An Old Tool and a New Challenge for Depicting Antenna Array Radiation Patterns

being also

A Classroom Demonstration for the Principle of Wave Interference Underlying Antenna Array Theory

Juan R. Mosig

LEMA-EPFL
Station 11, CH-1015, Lausanne, Switzerland
E-mail: juan.mosig@epfl.ch

Abstract

This paper is intended for teachers and students dealing with antenna engineering. Graphical tools have historically been used for illustrating many basic features of radiation patterns, such as the position and number of secondary lobes, null angles, and beam steering. This paper describes one of these graphical devices, which 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 modern computer software, such as *MATLAB*.

Keywords: Antennas; antenna radiation patterns; antenna arrays; electrical engineering education; interference; electromagnetic waves

1. Introduction

Classic antenna-array theory, as introduced in most standard antenna textbooks [1-4], is the perfect mathematical tool for obtaining radiation patterns through a quantitative description of wave-interference phenomena. However, unfortunately, the basic facts allowing the intuitive understanding of how antenna-array patterns arise from essential interference phenomena are frequently obscured by sophisticated expositions and technical treatments, which rely too much on the mathematical properties of complex exponential functions. This state of affairs frequently discourages students, who no longer possess the trigonometric skills of their elders.

However, many basic features of radiation patterns (e.g., the position and number of secondary lobes, null angles, beam steering) can always be successfully explained with graphical tools similar to those that were usual in old-fashioned courses on optics, acoustics, and vibrations and waves. These veteran tools deserve a modest revival.

This paper describes one of these graphical devices, which 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 modern computer software, such as *MATLAB*. Although surely not the best language to obtain impressive graphical representations, *MATLAB* can easily be used for a proof-of-concept trial, and to pave the way for more sophisticated approaches.

The following developments are strongly rooted in the author's acquaintance with the introductory physics courses that were compulsory reading for engineering students in the early seventies. F. S. Crawford at Berkeley [7], R. Feynman at CalTech [8], and A. P. French at MIT [9] were the authors of some of these memorable courses, which frequently provided a layman's approach and plenty of home experiences for enticing the young reader. Together with other classic books on waves [10, 11], they provide useful material for understanding wave interference. Most frequently, the developments in these texts were not based on electromagnetic waves, and it is hard to find the word "antenna" inside them. However, they contain the basic staples for a physical understanding of antenna patterns.

The younger readers are perhaps not so familiar with these musty old books. However, they can easily check that – at the moment of writing – a Google search for "*wave interference*" provides more than eight million links and more than eight hundred thousand images! It is hard to do any selection, here (bountiful information is the worst enemy of sound information). However, as usual, *Wikipedia* [12] has some good material, including nice graphics and animations. The Google image search yields some applets that sometimes are not so far away from the goal of this paper [13]. Finally, it is worth mentioning than even *YouTube* can be an invaluable source of resources [14].

This paper is intended for teachers and students dealing with antenna engineering. The material presented is not based on any previously existing published material. It has been conceived, tested, and polished by continuous exposure to young students, and

 ISSN 1045-9243/2011/$25©2011 IEEE

careful evaluation of their reactions. However, like almost everything that we naïvely believe original and the sole fruit of a sharp mind, the following developments were probably inspired in a subconscious way by several of the above-mentioned references, and by illustrations and pictorials (more recently, Internet files) encountered during a lifetime of interest about wave phenomena.

1. Basic Concepts

Array theory is based on that elusive concept called a "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 on phase properties, which are the most relevant in interference phenomena. Also, we will start with a snapshot of the phenomena taken at a given time, $t = t_0$ (the possibility of a "movie" including the time variation will be considered later). We will therefore assume, for the sake of simplicity, a unit amplitude, $A = 1$, and we will reformulate our basic expression as

$$f(r) = \cos\left[2\pi(r/\lambda - \tau/T)\right],$$

where we have introduced the normalized distance in terms of wavelengths, $r/\lambda = kr/2\pi$, and a normalized time shift or phase shift 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 distances, r, are respectively given by

$$r = |x - x_0|$$

and

$$r = \sqrt{(x-x_0)^2 + (y-y_0)^2}.$$

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 sinusoidal waveforms. This is roughly equivalent to an analog-to-digital conversion, which is formally expressed as

$$\left\{\begin{array}{l}\text{if } f(r,t) < 0 \;\Rightarrow\; f = -A \\ \text{if } f(r,t) > 0 \;\Rightarrow\; f = +A\end{array}\right\}.$$

The mathematically minded reader will recognize here a close connection with the function "sign." In mathematics, by definition, $\text{sign}(0) = 0$. In practice, the case $f = 0$ corresponds to a abrupt transition, where the square wave does not have a defined value. Also, if two waves have the same amplitudes, $A = B$, then $\text{sign}(A - B) = 0$. However, if some numerical noise exists in A and B, then $\text{sign}(A - B)$ may become a random quantity. If disregarded, these situations (which are treated differently by every software package) could introduce some problems during the graphical implementations.

