A MATLAB INTERFACE FOR ANALYZING CONFORMAL ARRAYS COMPOSED OF POLARIZED HETEROGENEOUS ELEMENTS

J. C. Brégains, J. A. García–Naya, M. González–López, L. Castedo–Ribas.

Electronic Technology and Communications Group
Faculty of Informatics, University of A Coruña
Campus de Elviña s/n – CP 15071 – A Coruña – Spain
E–mails: (Brégains) julio.bregains@udc.es – (García–Naya) jagarcia@udc.es
(González–López) mgonzalezlopez@udc.es – (Castedo–Ribas) luis@udc.es

ABSTRACT

As an extension and improvement of a previous work, this paper presents a graphical interface, designed with MATLAB®, which is capable of represent both the spatial and electrical configurations of a conformal antenna array, as well as the different patterns radiated by it. The tool considers the polarization of the elements of the array, not necessarily congruent to each other.

1. INTRODUCTION

In spite of the fact that conformal arrays constitute a well studied subject in the technical literature of antenna designers [1], nowadays it is not so easy to find a free tool capable of visualizing both the configuration of such arrays and their corresponding radiated patterns. In fact, and as far as the authors know, the currently available tools correspond to commercial electromagnetic simulators that make use of well known numerical techniques (Finite Difference Time Domain, Method of Moments, etc.), or a recent toolbox from MATLAB® called “Phased Array System Toolbox”. But, unfortunately for many investigators that own MATLAB®, neither of them is freely available [2]. In this sense, in a previous work [3] we presented an alternative, free, MATLAB®–based solution, which allowed the designers to visualize not only the elements spatial arrangement, but also the electrical configuration (amplitude and phase distributions), and the radiated fields generated by such an arrangement. The main drawback of this tool lies in that, to calculate the resulting power pattern, it performs the superposition of the fields emitted by the individual antennas by means of scalar addition, which is inadequate when using polarized elements, something that in practice represents the majority of conformal layouts, unless the polarization axes of the elements are aligned, as in the case of parallel dipoles, for example. As a consequence of this, such a tool does not adequately characterize the F^2 pattern of an array of

elements with unmatched polarization directions. This also prevents the tool from representing the vertical F_θ^2 and horizontal F_ϕ^2 components of the field.

In this paper we propose a solution that deals efficiently with the abovementioned problems. Furthermore, the new design has a graphical user interface that facilitates not only the loading, from text files, of the conformal array configurations, but also the handling of the generated patterns features.

2. MATHEMATICAL DETAILS

A large part of the description of the conformal array mathematical models that will be given here can be obtained from [3], taking basically into account that now the resulting vector field $\mathbf{F}\ \theta,\phi$ radiated by the array to a global far–field point θ,ϕ corresponds to the superposition of the individual element fields, considering that, on the one hand, each one has a specific polarization, on the other hand they are not necessarily congruent[1], and finally that the mutual coupling between them is negligible. All of this can be stated mathematically by the following vector summation:

$$\mathbf{F}\ \theta,\phi\ = \sum_{n=1}^{N} \mathbf{f}_n\ \theta_n,\phi_n\ I_n e^{j\mathbf{k}\cdot\mathbf{r}_n} = \sum_{n=1}^{N} \mathbf{a}_{fn} f_n\ \theta_n,\phi_n\ I_n e^{j\mathbf{k}\cdot\mathbf{r}_n} \qquad (1)$$

In this equation, $\mathbf{a}_{fn}$ represents the polarization unit vector of element *n*, whereas $I_n = |I_n| e^{j\Phi_n}$ and $\mathbf{r}_n$, are its excitation phasor and its position vector, respectively –the latter being specified in the global coordinate system. The $\mathbf{k}$ vector is equal to $2\pi/\lambda\ \mathbf{a}_R$, where λ is the working wavelength, with $\mathbf{a}_R$ being the usual radial unit vector, specified in spherical coordinates and pointing from the origin of the coordinate system to the angular position of the field point.

Each element has associated a local coordinate system represented by the vector (of vectors) $\mathbf{a}_n = \left[\mathbf{a}_{xn}\ \mathbf{a}_{yn}\ \mathbf{a}_{zn}\right]^\tau$ (τ indicates transposition). For each of these axes, it is possible to specify its direction angles. Therefore, the $\mathbf{a}_{xn}$ axis, for example, has associated with it the angular tuple $\left[\alpha_{xn}\ \beta_{xn}\ \gamma_{xn}\right]$, which leads, after calculating their cosines, to its components with respect to the global coordinate system triad $\mathbf{a} = \left[\mathbf{a}_x\ \mathbf{a}_y\ \mathbf{a}_z\right]^\tau$, namely:

$$\mathbf{a}_{xn} = \cos\alpha_{xn}\mathbf{a}_x + \cos\beta_{xn}\mathbf{a}_y + \cos\gamma_{xn}\mathbf{a}_z = \left[\cos\alpha_{xn}\ \cos\beta_{xn}\ \cos\gamma_{xn}\right]\left[\mathbf{a}_x\ \mathbf{a}_y\ \mathbf{a}_z\right]^\tau \qquad (2)$$

[1] Two elements *p* y *q* are congruent if, when they are –not simultaneously– located at the global coordinates center, and their local axis system (see text) are aligned with the global axis one, then $\mathbf{f}_p(\theta,\phi) = \mathbf{f}_q(\theta,\phi)$.

