Prepared By :	No.		
Neill Tucker	01:001		
Project Title : ArrayCalc v2.5	Date 15/02/2015	Rev A	File What's New in ArrayCalc v2_5

An Overview Of What's New In

ArrayCalc v2.5

At a glance :

- Global variables for array input power, efficiency and propagation-medium impedance have been added; allowing plots of absolute values such as gain, power flux density and field strength.
- Estimation of patch-efficiency function to assist in estimating overall array efficiency.
- Plotting of near-fields in the form of 2D planes or lines through 3D space around the array.
- Visualisation of wave propagation in the form of 3D wave animations.
- Normalisation of all element models to unity and use of global wave propagation velocity, to allow application in other homogeneous mediums e.g. Ultrasound
- Phase convention for the element excitations changed ! (see section 6)

CONTENTS

CONTENTS 2

1. INTRODUCTION 3

2. PLOTTING GAIN 4

3. PLOTTING FIELD PARAMETERS 7

4. WAVE SLICE AND WAVE ANIMATION 12

5. ULTRASOUND APPLICATIONS 13

6. PHASE AND POWER CONVENTIONS 17

REFERENCES 18

1. INTRODUCTION

ArrayCalc versions up to and including v2.4.2 provided basic analysis of arbitrary arrays of standard array elements such dipoles, slots and patches. Plots were limited to the far-field and of directive gain in dBi. While these are useful, by adding a simple estimate of array efficiency, plots can also be made of absolute gain in dB (Gain=Directivity-Losses). Including the input power to the array and the impedance of the wave propagation medium, absolute values for power flux density and field strengths can also be calculated.

For many applications, the near-field of antennas or transducers is just something to be aware of, and either avoided altogether or measured and then transformed into the far-field. However there are increasing numbers of applications such as medical imaging, collision avoidance and novel modulation schemes that operate in the near-field. Since ArrayCalc does not compute surface current densities on the actual array elements, it cannot calculate the true near-field of individual elements. However it can calculate near-field of the array as a whole i.e. Interactions between the localised array element models. Figure 1-1 shows the field between 2 vertical dipoles spaced at 2*lambda and fed with a combined power of 100W at 1Ghz.

Although static plots of near-fields are the most useful in terms of quantitative analysis; animations of wave propagation are also useful in that they allow you to ‘see’ the wave-front evolve as it transitions from near to far-field. To this end, a function has been added that allows visualisation of the wave propagation in any arbitrary 2D plane within the 3D space around the array.

To take full advantage of the additional functionality it is recommended that the user takes some time to look through the subsequent sections of this document, which describe in a little more detail, the new functions, their uses and their limitations.

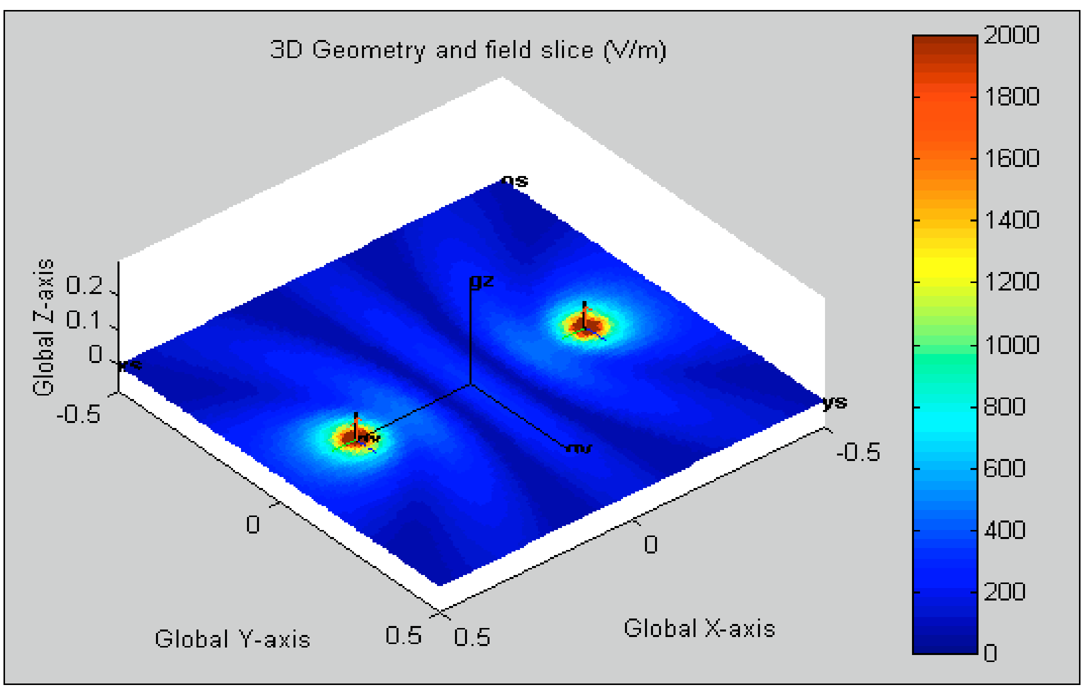


Figure 1-1 RMS E-field between 2 dipoles fed with a combined power of 100W at 1GHz

2. PLOTTING GAIN

In order to plot the gain of a single element or array, its directivity must first be calculated, as in previous versions of ArrayCalc. Once calculated using the calc_directivity.m function, plots are made with the normalisation option set to 'no' or 'none' depending on the type of plot. Plots are then automatically in dBi (dB relative to an isotrope).

In ArrayCalc v2.5 another global variable called arrayeff_config has been introduced which is the array efficiency in percent. If this has the value of 100 then plots remain in dBi, while for values less than 100, a value of gain is calculated according to :

$$\text{Gain(dB)} = \text{Directivity(dBi)} + 10^*\log10(\text{Efficiency\%} / 100).$$

Within ArrayCalc there is no separate variable for gain, the calc_directivity.m function just returns a modified value of directivity if the efficiency is less than 100. The plot functions also access the efficiency value and if it is less than 100 label the plots as gain (dB). A value of peak directivity is printed out by the calc_directivity.m function for reference.