To illustrate these formulas, we start by considering a basic one-dimensional case, with a source at the origin of coordinates, and no time shift. Then, $r = |x|$, $x_0 = 0$, $\tau/T = 0$, and the waveform (Figure 1a) can be simply written as

$$f(x) = \cos\left[2\pi|x|/\lambda\right].$$

We now introduce a "normalized time shift," $\tau/T = 0.25$ (equivalent to a phase shift of 90°) in the original wave. The new "delayed" wave (Figure 1b) is given by the equation

$$f(x) = \cos\left[2\pi\left(|x|/\lambda - 0.25\right)\right].$$

We next space shift the wave origin by shifting by 0.25λ towards the left ($x_0/\lambda = -0.25$).

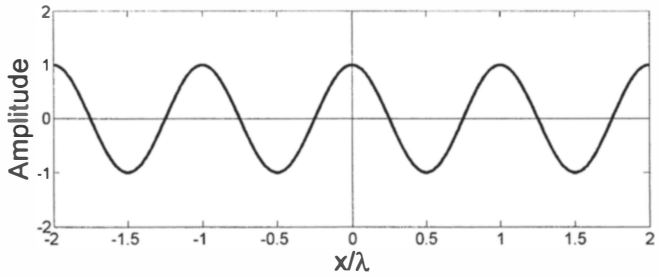

Figure 1a. A cosinusoidal wave originating at $x = 0$ and propagating towards both sides of the horizontal axis.

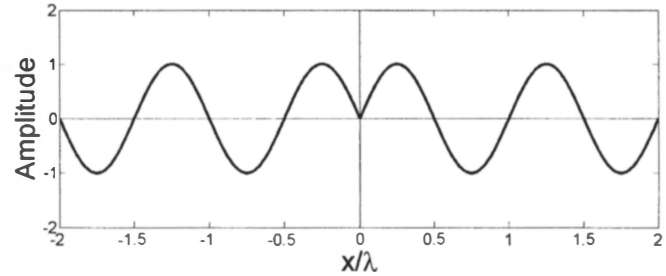

Figure 1b. The wave in Figure 1a phase shifted by 90°.

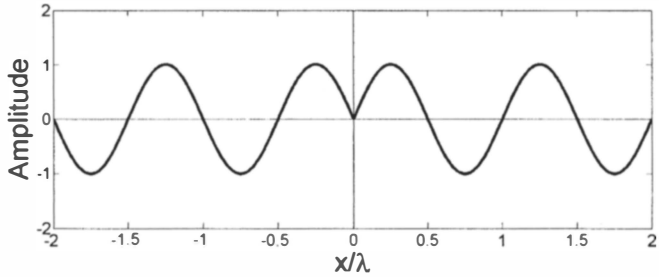

Figure 1c. The wave in Figure 1b space shifted towards the left by a quarter wavelength.

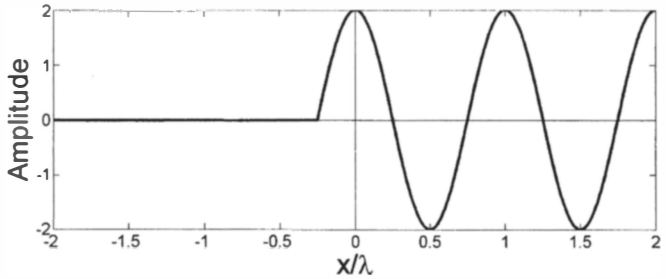

Figure 1d. The combination of the waves in Figures 1a and 1c leading to constructive-destructive interferences.

The equation of this delayed and displaced wave (Figure 1c) is now

$$f(x)=\cos\left[2\pi\left(|x+0.25|/\lambda-0.25\right)\right].$$

If now we superpose the "a" waveform ($\tau/T=0.00$; $x_0/\lambda=0.00$) with the "c" waveform, which has been shifted both in time and in space ($\tau/T=0.25$; $x_0/\lambda=-0.25$), we get twice the original waveform (constructive interference) for $x/\lambda>-0.25$, and a null waveform (destructive interference) for $x/\lambda<-0.25$, as shown in Figure 1d. In the next sections, it is shown that this simple one-dimensional behavior generalizes in two dimensions to a cardioid, as De Castillon appropriately named that nice heart-shaped curve in 1741 [15].