Both unit vector triads are then related by $\mathbf{a}_n = \mathbf{T}_n \mathbf{a}$, where $\mathbf{T}_n$ is the direction cosines matrix corresponding to element *n* [3]. Conversely:

$$\mathbf{a} = \left[\mathbf{a}_x \ \mathbf{a}_y \ \mathbf{a}_z \right]^{\tau} = \left(\mathbf{T}_n \right)^{-1} \mathbf{a}_n \tag{3}$$

The zenithal and azimuthal angles measured with respect to the *n*–th element's system can be found by means of the expressions [3]:

$$\theta_n = arccos\left(\mathbf{a}_R \cdot \mathbf{a}_{zn} \right);\ \phi_n = arc\,tan\left(\frac{a_{\rho y_n}}{a_{\rho x_n}} \right), \text{with } \mathbf{a}_{\rho n} = \frac{\mathbf{a}_R - \mathbf{a}_{Zn} \cos\theta_n}{sin\,\theta_n} \tag{4}$$

The vector $\mathbf{a}_{\rho n}$ represents the projection of $\mathbf{a}_R$ onto the plane generated by $\mathbf{a}_{xn}$ and $\mathbf{a}_{yn}$, being $a_{\rho x_n}$ and $a_{\rho y_n}$ the components of the former with respect to the latter local unit vectors [3].

As usually the polarization direction $\mathbf{a}_{fn}$ of element *n* is referred to its own coordinate system, it is necessary to use projections (3) to find its components in the global system.

Once the total complex vector field is obtained, its components simply correspond to the projections with respect to the unit zenithal $\mathbf{a}_\theta$ and azimuthal $\mathbf{a}_\phi$ directions:

$$F_\theta\ \theta,\phi\ = \mathbf{F}\ \theta,\phi\ \cdot \mathbf{a}_\theta\ \theta,\phi\ , \quad F_\phi\ \theta,\phi\ = \mathbf{F}\ \theta,\phi\ \cdot \mathbf{a}_\phi\ \theta,\phi \tag{5}$$

Finally, the amplitude of the total field, normalized with respect to the maximum value obtained at point θ_0, ϕ_0, and expressed in decibels, is given by:

$$F_{Norm,dB}\ \theta,\phi\ = 10\,log_{10}\left[\frac{F_\theta\ \theta,\phi\ F_\theta^*\ \theta,\phi\ + F_\phi\ \theta,\phi\ F_\phi^*\ \theta,\phi}{F_\theta\ \theta_0,\phi_0\ F_\theta^*\ \theta_0,\phi_0\ + F_\phi\ \theta_0,\phi_0\ F_\phi^*\ \theta_0,\phi_0} \right]. \tag{6}$$

In this equation, the asterisk represents complex conjugation. For an adequate representation of levels, the quadratic amplitudes of (5) are normalized with respect to the same value indicated in the denominator of equation (6).

3. DESCRIPTION OF THE TOOL

We proceed now to briefly describe the tool proposed here.

The main form of the interface can be edited by typing, in the MATLAB command line, the sentence >> `guide ConfPolPow3d.fig` [4]. The program is then executed in the opened MATLAB® GUI compiler by, for example, using the Ctrl+T shortcut. The main window of the

program is observed in Figure 1 (the Visualization Space is empty when the interface emerges for the first time). Alternatively, the program can be executed directly by typing >> `ConfPolPow3D`.

The "Browse…" button allows the program to proceed, by calling the `uigetfile` command [4], to load the text file (.dat extension) containing the conformal array configuration. The content of such a configuration file must have the following structure:

```
X TAB Y TAB Z TAB Amp TAB PhaDeg TAB Axn TAB Bxn TAB Gxn TAB Ayn TAB
Byn TAB Gyn TAB KElem\n
```

In this row of numbers, [`X,Y,Z`] are the components of the position vector $\mathbf{r}_n$ of the *n*–th element local coordinate system origin, expressed in λ–units, whereas the variables `Amp` y `PhaDeg` represent the relative amplitude $|I_n|$ and phase Φ_n in degrees, respectively. Next, the tuples [`Axn,Bxn,Gxn`] and [`Ayn,Byn,Gyn`] correspond to the direction angles $[\alpha_{xn}\ \beta_{xn}\ \gamma_{xn}]$ and $[\alpha_{yn}\ \beta_{yn}\ \gamma_{yn}]$, respectively (specified in degrees), from which the unit vectors $\mathbf{a}_{xn}$ and $\mathbf{a}_{yn}$ –see equation (2)– are computed. The user is able to specify these angles without the restriction of considering $\mathbf{a}_{xn}$ and $\mathbf{a}_{yn}$ to be perpendicular: the program will proceed to make appropriate adjustments in order to accomplish such a condition, keeping unchanged both the direction of $\mathbf{a}_{xn}$ and the plane determined by that unit vector and $\mathbf{a}_{yn}$. The remaining unit vector will then be obtained by means of cross product[2]: $\mathbf{a}_{zn} = \mathbf{a}_{xn} \times \mathbf{a}_{yn}$. The `KElem` variable represents an integer that will allow the program to identify the vector field function $\mathbf{f}_n$ that the program will use for that specific element, something that is explained later in more detail. The symbols `TAB` and `\n` indicate tabulation and new–line characters, respectively. In order for the program to correctly establish –by means of the `NElem = size(X,1)` sentence [4,5]– the total number of elements *N* of the array, the line break should be omitted in the last row of the file.

The program will then load the data and run the corresponding previous calculations [3, 5] informing the user, by means of a window message (`mssbox`, [4]) if there have been any errors when accessing the file. If the data has been correctly loaded, then both the number and kind of elements (patches, dipoles, etcetera –see below) will be shown to the user, indicating also that the configuration is ready to be drawn. Before proceeding to plot such configurations, the user will be able to establish the relative size, given in λ–units, of the maximum amplitude and phases, represented by oriented cones [3] and non–oriented cylinders, respectively. All of these parameters

[2] In fact, $\mathbf{a}_{zn}$ will be calculated first, by means of cross product $\mathbf{a}_{xn} \times \mathbf{a}_{yn}$ and a subsequent normalization, and then the program will proceed to modify $\mathbf{a}_{yn}$.

can be configured by the controls gathered on panel “Cones and Cylinders Features”, see Figure 1. Then, by clicking on “Draw!” button, from the “Array Configuration” frame, the program will proceed to plot the array relative amplitude and/or phase distributions (according to the selections performed within the “Draw” panel, see Figure 1). Once plotted, the user will have at his/her disposal the options of erasing the graphics by clicking on “Reset” button, or of obtaining detached figures, capable of being adequately edited, by clicking on any of the “Popup” buttons, located on the bottom vertex of the given plots on the main form.