Although a somewhat trivial calculation, it does make comparisons with measured or full wave simulation outputs easier. The value for arrayeff_config can be an arbitrary figure selected by the user based on experience, or it can be based on estimates of the efficiency of individual array components. Two functions calc_patchr_eff.m and calc_patchc_eff.m are provided to give estimates of rectangular and circular patch efficiencies respectively. Since all power supplied to an array's elements will be subject to losses within them, the efficiency for a single element is representative of efficiency of the array as a whole (not counting feed network losses), regardless of power distribution. Ref figures 2-2 and 2-3

The table in figure 2-1 shows some parameter comparisons between typical patch elements modelled in a full-wave solver (Ansoft Designer v2.2) and ArrayCalc. The nominal patch dimensions were obtained using ArrayCalc's design_patchr.m and design_patchc.m functions; efficiency and bandwidths were then estimated using ArrayCalc's calc_patchr_eff.m and calc_patchc_eff.m functions. Each patch was then laid out in Ansoft Designer and fine-tuned (length, width and feed point location) to obtain the desired resonant frequency. On the Smith Chart the impedance locus crossed the centre of the 50 ohm chart at the resonant frequency. The figures in brackets are those obtained from the Ansoft Designer analysis. See m-files ef1, ef2, ef3...ef7 in the validation directory for plots.

Ansoft Designer/ArrayCalc Patch Comparison Examples :
Ansoft values shown in (brackets)

File	Type	Substrate	Er	tand	sigma(Cu)	h(mm)	W(mm)	L(mm)	a(mm)	Fo(MHZ)	Gain(dB)	BW VSWR=2 (MHz)
ef1	Rect	RO4350	3.48	0.004	5.8e7	0.76	41.8 (41.7)	33.3 (33.2)	---	2400	4.33 (4.59)	21 (23)
ef2	Rect	FR4	4.4	0.02	5.8e7	0.76	38.0 (38.0)	29.7 (29.6)	---	2400	0.01 (0.48)	46 (46)
ef3	Rect	FR4	4.4	0.02	5.8e7	3.00	38.0 (38.0)	28.8 (28.6)	---	2400	3.13 (3.81)	90 (87)
ef4	Rect	Air	1.0	0.00	5.8e7	6.00	62.5 (62.5)	54.3 (54.3)	---	2400	10.00 (9.47)	172 (184)
ef5	Rect	TMM 10i	9.8	0.002	5.8e7	0.76	6.5 (6.5)	4.6 (4.5)	---	10000	4.18 (4.16)	172 (190)
ef6	Circ	RO4350	3.48	0.004	5.8e7	0.76	---	---	19.3 (20.5)	2400	4.44 (4.31)	19 (21)
ef7	Circ	RO4350	3.48	0.004	5.8e7	1.60	---	---	18.7 (20.0)	2400	5.26 (5.25)	35 (37)

Figure 2-1 Ansoft Designer / ArrayCalc comparisons

For the most part the ArrayCalc results compare well with those using Ansoft Designer. The main exception is for the thick, air-spaced patch (ref ef4), where the asymmetry of the feed point significantly affected the current distribution on the patch and skewed the pattern in the E-field (Phi=0) direction. A balanced feed using anti-phase probes or a slot-coupled patch will reduce the pattern asymmetry and give results closer to those of the simple model. Note also that thick patches require a small series capacitor at the feed point to tune out the feed probe series impedance, in the ef4 model this was 2pF.

Obviously this is just a comparison between one set of calculated results and another. It should be noted that the ArrayCalc and Ansoft Designer models assume an infinite ground-plane and dielectric structure, implying that surface waves launched into the dielectric would eventually be lost due to tan-delta losses. In most practical cases the ground-plane and dielectric is finite and the surface waves would then radiate to some extent at the dielectric-air interface. Depending on the exact dimensions this may increase or reduce gain and bandwidth and add some ripple to the patterns.

Leaving aside the finite ground-plane issues, full-wave analysis such as that carried out in Ansoft Designer is generally accepted to be about as accurate as Electromagnetic (EM) modelling gets, so makes a good bench-mark for more approximate methods, such as those used by ArrayCalc. Figure 2-2 shows a pattern comparison for a 4-element array.

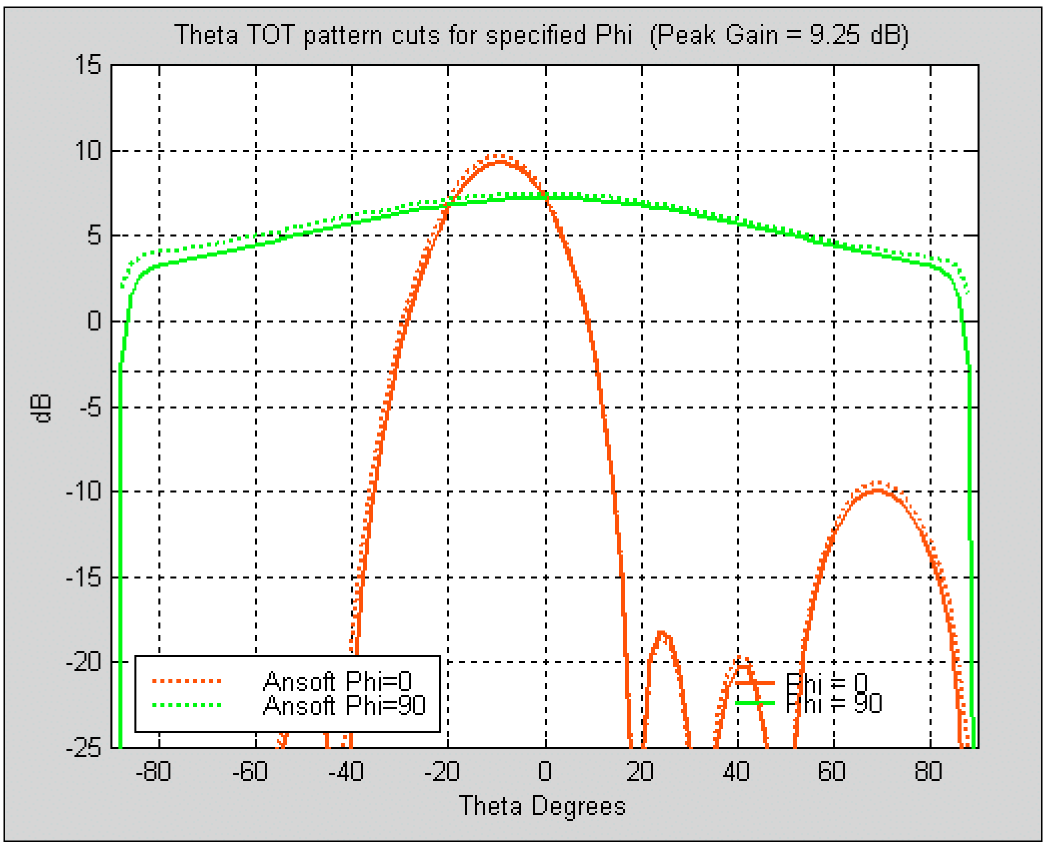


Figure 2-2 Array comprising 4 x (ef1 rectangular patch) File: ef1_4element.m
Element spacing : 0.7*lambda in H-plane
Steered to : Theta=-20deg Phi=0deg
Window : Taylor (-25db SL)