2. A Pre-Computer-Era Device for the Graphical Implementation of the 2D Case

The two-dimensional (2D) generalization of the square waveforms presented in Figure 2 is simply a series of N concentric circles, with radii $R=R_0+n\lambda/2$, $n=0,1,2...N$. The rings between the circles alternatively correspond to positive and negative values of the square waveform. If your classroom doesn't yet have a "beamer" but still boasts of an overhead transparency projector, the best realization is to plot these rings on a transparent sheet, and to use (for instance) a bright color for the positive rings and no color (transparent) for the negative rings. The radius, R_0, and the condition (colored/uncolored) of the inner circle depend on the normalized time shifts, 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 phase shifts of 0°, 90°, and 180°, and a second source with another primary color (for instance, green), in just the 0° degree phase-shift version. The Table 1 summarizes the properties of the sheets representing these four cases.

We thus start with three red sheets, R-0, R-90, and R-180, which will represent our first source (an isotropic antenna) fed by three currents with relative phases 0°, 90°, and 180°. Our graphical displays will be based on the optical properties of light, and on the additive color rules. A second antenna source – the interferences of which with the first source are to be depicted – will therefore be represented by a fourth sheet, G-0, which is a copy of R-0 but with green replacing red. Figure 3 shows the four transparencies you need. A black dot in the center is added for alignment purposes.

The superposition of red and green transparent sheets under the backlight of an overhead projector will result in the obvious result pictorially represented in Figure 4. Therefore, red and green

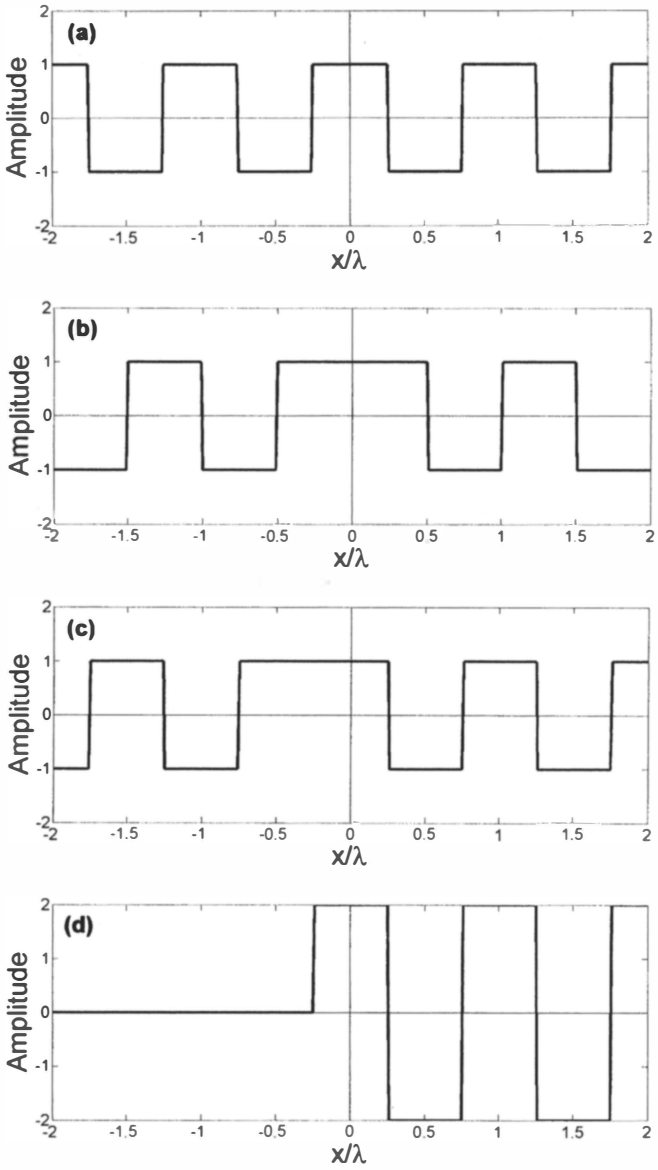

Figure 2. This is a version of the four situations depicted in Figure 1, but using square rather than sinusoidal waves.

areas in the superimposed sheets are non-radiation zones, whereas black and white areas correspond to strong radiation. Exact superposition of R-0 and G-0 gives a black-and-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 and green concentric pattern, i.e., null fields everywhere.

To check the validity of the results that will be obtained with our graphical procedure, we need to generate some rigorous results, providing a good benchmark for comparisons.

Table 1. The four needed transparent sheets and their properties.