Once the array configuration has been loaded, it will be necessary to set the power pattern parameters to be further calculated: on the one hand, it will be necessary to indicate if the 3D pattern is requested to be calculated (“Total 3D Power Pattern” at the top of the “Patterns Options” panel), over the whole space (“Whole Space”) or the southern hemisphere (“Hemisphere”), and, on the other hand, the user will have to specify a 2D pattern by setting either $\theta = \text{constant}$ or $\phi = \text{constant}$, at an angle specified by the “Angle (deg):” edit text box, this latter having the option of being selected with the help of a slider located below it.

The user should take into account that, if just a planar pattern is selected, the denominator of equation (6) could be different from that obtained when the 3D pattern is also calculated. The same consideration is applied when the 3D pattern is calculated on the upper hemisphere. In any case, the program will inform the user about these possibilities by means of a warning window.

The “Pattern Fitting” options list will set the kind of limits adjustment of the axis box containing the 3D pattern. The user will be able to select the more adequate solution according to his/her requirements [4,6]: “Automatic”, corresponding to the `axis auto` MATLAB® option; “Packed”, setting `axis fill`, or “Level”, establishing the box limits between $-lev_{min}$ and lev_{min}, being lev_{min} the value indicated on “Min Level in dB (Positive)” text box, from the “Patterns Features” panel. The other alternative, not given in the list, will be similar to “Level” but with the *z* axis of the box between 0 and lev_{min}, and this will be established after selecting the “Hemisphere” choice in the “3D Pattern Options” panel.

The “Theta and Phi Points (3D)” and “Theta or Phi Points (2D)” edit texts will indicate the program the number of points selected for sweeping the θ and ϕ angles on the 3D plot, and for sweeping the ϕ (for $\theta = \text{constant}$) or θ (for $\phi = \text{constant}$) on the planar patterns, respectively.

The “Color Gradient for 3D Patterns” option will set a color map on the 3D power surfaces, with corresponding color bars that will inform the user about the concerned levels (unless the field is null, in which case those bars will be omitted). If such an option is not selected, then the 3D patterns will be drawn with uniform colors and no level bar. In both cases, the axis box will be given

with ticks and labels on the *x*, *y* and *z* axes, in the same way it was made in our previous works [3, 5].

When clicking the "Calculate" button, the program will perform the computations of all the fields (total, θ and ϕ components) specified by the user (3D and/or 2D patterns), by using equation (1). It is important to mention that, in order to accelerate the computing process, the operations were vectorized (computations performed by using MATLAB® matrix and vector operations), a solution that has produced a remarkable improvement on speed, reducing on about 50 times the computing time when compared to the brute–force didactical option (triple `for-end` loop [4], for θ, ϕ and *n* sweepings) given in [3].

Regarding the element factor $\mathbf{f}_n$ vector function, the code is performed so as to give the user the option of having at his/her disposal several options in one function, thus making possible the alternative of setting up a heterogeneous array of incongruent elements. On the source code, such a function, named `PolElemF`, has among its arguments an integer variable that indicates, according to its value, what function will be used. In our code we have considered three options: a half–wavelength dipole aligned along *z* axis (with a groundplane parallel to whether $x = 0$ or $y = 0$ planes, if required), a $cos^2\theta$ patch radiating towards the upper hemisphere and with an $\mathbf{a}_\theta$ polarization, and finally an (unrealistic, but useful for further testing the program) omnidirectional element with $\mathbf{a}_\phi$ polarization. In the code given below, note that the specific element factor, which returns the $\left[f_{nx}\ f_{ny}\ f_{nz}\right]$ triad, is established by the `KElem` variable:

```
function [Felx Fely Felz] = PolElemF(Thetan,Phin,CosAnx,CosBnx,CosGnx,
CosAny,CosBny,CosGny,CosAnz,CosBnz,CosGnz,KElem)
% Arguments:
% Thetan, Phin: angles measured on n-th element local coordinate system
% CosAnx, CosBnx, CosGnx: direction cosines of axn unit vector
% CosAny, ... ,CosGnz: direction cosines of ayn and azn
% KElem: an integer that indicates which element factor must be chosen
% KElem = 1 means short dipole aligned along z axis
if KElem == 1,
         Den = sin(Thetan);  % To avoid division by zero
```

```
    if(Den == 0) Felx = 0; Fely = 0; Felz = 0;
    else
        BetaHalfLength = pi*0.5; % A half wavelength dipole
      % Return the element factor projected on z axis
        Fel = (-(cos(BetaHalfLength.*cos(Thetan))-
        cos(BetaHalfLength))./Den)*[CosAnz CosBnz CosGnz];
        Felx = Fel(1,1); Fely = Fel(1,2); Felz = Fel(1,3);
    end;
    % If you want a ground plane parallel to y = 0, uncomment lines below:
%     SPhn = sin(Phin);
%     if (SPhn < 0), Felx = 0; Fely = 0; Felz = 0; % Just radiate on
      0<=Phi<=pi zone
%     else
%        H = 0.25;          % Distance from dipole to groundplane
%        Coeff = sin(2*pi*H*sin(Thetan)*SPhn);  % Coeff due to the presence
         of the groundplane
%        Felx = Felx*Coeff; Fely = Fely*Coeff; Felz = Felz*Coeff;
%     end;

     % If you want a ground plane parallel to x = 0, uncomment lines below:
%     CPhn = cos(Phin);
%     if (CPhn < 0), Felx = 0; Fely = 0; Felz = 0;  % Just radiate
      on -pi/2<=Phi<=pi/2 zone
%     else
%       H = 0.25;           % Distance from dipole to groundplane
%       Coeff = sin(2*pi*H*Den*CPhn);     % Coeff due to the presence
```

```
            of the groundplane
     %       Felx = Felx*Coeff; Fely = Fely*Coeff; Felz = Felz*Coeff;
     %     end;
% KElem = 2 means cos^2(theta) pattern polarized along theta direction
elseif KElem == 2,
     CThn = cos(Thetan);
     if(CThn >= 0) % Be sure it will radiate over upper hemisphere
     % Notice that the field is cos^2(theta) times a_sub_theta times the
      direction cosines matrix:
     fR = CThn^2*[CThn.*cos(Phin) CThn.*sin(Phin) -sin(Thetan)]*[CosAnx CosBnx
     CosGnx; CosAny CosBny CosGny; CosAnz CosBnz CosGnz];
     % You can polarize it along Phin axis if you want (must uncomment
      the following line)
     % fR = CThn^2*[-sin(Phin) cos(Phin) 0]*[CosAnx CosBnx CosGnx; CosAny CosBny
      CosGny; CosAnz CosBnz CosGnz];
     % Return the projections over the global axes
     Felx = fR(1,1); Fely = fR(1,2); Felz = fR(1,3);
     else    % The field is zero on the lower hemisphere
     Felx = 0; Fely = 0; Felz = 0;
     end;
% Any other KElem value corresponds to an omnidirectional field polarized,
for example, along Phin:
else
      fR = [-sin(Phin) cos(Phin) 0]*[CosAnx CosBnx CosGnx; CosAny CosBny CosGny;
      CosAnz CosBnz CosGnz];
     Felx = fR(1,1); Fely = fR(1,2); Felz = fR(1,3);
end;
```