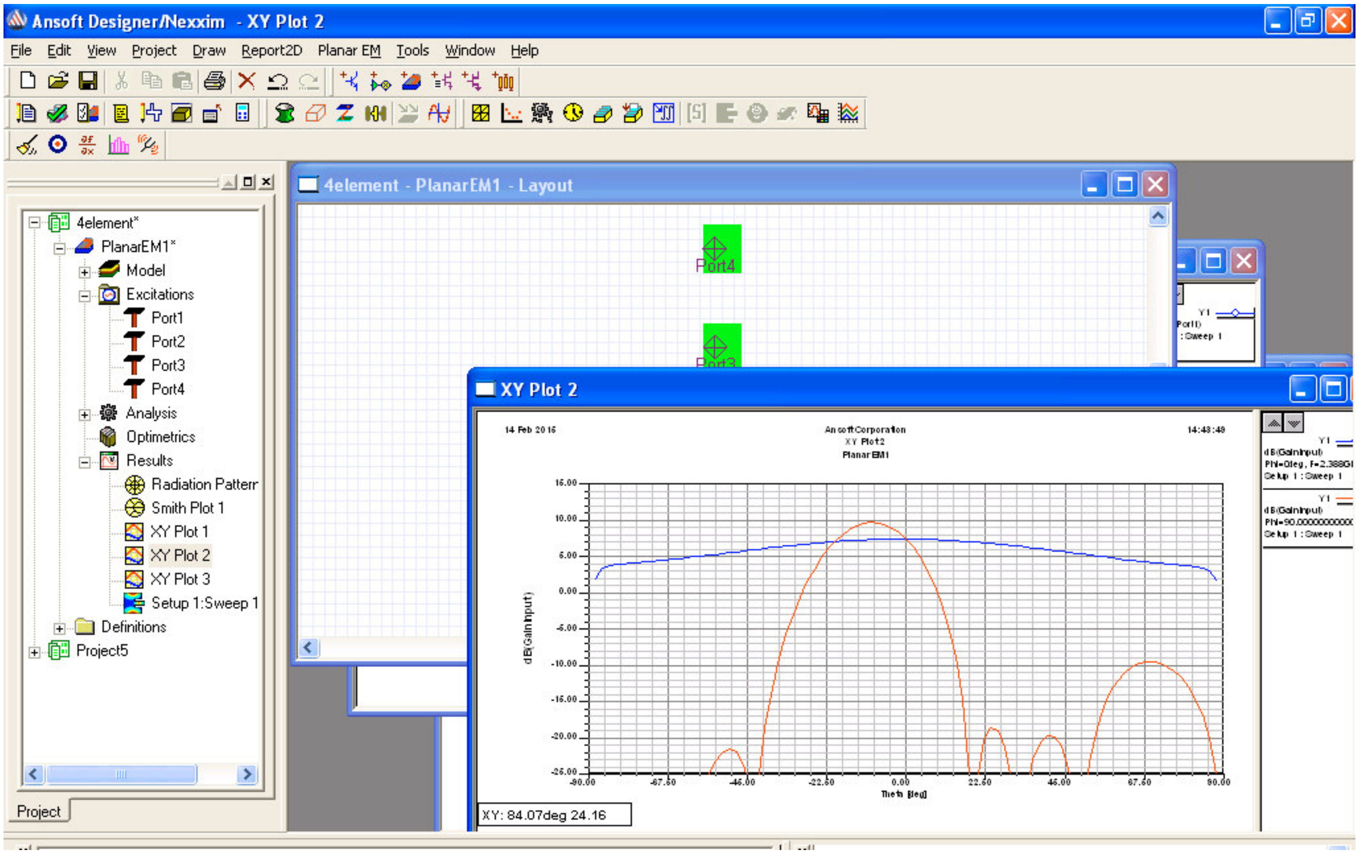


Figure 2-3 Screen shot of Ansoft Designer model of 4-element array

Regarding the equations and code used in ArrayCalc to estimate patch efficiency and bandwidth, they are somewhat of a mash-up from various sources [1], [2], [3]. Despite numerous attempts I was unable to find a set of equations from a single reference that gave acceptable results. Since the objective was to achieve the closest match to the full-wave solution for a wide range of patch configurations, I settled for performance over mathematical purity.

3. PLOTTING FIELD PARAMETERS

In the previous section we saw that an efficiency value for the array allowed us to convert a directivity value into a value for gain as a function of direction G(theta,phi). If we know how much power is input into the array and the distance from the array we can calculate the power flux density (W/m^2) as a function of direction and distance :

Pden(R,theta,phi) = (ArrayPower * G(theta,phi)) / (4*pi*R^2) Equation 3-1

Furthermore, if the impedance of the medium in which the power flux is present is known, then the RMS E-field strength (V/m) can be calculated :

RMS Efield = sqrt(Pden * Z) Equation 3-2

Where Z = 377Ohm for free space.

I n ArrayCalc v2.5 there are two extra global variables arraypwr_config and impedance_config to represent the power into the array (Watts) and the impedance of the medium (Ohms) respectively, see the init.m file.

To establish the accuracy and limitations of ArrayCalc's near-field calculations a comparison was made with 4NEC2 [6]. The plots in figures 3-1 and 3-2 show near field surface plots for 2 vertically orientated (z-axis) half-wave dipoles, spaced 2*lambda apart. The operating frequency was 1Ghz, so lambda=0.3m. Plots are in the X-Y plane at Z=0.