Sheet's Code	Time Shift	Inner Circle	
		Radius	Color
R-0 (red & transparent)	$\tau/T=0.00$	$R_0=0.25\lambda$	red
R-90 (red & transparent)	$\tau/T=0.25$	$R_0=0.50\lambda$	red
R-180 (red & transparent)	$\tau/T=0.50$	$R_0=0.25\lambda$	transparent
G-0 (green & transparent)	$\tau/T=0.00$	$R_0=0.25\lambda$	green

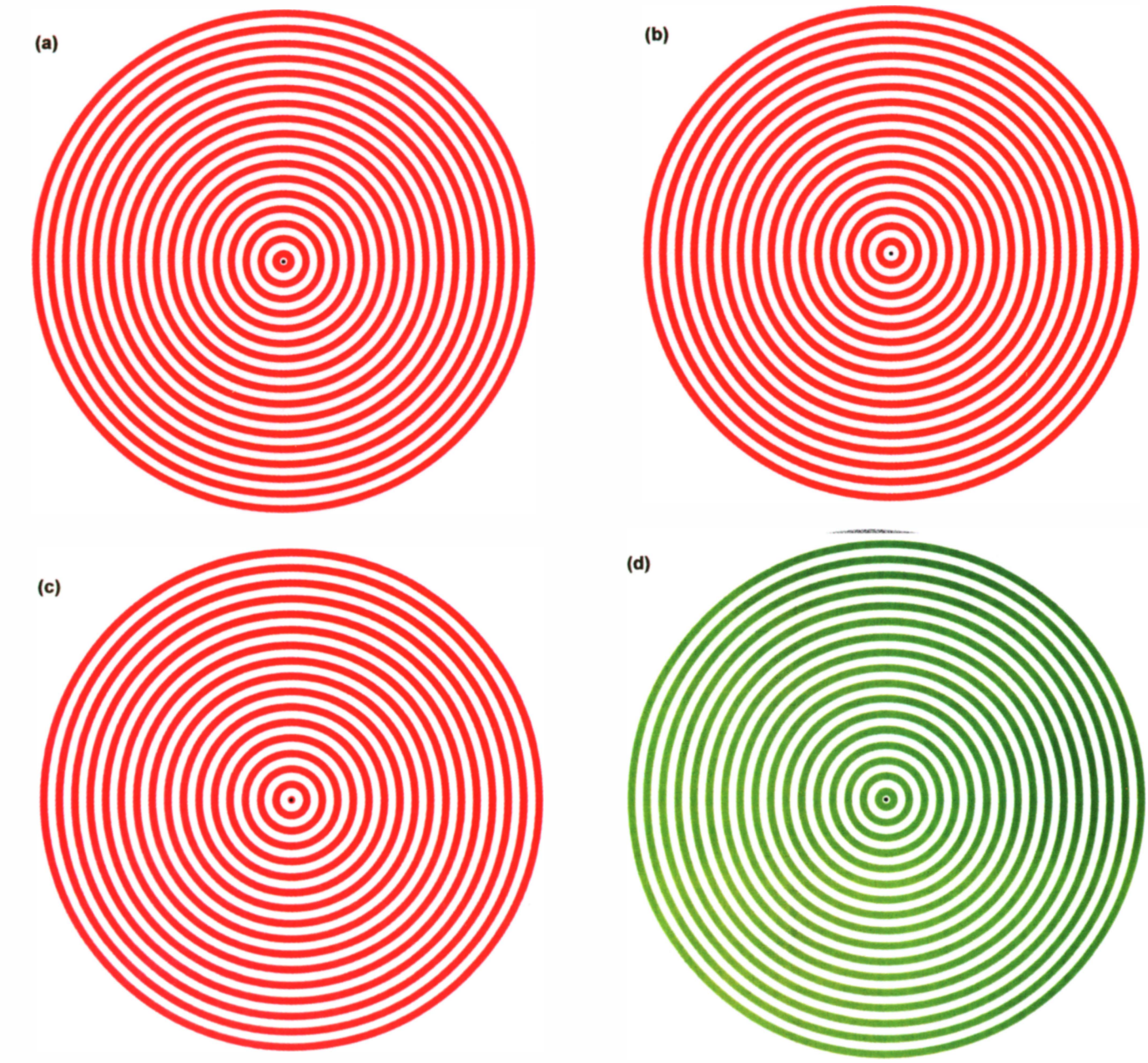

Figure 3. A practical realization of the four transparent sheets described in Table 1: (a) R-O; (b) R-90; (c) R-180; (d) G-0.

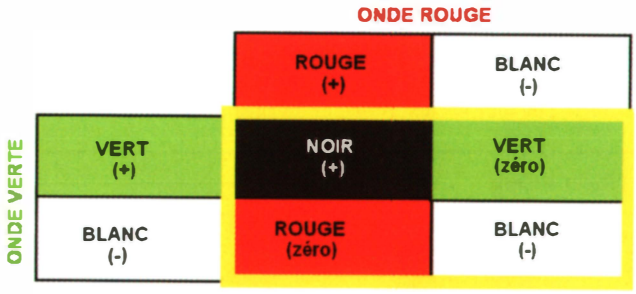

Figure 4. The color optical combinations and associated wave-form values.