The user can even speed up further the computing process by vectorizing this `PolElemF` function also, at the expense of the comprehensibility of the code.

Once the calculations have been performed, the user will have at his/her disposal the option of properly drawing the 3D [3] and 2D patterns by clicking on the corresponding "Draw!" button. After this, by clicking on the "Popup" button of the 3D diagram, the program will plot, in separate windows, the graphics corresponding to the total, $\theta-$ and $\phi-$ components of the field. The procedures of building up these polar surfaces are based on the one given in [3] and [5], by previously setting up the matrices used by the `surf` function [4, 6]. In order to help the user to determine the global orientation of the 3D fields, a set of three small cones oriented along the global *x*, *y* and *z* axes has been added to each of these polar figures, by taking advantage of the same function that draws the amplitude representative oriented cones [5]. The location and size of such a three–cone system can be, of course, modified on the source code according to the requirements of the user.

Analogously, the θ and ϕ components, together with the total field, will be drawn already on three separate plane figures after clicking on the "Pop up" button located at the right–hand–side bottom of the "Total Power Curve (at Constant Angle)" panel. In all these cases, including the plot belonging to such a panel, we have used a function that performs planar polar plots. The code of such a function is based on the editable `polar` MATLAB function [4], by making the function, named `PlanarPlot`, to be able to firstly, plot the curve with the θ angle having a clockwise sweeping, and with the zero located at the top of the vertical axis; secondly, to plot the curve with the ϕ angle on a counterclockwise approach; and finally, to specify the field levels from 0 to $-lev_{min}$ decibels.

4. SIMULATIONS AND RESULTS

Following the previous description, two configurations are presented in order to show the reader the utility of the tool.

4.1. A λ/2 DIPOLE LYING ON THE Y–Z PLANE AND ORIENTED ALONG θ = 45º

Let us consider first a 0.5λ dipole centered on the global coordinate system and oriented along an axis that lies on the *y*–*z* plane, with an angle of 45º with respect to the *z* axis over the θ direction. In this case –considering that in the dipole is aligned along its *z* local coordinate system– the .dat configuration file is:

```
0     0     0     1     0     0     90    90    90    45    135   1
```

Namely: $\mathbf{r}_n = [0\ 0\ 0]$, $I_n = [1 0] = e^{j0}$, $[\alpha_{xn}\ \beta_{xn}\ \gamma_{xn}] = [0\ 90\ 90]$, $[\alpha_{yn}\ \beta_{yn}\ \gamma_{yn}] = [90\ 45\ 135]$, `KElem` = 1 (which corresponds to a dipole, see `PolElemF` code above).

The corresponding 3D power patterns obtained with the tool are given in Figures 2 and 3.

Figures 4 and 5 show the $\phi = 35^o$ patterns obtained for each of the fields (total and two components) radiated by the dipole. If the dipole were aligned along the global *z* axis, it would produce a $|F_\phi| = 0$ result, as actually happens if the configuration is tried with the program. This latter result is in accordance to the simple observation that, in this case, $\mathbf{F} \cdot \mathbf{a}_\phi = (F\mathbf{a}_z) \cdot \mathbf{a}_\phi = 0$ over the whole space.

4.2. ARRAY COMPOSED OF 40 $\cos^2\theta$ PATCHES CONFORMED OVER A CONICAL SURFACE.

With this tool it was decided to simulate again the array configuration presented in [3] composed of forty $cos^2\theta$–type circular patches conformed over a cone of base radius and height of equal size ($r_C = h_C = 4\lambda$), and whose revolution axis is located along the *z* axis of the global coordinate system. The antenna elements are then arranged as follows: 16 patches with $I_n = e^{j3\pi/4}$ and distributed over the base circumference $z = 0$, 12 patches with excitations $I_n = 0.75e^{j\pi/2}$ and distributed over a smaller concentric circumference located at the plane $z = \lambda$, 8 patches $I_n = 0.5e^{j\pi/4}$ arranged over another concentric circumference at $z = 2\lambda$, and finally 4 elements with $I_n = 0.25e^{j0}$ and $z = 3\lambda$. Over each circumference the elements are uniformly distributed and tangent to the cone. Also, four of the patches corresponding to different circumferences are parallel to a cone generating axis. This time, nevertheless, the elements are considered to radiate with θ polarization, the element factor therefore being $\mathbf{f}_n = cos^2\theta_n \mathbf{a}_{\theta_n}$, where $\mathbf{a}_{\theta_n}$ corresponds to the unit vector pointing towards the increasing local θ_n coordinate.