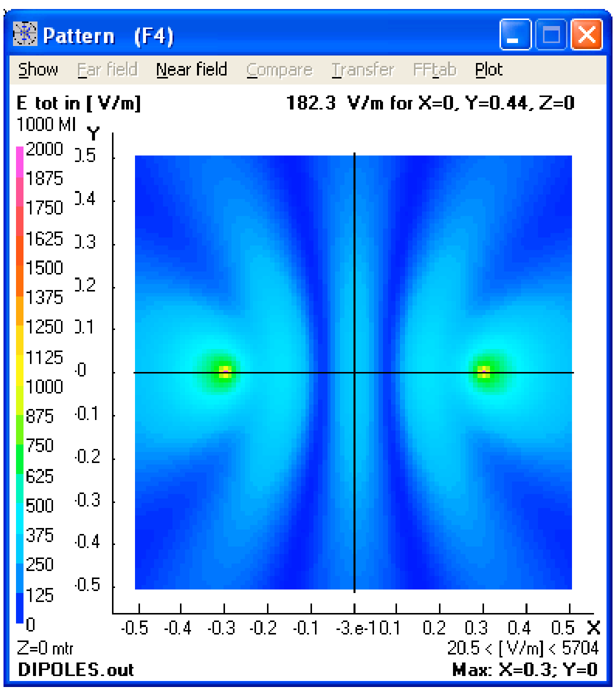


Figure 3-1 4NEC2 Nearfield plot of dipoles

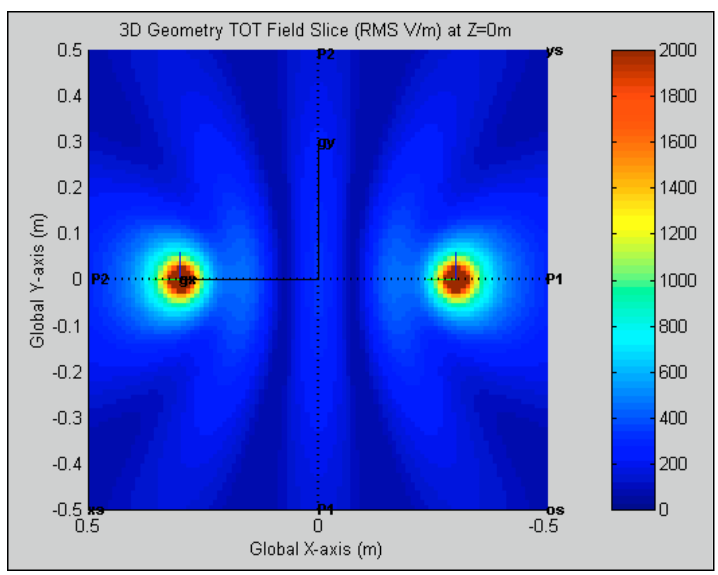


Figure 3-2 ArrayCalc Nearfield plot of dipoles

Although the surface plots look similar, a more accurate comparison can be made by plotting the RMS E-fields along the primary X & Y axes, as shown in figure 3-3. It can be seen from the graphs that ArrayCalc's near-field calculation is quite accurate at a distance of 1*lambda (0.3m) from the dipoles. Which is well within the far-field distance for the array, given by :

Array FarField = (2^*D^2)/lambda   Where D=2*lambda for this array, so

Array FarField = 8*lambda

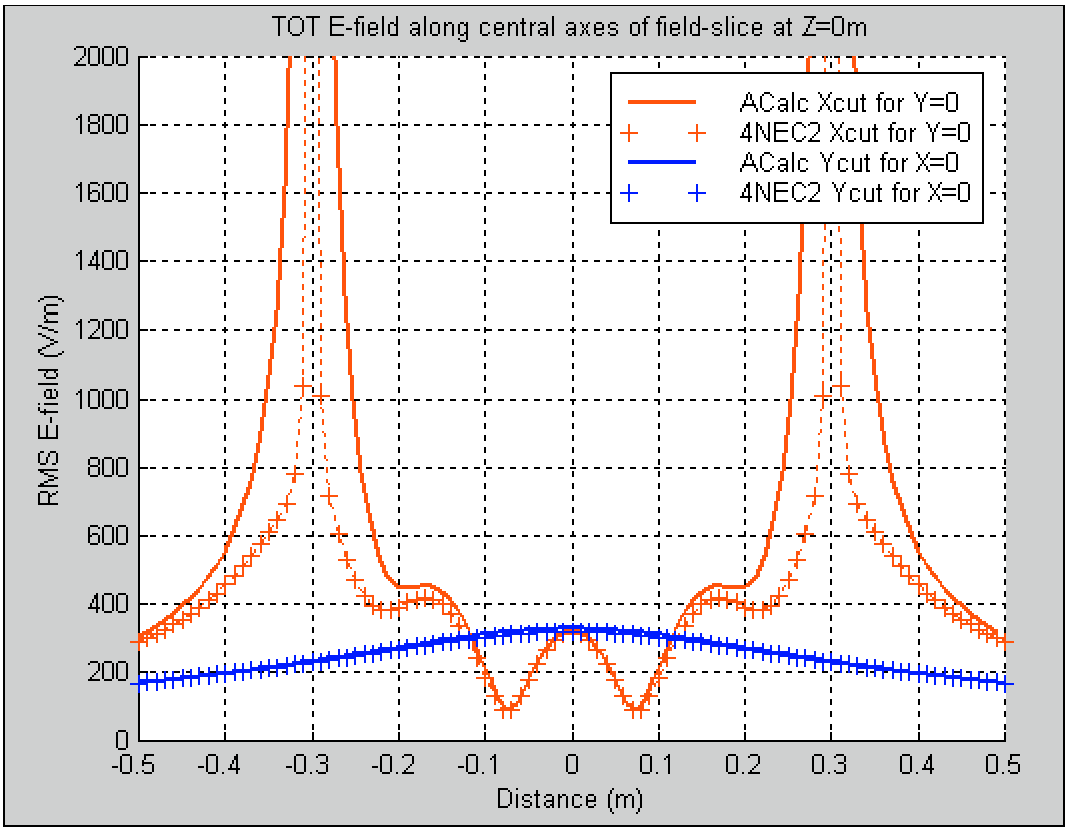


Figure 3-3 RMS E-Field plot comparisons along primary X & Y axes

For further verification, near-field plots were also made in the X-Z plane, board-side to the array at Y-values of 1*lambda (0.3m) and 2*lamda (0.6m). The 3D geometry of the arrangement is shown in figure 3-4. Comparisons of the primary axis cuts at Y=0.3m and Y=0.6m are shown in figures 3-5 and 3-6 respectively.