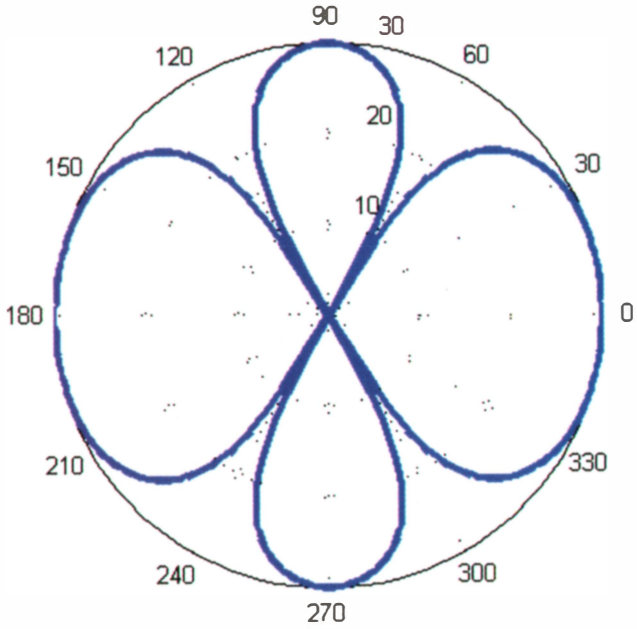

Figure 5a. The radiation patterns of two isotropic in-phase sources separated by one wavelength.

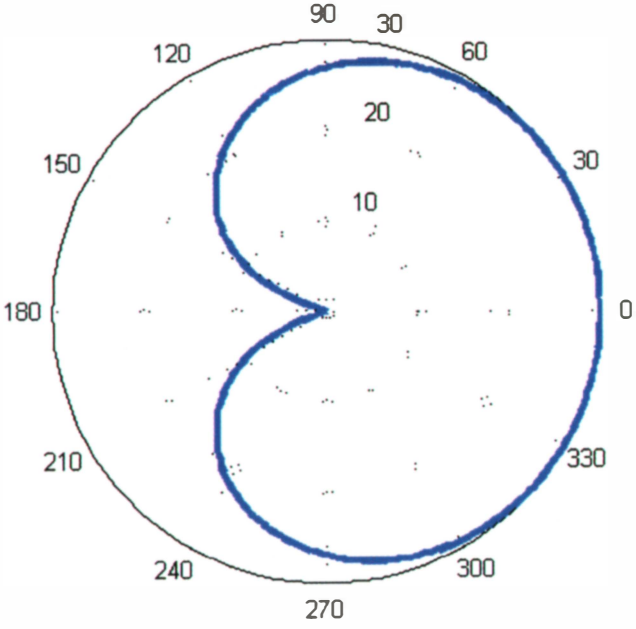

Figure 5b. The radiation patterns of two isotropic sources separated by one-quarter wavelength and with a 90° phase shift between them.

If we consider two isotropic sources, it is straightforward to predict with array theory [1-6] the radiation patterns that we must get for different normalized distances between sources and different phase shifts between them. For comparison purposes, two theoretical results are considered here. They have been obtained with a simple implementation of the array theory principles in *MATLAB*, limited to the specific two-source two-dimensional problem (see the Appendix).

Although not really necessary here (but diehard antenna engineers will surely appreciate), the radiation patterns are given in dB, and cut at –30 dB with respect to the maximum level. In Figure 5a, the sources are in phase, located on the horizontal axis, and separated by one wavelength. The result is a nice asymmetric four-petal flower. In Figure 5b, the two same sources are now phase shifted by 90° and separated by a quarter wavelength. The well-known result is a cardioid shape (in fact, a logarithmic version of it).

Now let the fun start.

Our transparent sheets easily reproduce the patterns in Figures 5a-5b in a very eye-catching way, by just slightly sliding one sheet above the other. For instance, to obtain results equivalent to Figure 5a (sources in phase), we use sheets R-0 and G-0. Figure 6a is the graphical result you see in your overview projector when you slide R-0 over G-0 for a distance of one wavelength between sources. Our asymmetric flower is easily seen, with the separation between lobes/petals (the pattern nulls) particularly conspicuous.

Similarly, we can mimic the theoretical results of Figure 5b by sliding R-90 over G-0. The expected classical cardioid pattern is easily recognizable in Figure 6b (maximum to the right, zero to the left, intermediate values towards the top and bottom).

3. Appraisal and Critique of the Old Ways

The graphical power of this humble set of concentric rings is certainly amazing. Most teachers are surprised by how easily they evoke forgotten physical concepts among the students, and how decisively they 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 "beam-forming live show," where new sidelobes arise and zeros are continuously steered.