Figures 6 to 9 show the power patterns generated by such a conformal arrangement, with the plane plots obtained for $\theta = 35^o$. It can be seen, as expected, that the 3D plot, shown in Figure 6, differs from the analogous pattern given in [3], as this time the field is configured by means of a vector summation.

5. FINAL COMMENTS

For the visual representation of the fields generated by conformal arrays composed of elements whose patterns are well studied, and with configurations where the mutual coupling can be neglected, the presented tool constitutes a reliable alternative to the commercial programs

based on known numerical techniques. The user has also the option of modifying at will the source code, which that remarkably increases the utility of the tool.

In spite of the completeness of the tool, some improvements could be suggested for it. The user could add the option of visualizing other field components. The possibility of writing an output file containing any of the calculated patterns is another useful feature. The three components of the field for the constant θ (or ϕ) polar cuts could be represented on a single figure. Some users would prefer Cartesian planar plots instead of polar ones. Additional pattern attributes could be controlled by the inclusion of further controls (buttons, check boxes, etcetera), but at the price of needing more space to locate them.

Finally, it is important to remark that several results obtained with the proposed tool were successfully validated –taking into account that the negligibility of the mutual coupling between elements would cause apparent differences with respect to the more realistic results– with a commercial software tool (based on the finite difference time domain [7]).

The source code is freely available at

http://gtec.des.udc.es/web/images/files/confpolpow3d.zip . This .zip file also includes several antenna configurations, such as the ones given here as examples (_1_rotated_dipole.dat for the dipole and _40_patches_on_cone.dat for the conical arrangement), together with a brief tutorial that would help the reader to quickly grasp the program handling.

6. ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Ministry of Science and Innovation and FEDER funds from the European Union (under projects TEC2010–19545–C04–01 and CSD2008–00010, respectively).

7. REFERENCES

[1]. L. Josefsson and P. Persson, *Conformal Array Antenna Theory and Design*, New York, IEEE Press/John Wiley, 2006.

[2]. The MATLAB Group, Inc., www.mathworks.com/products/phased-array.

[3]. J. C. Brégains, J. A. García–Naya, A. Dapena, M. González–López, "A MATLAB Tool for Visualizing The 3D Polar Power Patterns and Excitations of Conformal Arrays", IEEE Antennas and Propagation Magazine, **52**, 4, 2010, pp. 127–133.

[4]. The MATLAB Group, Inc., "MATLAB Function Reference: Volumes 1–2–3", (printed version of the corresponding MATLAB Help menu), 2008.

[5]. J. C. Brégains, F. Ares, E. Moreno, "Visualizing the 3D Polar Power Patterns and Excitations of Planar Arrays with MATLAB ", IEEE Antennas and Propagation Magazine, **46**, 2, 2004, pp. 108–112.

[6]. The MATLAB Group, Inc., "MATLAB: 3–D Visualization", (printed version of the corresponding MATLAB Help menu), 2008.

[7]. Schmid & Partner Engineering AG, *SEMCAD X. Designer's handbook*, SPEAG, 2011. More information at www.semcad.com.

FIGURES

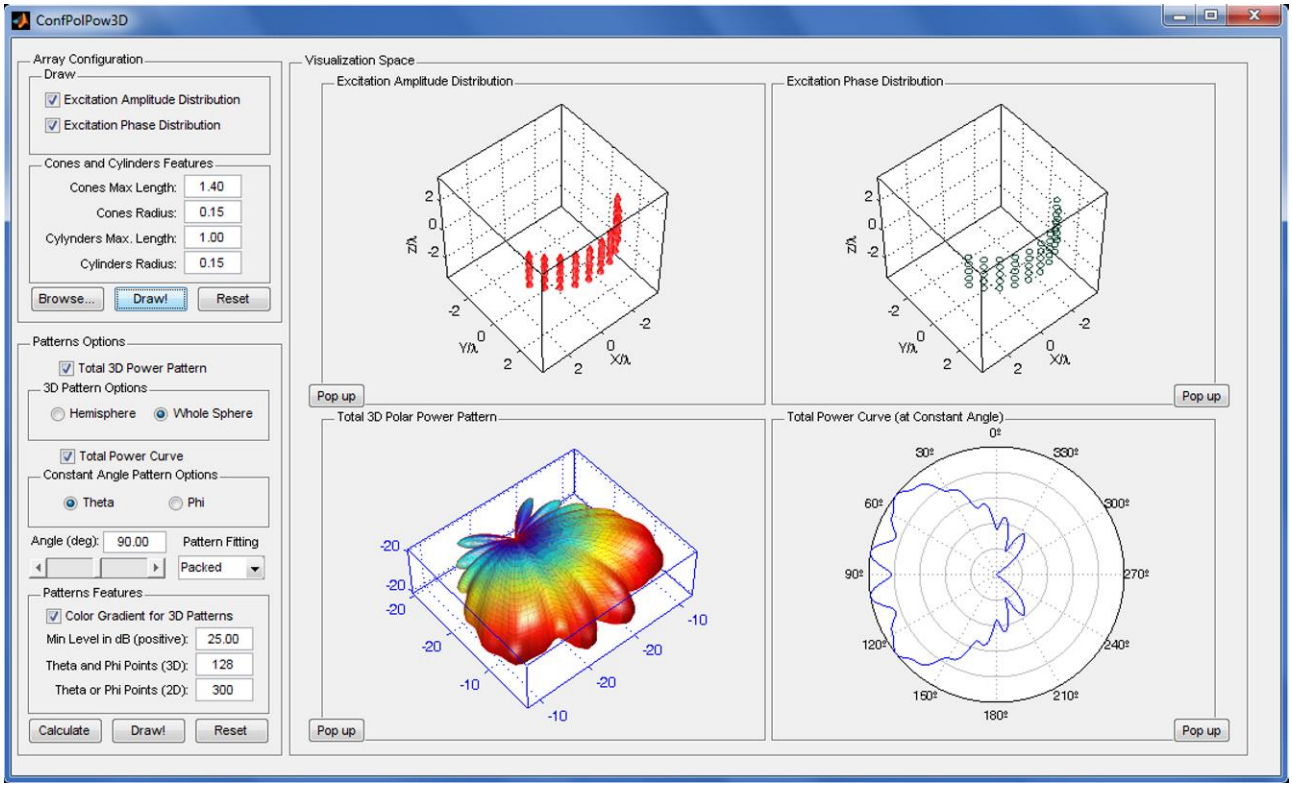

Figure 1. Outward appearance of the main interface of the tool.