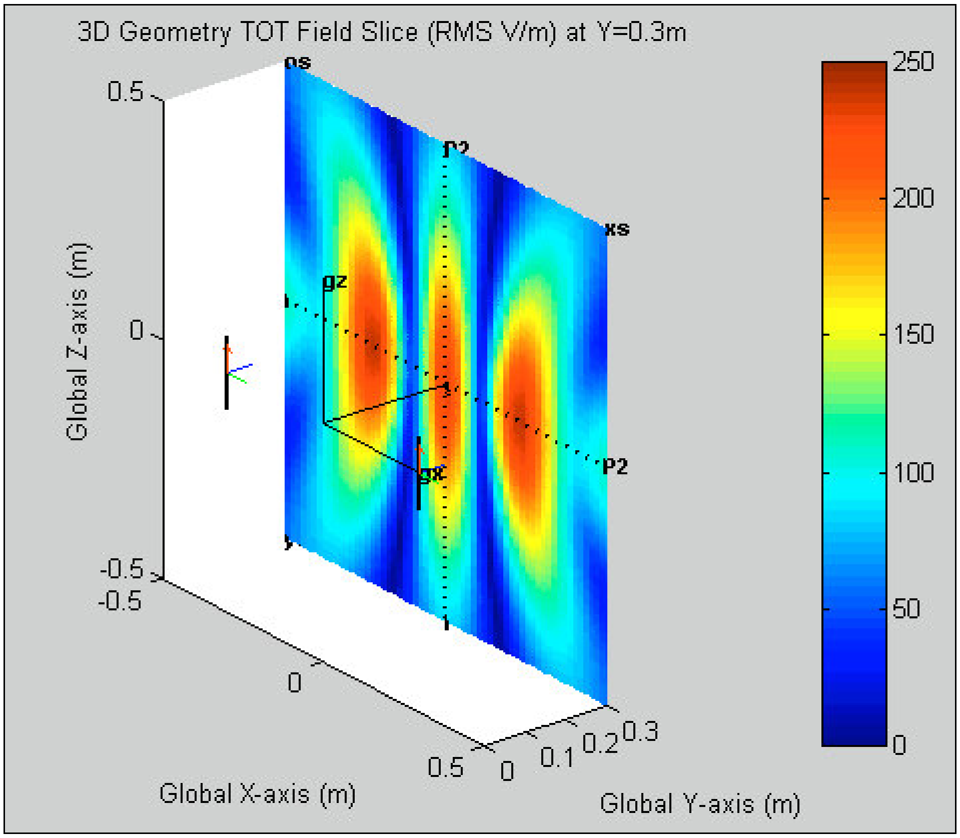
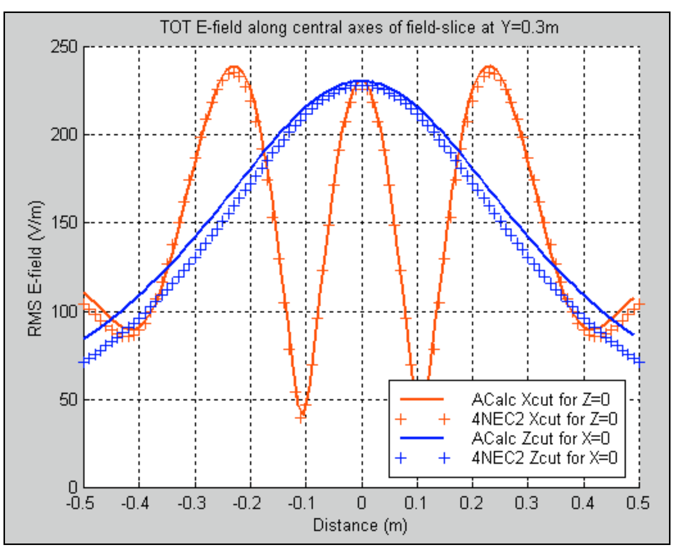


Figure 3-4 3D Geometry of the X-Z plane field slice at a Y=0.3m (1*lambda)



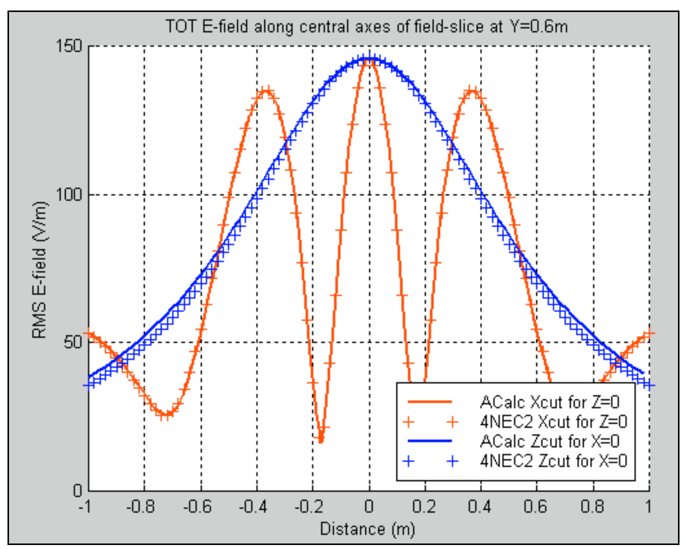


Figure 3-5 X & Z-axis cuts for Y=0.3m (1*lambda) Figure 3-6 X & Z-axis cuts for Y=0.6m (2*lambda)

From the graphs it can be seen that there is reasonable agreement with the 4NEC2 results at 0.3m (1*lambda) and very good agreement at 0.6m (2*lambda) from the array. Overall, the minimum recommended distance for near-field calculations is 2*lambda although distances down to 1*lambda could be used for visualisation rather than precise numeric values. The main assumption is that the elements are half-wave resonant or smaller i.e. most dipoles and patches. The ArrayCalc files, valdipoles1/2/3.m and the 4NEC2 model valdipoles.nec can be found in the validation directory.

Practical application in an array

For the calculations to work for an array, the array excitations must be normalised. That is, the linear amplitudes of all the element excitations must sum to unity. The function norm_array.m is provided for this and should be used before any flux density calculations are requested.