However, all that glitters is not gold (often have you heard that told). 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 could use natural additive color combinations to study two sources having different amplitudes. Secondly, it is also hard to find color combinations that give good results for more than two sources. We must sadly conclude that for all its charm, the transparency sheets are limited to an equi-amplitude array of two elements.

Now the obvious questions arise: can we replace the sheets by a computer screen, and can we use algebra or Boolean true and 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. However, before exploring some promising paths, it must be pointed out that the system of two superimposed sheets is a natural parallel computer: the operation symbolized in the table of Figure 3 is simultaneously done for all the couples 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 two-dimensional graphical software, where intensities (colors) would be assigned to each pixel on the screen. A combination of two sources would then correspond 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 × 1000 pixels can accurately represent our figures, the computer must perform 10^6 elementary operations fast enough

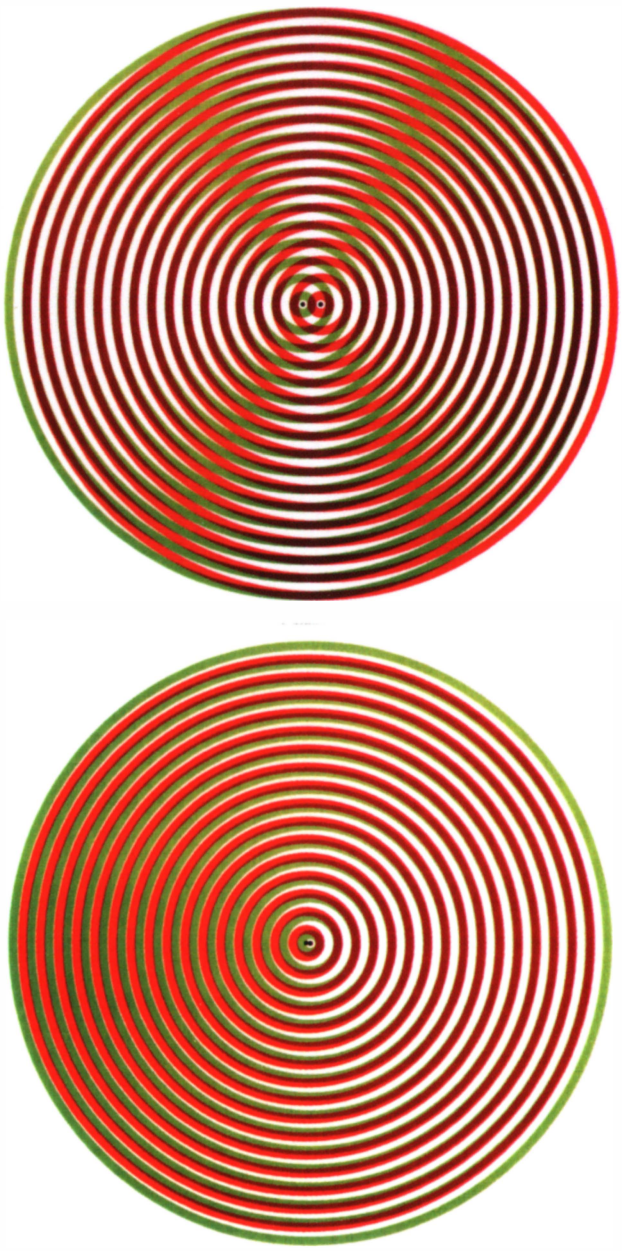

Figure 6. Two realistic patterns obtained with the transparent sheets: (a) Two in-phase sources separated by one wavelength (compare with Figure 5a); (b) A cardioid pattern (compare with Figure 5b).

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 to reproduce the superposition of transparencies on a computer screen. We essentially define x and y coordinates of a set of points in a rectangular grid (pixels). As usual in *MATLAB*, they are represented by two-dimensional arrays `x,y`. We then compute the polar coordinate, `rho=sqrt(x.*x+y.*y)`, we define the normalized time or phase shift by the parameter `time-shift`, and, finally, the two-dimensional square waveform is simply given by

```
waveform=sign(cos(2*pi*(rho-time-shift)))
```

Plotting the variable `waveform` against `x,y` with a standard *MATLAB* graphical routine such as `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 `(x0;y0)`, the analytical strategy is to define the radial coordinate as

```
rho=sqrt((x-x0).*(x-x0)+(y-y0).*(y-y0))
```

and then the two-dimensional array `waveform` is fully recalculated. However, it is perhaps more efficient in a *MATLAB* framework to introduce the space shifts a posteriori by acting directly on the two-dimensional array `waveform`. The snag is that only discrete space shifts (one pixel) can be considered, but this drawback could be compensated for in some implementations by the easiness of just shifting rows and columns in an existing two-dimensional array `waveform`. This shift of rows and columns is performed by the *MATLAB* `M`-function `MAT_SHIFT`.