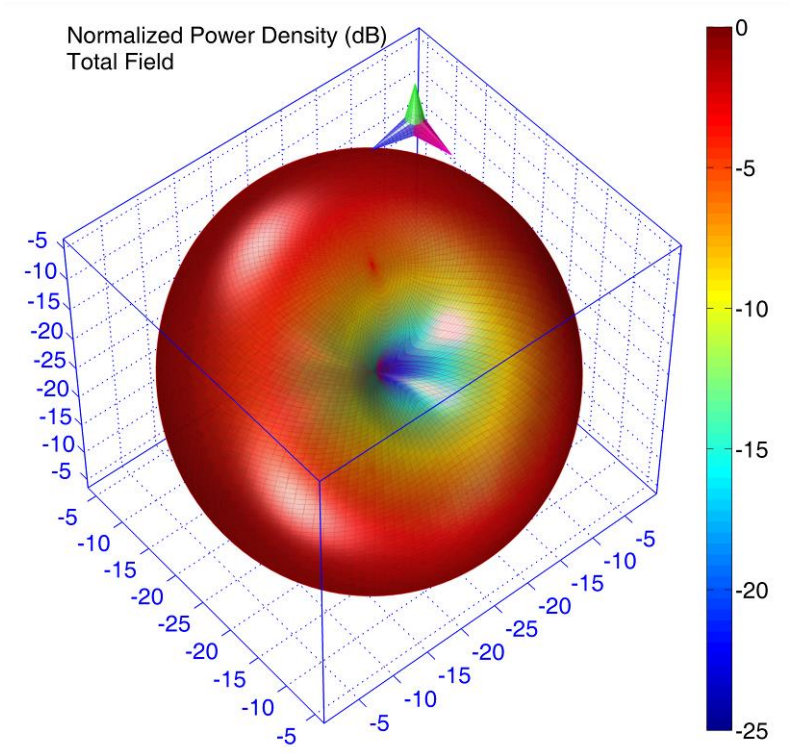

Figure 2. Total field pattern radiated by a $\lambda/2$ dipole located on the x = 0 plane, rotated an angle $\theta = 45^{o}$.

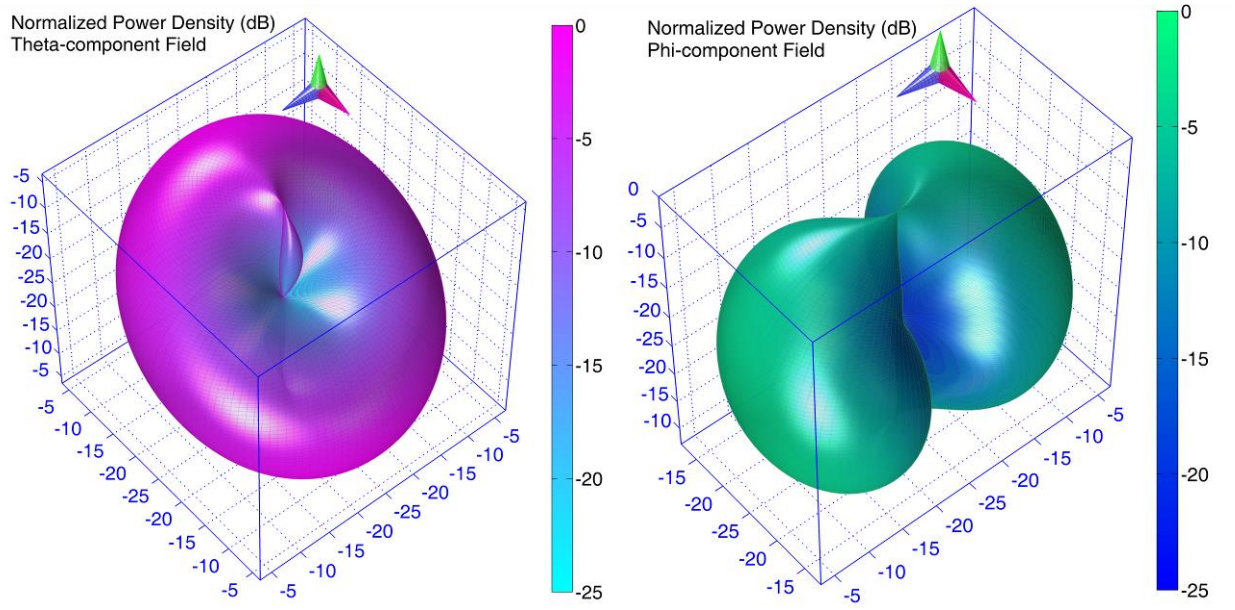

Figure 3. θ and ϕ components of the field given in Figure 2.