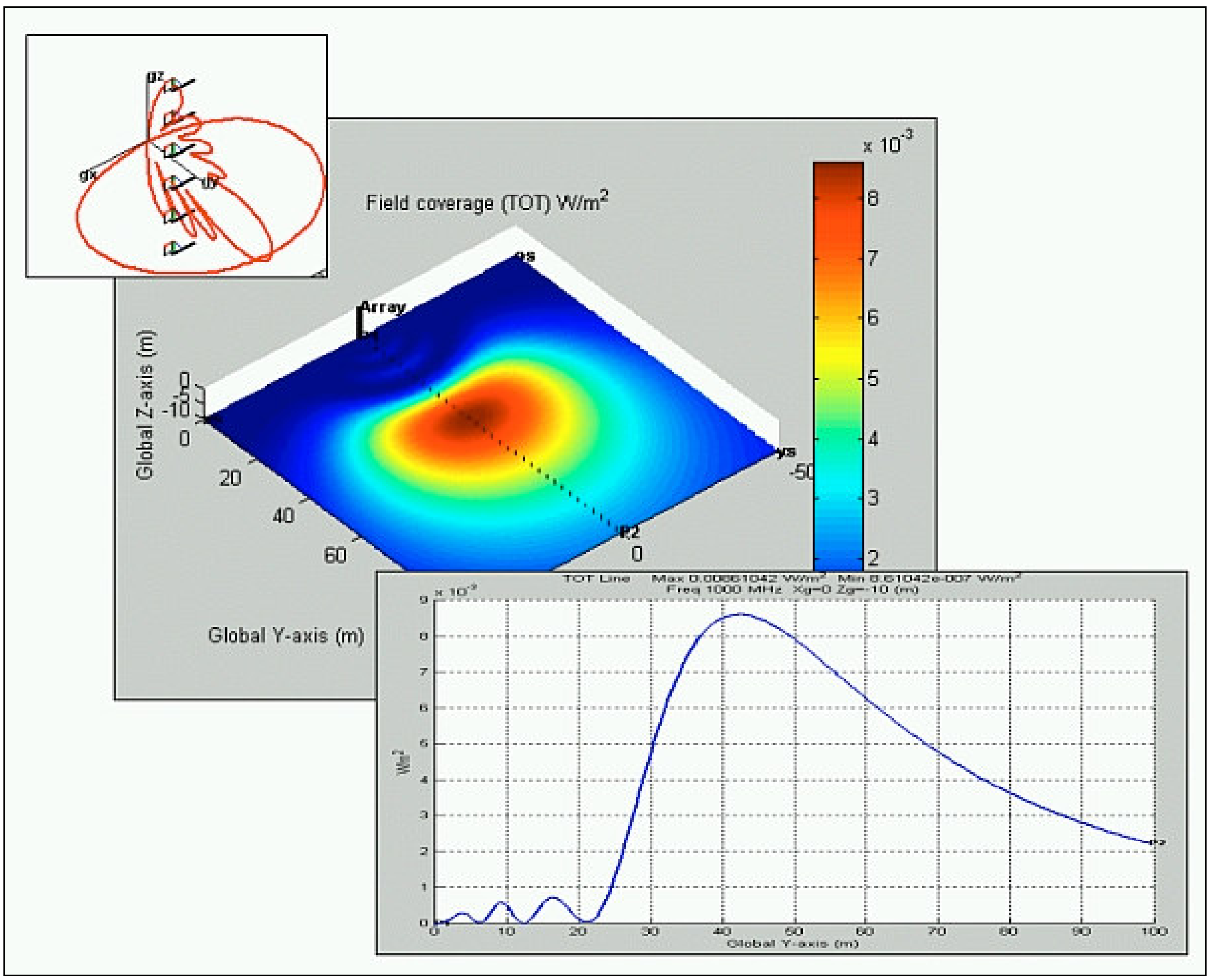


Figure 3-4 Output from excoverage.m example file

In the far-field, gain and directivity are plotted as a function of theta and phi over a spherical surface of fixed radius. However, in the near-field it is more useful to make plots over other surfaces on which the power flux may be incident or traversing, or along a line in the direction of propagation. In the first case the ‘surface’ might be the intended coverage area for the antenna or a receiving aperture. By integrating the power flux density over the surface, the total intercepted power can be calculated. In the second case, a power profile can be plotted as a function of distance from the antenna, which is helpful when establishing an antenna’s useful operating range or for radiological safety considerations. The direction of maximum radiation [Theta,Phi] is now (v2.5) returned by the directivity function calc_directivity.m, subject to the calculation resolution parameters [d(theta),d(phi)]. Figure 3-4 shows some of the output generated by the excoverage.m example file. This example illustrates both of the applications mentioned above.

The function plot_field_slice.m is provided to define a rectangular plane of arbitrary dimension, orientation and position, on which is plotted the requested data. The plane dimensions are defined in ‘local’ x,y co-ordinates (ArrayCalc’s X-Y plane) and then rotated and translated into the desired position in ‘global’ co-ordinates (ArrayCalcs’s full X,Y,Z space). If the eventual orientation is parallel to one of ArrayCalc’s primary planes (XY,XZ or YZ) the plots are in ‘global’ co-ordinates. Otherwise plots are made using the ‘local’ co-ordinates, see plot_field_slice.m help for more details.

The primary field quantities are calculated by a call to the function fieldsum.m. In order to avoid excessively large field values (due to the 1/r relationship for small r), there is minimum distance (lambda/(4*pi)) from the array element centres at which the field is calculated. For points on the field slice below this distance, the calculation just uses the minimum value mentioned above. The variable used for the minimum distance is rlocmin and is set at the beginning of the function fieldsum.m.

Although there may seem to be a lot of parameters to supply to the field_slice.m and related functions, it does give maximum flexibility. If separate parameter variables are defined before the function call, these can be used rather than a long list of numbers and should make the code more readable. See expointsrce1.m, exrtpslice.m and excoverage.m examples. To get the hang of using the rotations and offsets try experimenting with exdipoles.m, it is a very simple geometry (2 dipoles) and runs quickly.

4. WAVE SLICE AND WAVE ANIMATION

In some cases traditional plots of field amplitude and phase do not convey very well how the wave front is formed and propagates. To visualise this, 2 functions have been added, plot_wave_slice and plot_wave_anim.

The plot_wave_slice function uses 2D field_slice data (see previous sections) for phase and magnitude in dB and uses it to plot a Cosine surface function. The surface function is calculated using the field_slice local co-ordinates and is of the following form :

WaveAmp=Amp*cos(Phase)

Where :
Amp is 20*log10(E(x,y,z))*SF
(SF is to normalise and scale max(WaveAmp) to lambda)

Phase is Phase(E(x,y,z))

The local data is then transformed to global co-ordinates so that the wave amplitude is normal to the plane of the field_slice.m. The result is a wave that is wave that is well scaled in x,y and z and decays nicely with distance from the source. The objective of this function is purely for visualisation and has been optimised with this in mind.

The plot_wave_anim.m function makes repeated calls to plot_wave_slice.m, incrementing the phase of the array elements for each frame. The frames are then animated using Matlab's 'movie' command. The phase increment, frame rate and view angle are set using a global variable waveanim_config, see init.m for details.