The combination of two waveforms can now 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 first waveform (`waveform1`), and

```
time_shift=0.25 , space_shift_x=0.00,
space_shift_y=-0.25;
```

for the second waveform (`waveform2`). The total waveform is then simply

```
waveform=.5*(waveform1+waveform2);.
```

A direct drawing using `contourf(x,y,waveform)` gives the result of Figure 7, which is the *MATLAB* counterpart of Figure 6b. It convincingly shows the cardioid-like pattern. However here, due to the *MATLAB* colormap used (`hot`), constructive interferences appear as white $(++)$ or black $(--)$, while negative interferences

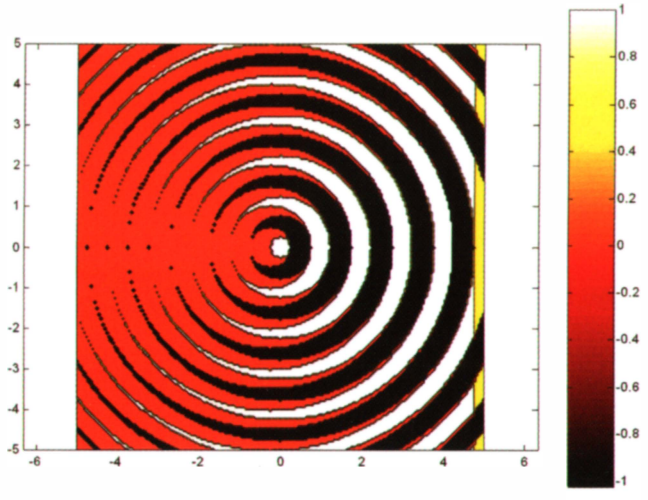

Figure 7. A *MATLAB*-generated cardioid pattern (two sources with 90° phase shift separated by 0.25λ. This should be compared with Figures 5b and 6b.

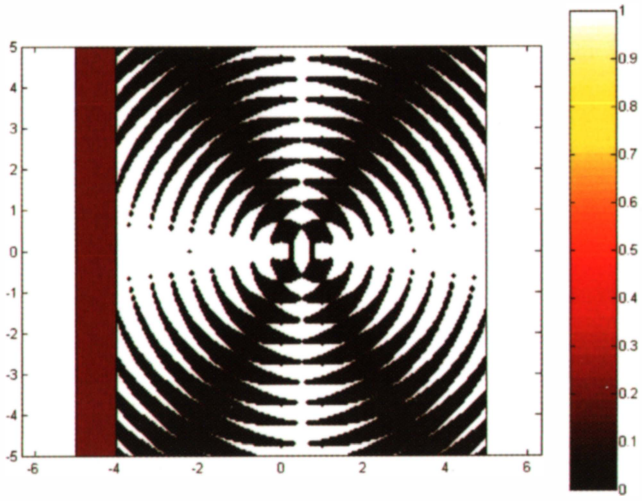

Figure 8. A *MATLAB*-generated power pattern (two sources in phase separated by 1.00λ. This should be compared with Figures 5a and 6a.

of both types $(+-)$, $(-+)$ appear as red, which is the color assigned to the zero value in this color map.

The code used to generate this Figure 7 has the merit of its simplicity, but is far from being perfect. In any case, it is obviously not well suited to generating the sought-after movie effect. For instance, an obvious limitation comes from the homemade *MATLAB* function `mat_shift`, used to simulate the spatial shift of one of the waves by shifting the values of a two-dimensional array and padding the extra spaces with zeros. As a result, a thin vertical bar with obviously wrong colors appears at the extreme right of the figure. An alternative could be the *MATLAB* intrinsic function `circshift`, but no real improvement is obtained in either graphical aspect or in computer time.

In principle, there shouldn't be any problem in obtaining results for more than two sources, and, moreover, with arbitrary amplitudes and continuous cosine-like behavior. Also, power diagrams are obtained just 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 8, equivalent to the case $d=1\lambda$ in Figure 4. In Figure 8, 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 and white, thus generating very nice figures reminiscent of bygone op-art graphics. In addition to their scientific merit, these could be perceived by many as being of artistic interest (if you were born after the sixties, check [15]). Again, the red vertical bar at the left is the result of padding with zeros the empty positions introduced by a spatial shift.

5. Conclusions

A simple set of transparencies allow a direct graphical demonstration of the fact that antenna array patterns are the obvious fruit of wave interference. 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. However, 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 to the number of sources, their relative amplitudes and phases, and in the nature of the depicted quantity (amplitude, power, or other). Hopefully, this has been convincingly demonstrated by writing the 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 should ideally act directly on the computer-screen pixels, and therefore should be written using *Open-GL* or any other equivalent programming language that can directly drive the computer's graphics card. The author humbly acknowledges that he lacks the needed experience and know-how, but he hopes that either this challenge has been already met (please let me know if this is the case), or that it will push more knowledgeable colleagues and computer-graphics mavericks to provide a convincing answer.