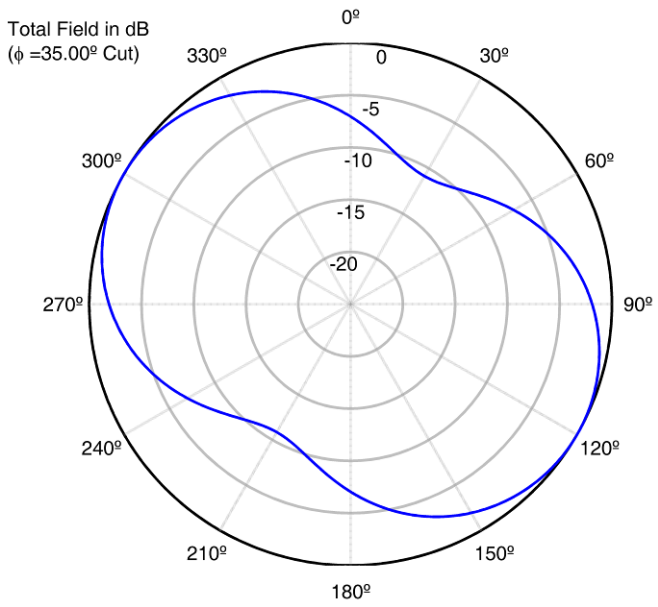

Figure 4. $\phi = 35^o$ cut of the total field given in Figure 2.

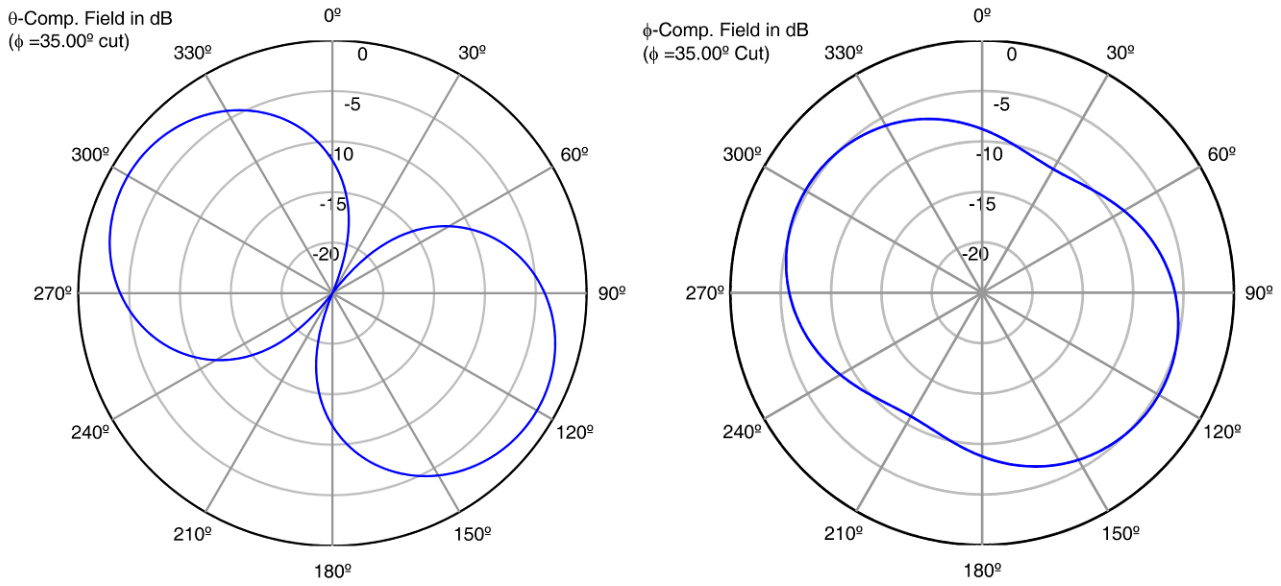

Figure 5. θ and ϕ components of the $\phi = 35^o$ cut of the field patterns given in Figure 3.

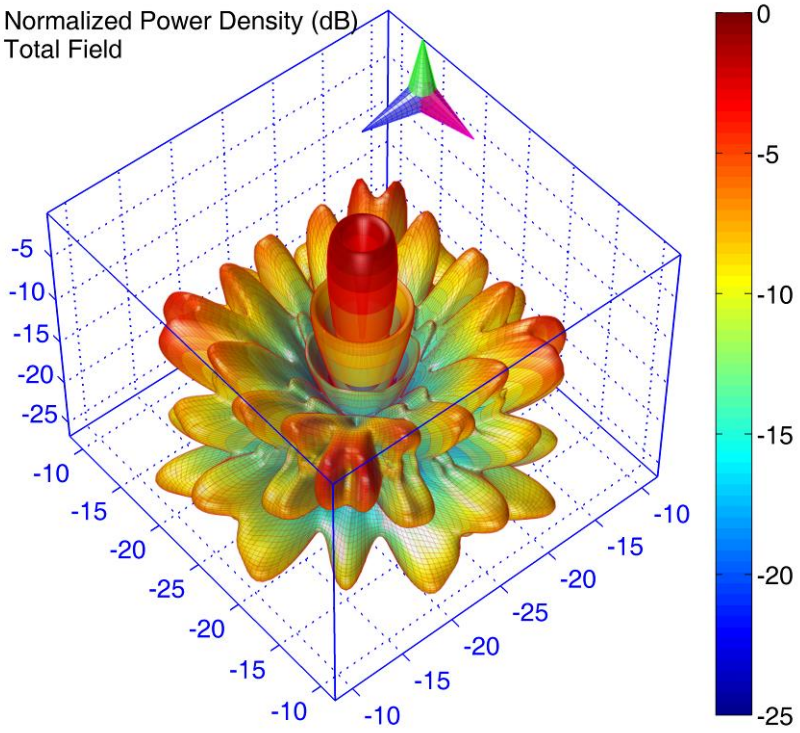

Figure 6. Total 3D field radiated by a conical array composed of 40 patches [3] with $\mathbf{f}_n = cos^2\,\theta_n \mathbf{a}_{\theta_n}$.