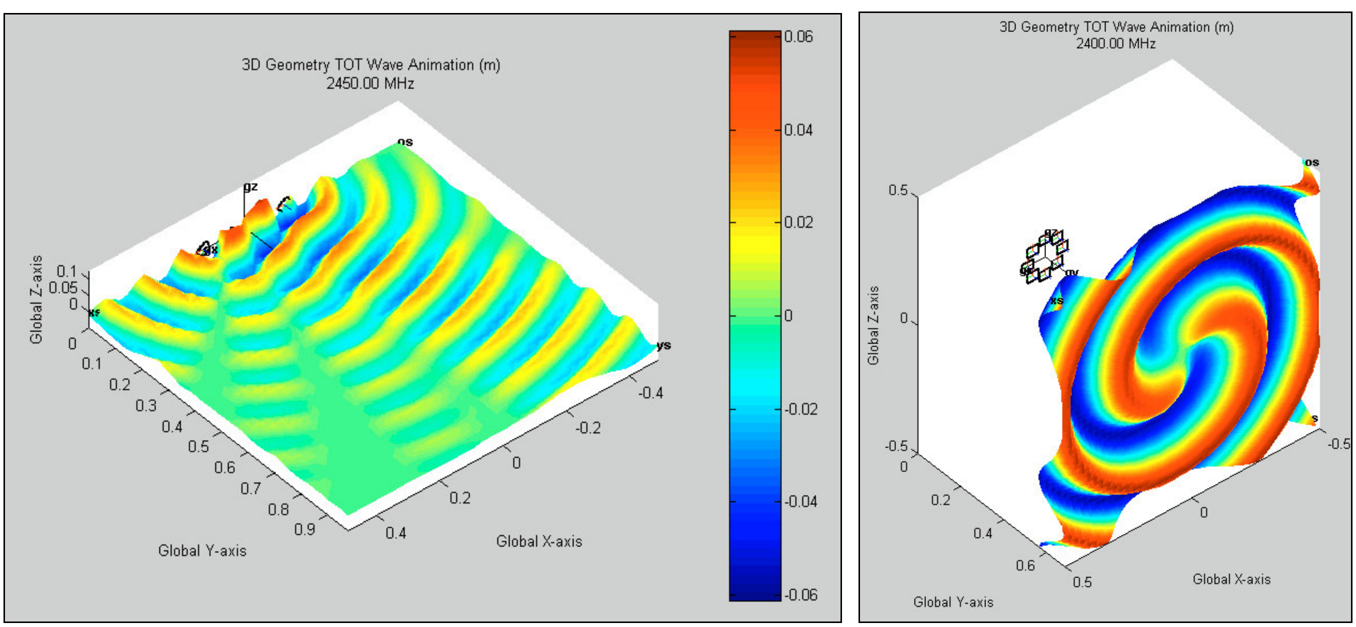


Figure 4-1 Output from animation examples : exanim2.m (left) and exanim3 (right)

5. ULTRASOUND APPLICATIONS

While ArrayCalc is intended primarily for EM antenna array analysis, it was written with a view to analysing wave propagation in any homogeneous medium. In ArrayCalc v2.5 all the element models now use the global variable for wave propagation velocity velocity_config rather than a fixed value of $3x10^8$. Obviously some careful thought has to be given as to how applicable the element models are to your intended application. The aperture based models 'aprect', 'apcirc' and 'dish' are fairly generic and can be used for ultrasound applications. In addition to these, the 'user1' element is now a half-isotropic (hemispherical radiation pattern) element, which can be useful as an infinitesimal source to sample larger apertures.

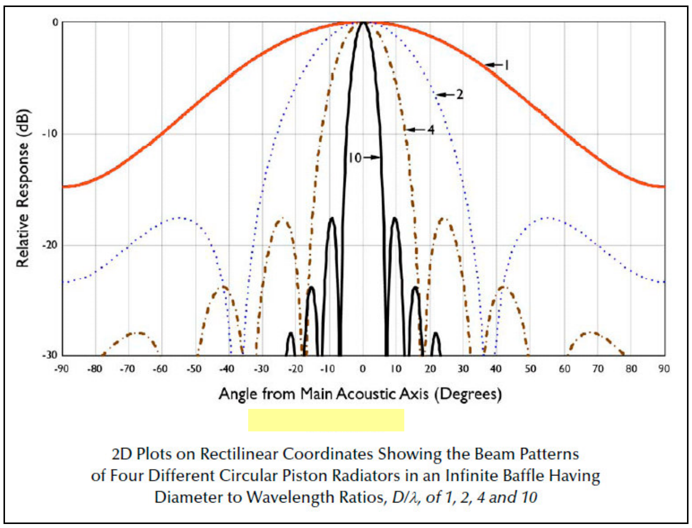


Figure 5-1 Circular Transducer Patterns [4]

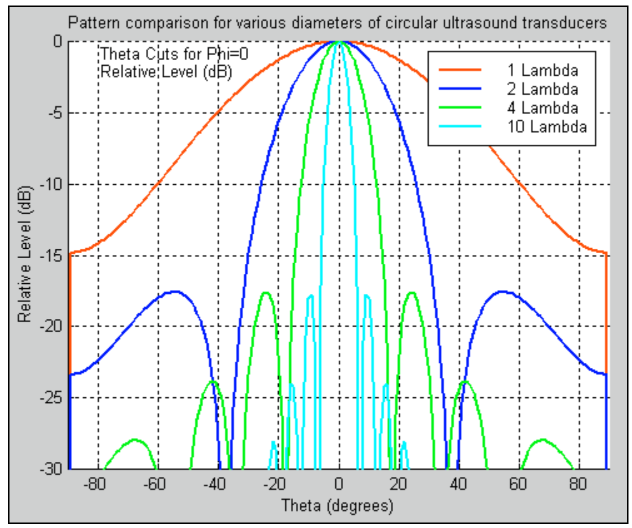


Figure 5-2 Circular aperture patterns ArrayCalc

The graphs in figure 5-1 show response patterns for circular ultrasound transducer patterns for a variety of diameter to wavelength ratios [4]. The graphs in figure 5-2 are E-plane pattern cuts generated by the extransducer.m example in ArrayCalc. The two figures are essentially the same and demonstrate that with some care, the 'apcirc' aperture model can be used in ultrasound applications. Note there are differences in the patterns when the EM model patterns are in the H-plane rather than E-plane, most notable for diameters less than 10*lambda.