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

6. Acknowledgements

The author wants to thank Roberto Torres (EPFL Switzerland), Prof. Ari Sihvola (Aalto University, Finland), and Prof. Yozan D. Mosig (University of Nebraska at Kearney, USA) for invaluable discussions, suggestions, and help in preparing this paper. Last but not least, appreciative thanks go to the two anonymous reviewers of this paper, who not only made pertinent corrections but personally tested the *MATLAB* codes and suggested possible improvements.

7. References

1. C. A. Balanis, *Antenna Theory, Analysis and Design, Second Edition*, New York, Wiley, 1996.

2. L. Blake and M. W. Long, *Antennas: Fundamentals, Design, Measurement, Third Edition*, Raleigh, NC, SciTech Publishing, 2009.

3. R. C. Johnson (ed.), *Antenna Engineering Handbook, Third Edition*, New York, McGraw-Hill, 1993.

4. J. D. Kraus, *Antennas, Second Edition*, New York, McGraw-Hill, 1988. (There is also a modern revised edition: J. D. Kraus and R. J. Marhefka, *Antennas: For All Applications, Third Edition*, New York, McGraw Hill, 2001.

5. T. A. Milligan, *Modern Antenna Design*, New York, McGraw-Hill, 1985.

6. W. L. Stutzman and G. A. Thiele, *Antenna Theory and Design, Second Edition*, New York, Wiley, 1998.

7. F. S. Crawford, *Waves (Berkeley Physics Course, Volume 3)*, New York, McGraw-Hill, 1968.

8. R. P. Feynman, R. B. Leighton and M. Sands, *The Feynman Lectures on Physics*, Addison-Wesley, New York, 1964; see in particular Volume 1, Chapter 28, "Electromagnetic Radiation," and Chapter 29, "Interferences;" see also http://www.feynmanlectures.info/

9. A. P. French, *Vibrations and Waves*, New York, W. W. Norton & Company, 1971.

10. W. C. Elmore and M. A. Heald, *Physics of Waves*, New York, Dover Publications, 1985.

11. J. R. Pierce, *Almost All About Waves*, New York, Dover Publications, 2006.

12. http://en.wikipedia.org/wiki/Interference_(wave_propagation)

13. http://www.phy.ntnu.edu.tw/ntnujava/index.php?board=3.0

14. http://www.youtube.com/watch?v=5PmnaPvAvQY&feature=related

15. http://mathworld.wolfram.com/Cardioid.html

16. http://arthistory.about.com/cs/arthistory10one/a/op_art.htm

8. Appendix: The *MATLAB* Codes

8.1 Code for Generating Figures 5a and 5b

```
% Simple polar logarithmic plot of a two-sources
% radiation pattern using basic array theory.

% Juan Mosig, EPFL Lausanne, Switzerland,
% December 2009
% juan.mosig@epfl.ch
clear all
phi_deg=linspace(0,360,361);
phi=phi_deg*pi/180; % Mathematical angles in
     radians

% Field of two isotropic sources separated by a
% normalized distance KD and with a phaseshift
% ALPHA between them
kd=0.5*pi
alpha=-0.5*pi
ephi=1+exp(1i*(kd*cos(phi)+alpha));
ephi_db=20*log10(abs(ephi))     %  Value in dB

% Normalizing the fields (maximum value 0dB)
max_ephi_db=max(ephi_db)
ephi_db_norm = ephi_db - max_ephi_db

% Truncating the fields. Values below LEVEL are
set to LEVEL
level=-20
flag=ephi_db_norm < level;
ephi_db_trunc=ephi_db_norm.*(1-flag)+level*flag

% Decibel plot in polar coord.
figure
polar(phi,ephi_db_trunc-level,'b')
grid on
```

8.2 Code for Generating Figures 7 and 8

```
% Rough simulation of the interference patterns
% obtained by physical superposition
% of two transparencies where sets of concentric
% colored rings are drawn.
% This script calls the external MatLab function
% mat_shift

% Juan Mosig, EPFL Lausanne, Switzerland,
% December 2009
% 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;
nw=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. See the
% homemade 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)));
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.
% The library function circshift doesn't provide
% better results.
% Juan Mosig, EPFL Lausanne, Switzerland,
% December 2009
% juan.mosig@epfl.ch

% EXAMPLE
%If we define the matrix a as
%  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
%Then shiftmat=mat_shift(a,2,-1) yields:
%       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];
end
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)];
```