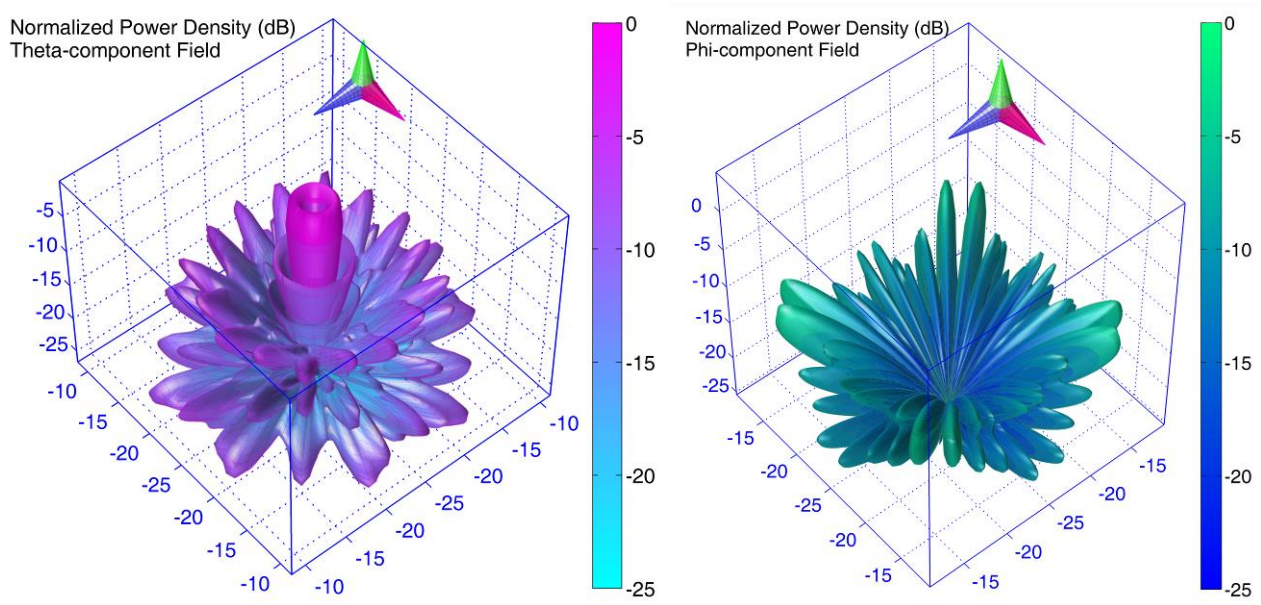

Figure 7. θ and ϕ components of the field given in Figure 6.

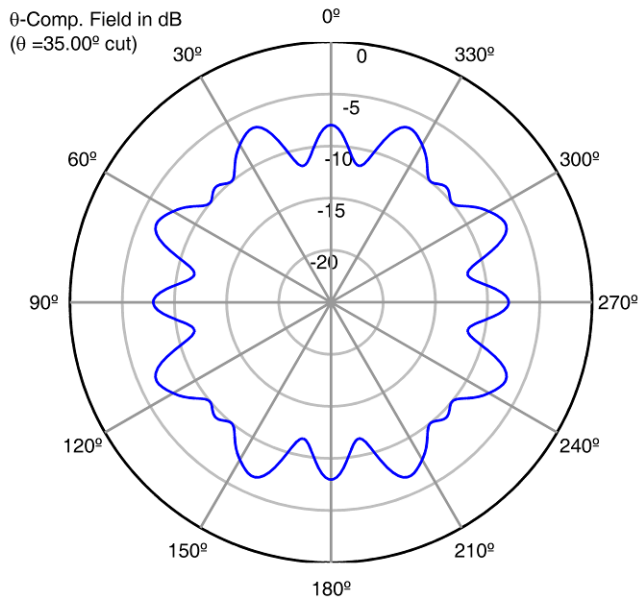

Figure 8. $\theta = 35^o$ cut of the field shown in Figure 7.

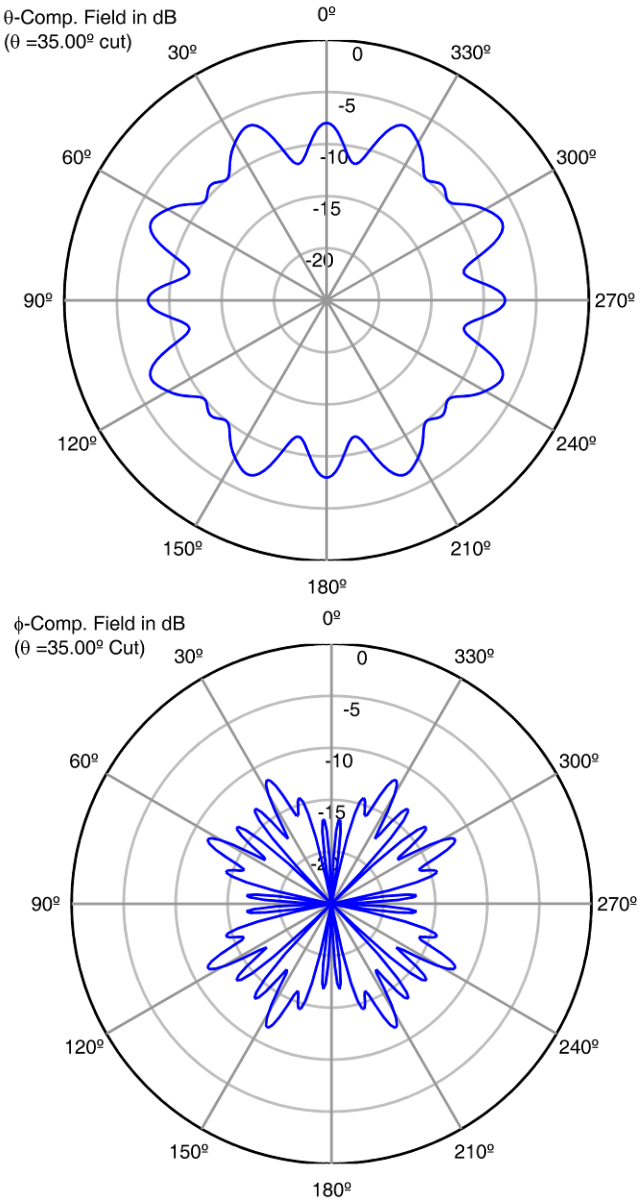

Figure 9. θ and ϕ components of the $\theta = 35^o$ cuts of the fields observed in Figure 8.