  0.3784  0.4966  0.8180

%>> shiftmat=mat_shift(a,2,-1)
%shiftmat =
% 0 0 0 0 0
% 0 0 0 0 0
% 0.1897 0.5417 0.8600 0.8998 0
% 0.1934 0.1509 0.8537 0.8216 0

[m,n]=size(matrix);
shiftmat=matrix;

% Shifting rows

if(krow<0)
    addrow=zeros(abs(krow),n);
    shiftmat=[matrix(1-krow:m,:) ; addrow];
if(krow>0)
    addrow=zeros(abs(krow),n);
    shiftmat=[addrow ; matrix(1:m-krow,:)];
end

% Shifting columns

if(kcol<0)
    addcol=zeros(m,abs(kcol));
    shiftmat=[shiftmat(:,1-kcol:n) addcol];
end
if(kcol>0)
    addcol=zeros(m,abs(kcol));
    shiftmat=[addcol shiftmat(:,1:n-kcol)];
end