The principal difference with ultrasound applications (e.g. medical imaging / treatment) is the significantly lower propagation velocity e.g. 1522m/s approx for sound in salt water. Assuming this velocity, this results in very short wavelengths (0.4mm) for quite modest frequencies (e.g. 3.5Mhz). This gives rise to large apertures in terms of wavelength (35 lambda), for an array that is still quite small physically (e.g. Diameter 14mm). All this means that the far-field distance $2*D^2$/lamda (0.98m) can often be outside the desired operating region, typically a few cm for an internal ultrasound imaging probe. To overcome this, the arrays are often geometrically focused, so the beam is concentrated in a region closer to the array. Figure 5-3 to 5-10 show output from the exultrason3.m example for un-focused and focused conditions (3 concentric sections f1=20mm, f2=25mm and f3=30mm).

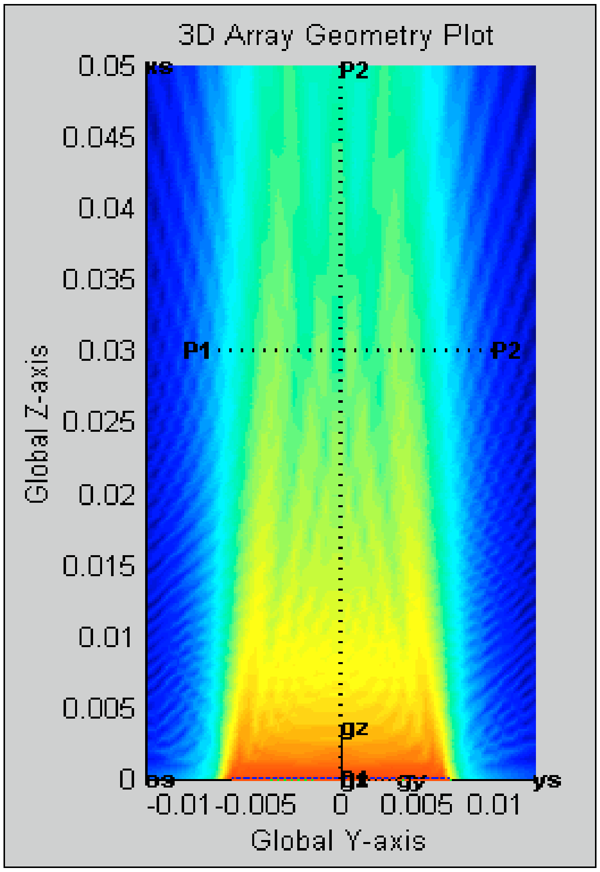


Figure 5-3 2D field slice un-focused

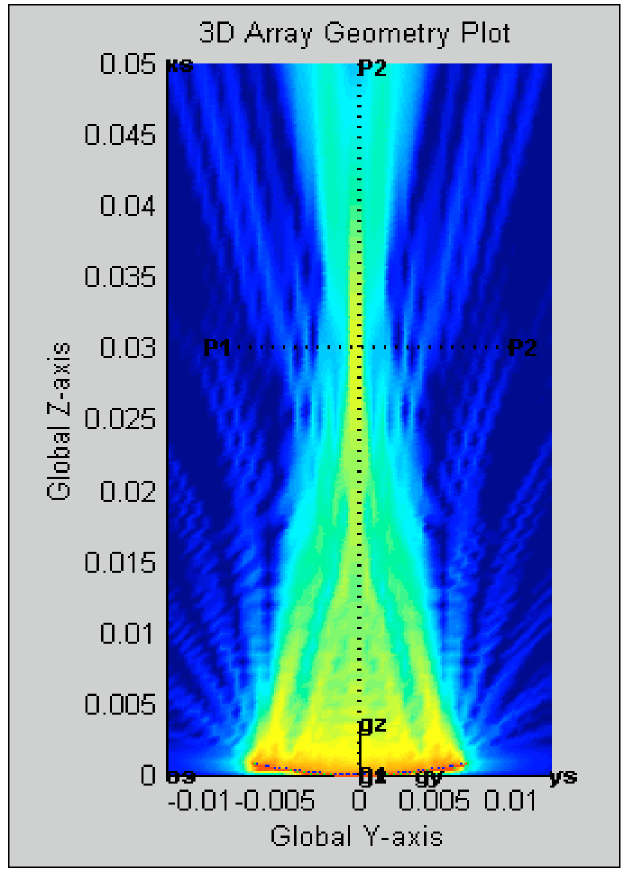


Figure 5-4 Focused

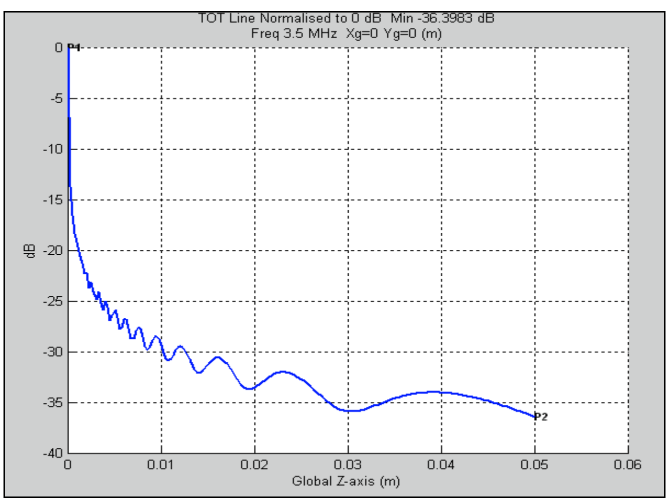


Figure 5-5 Beam axis power profile un-focused

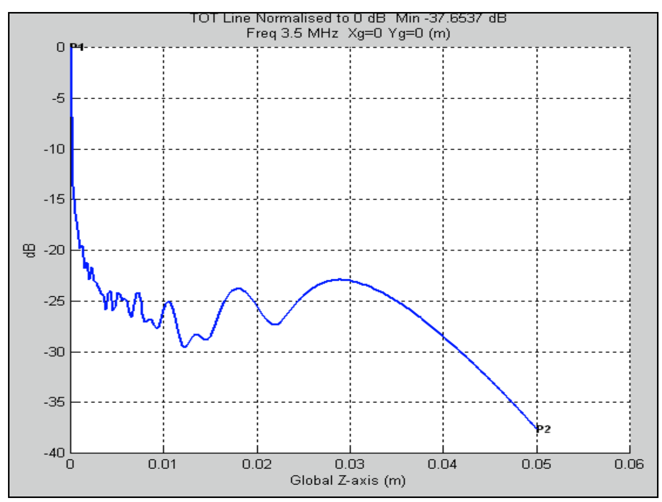
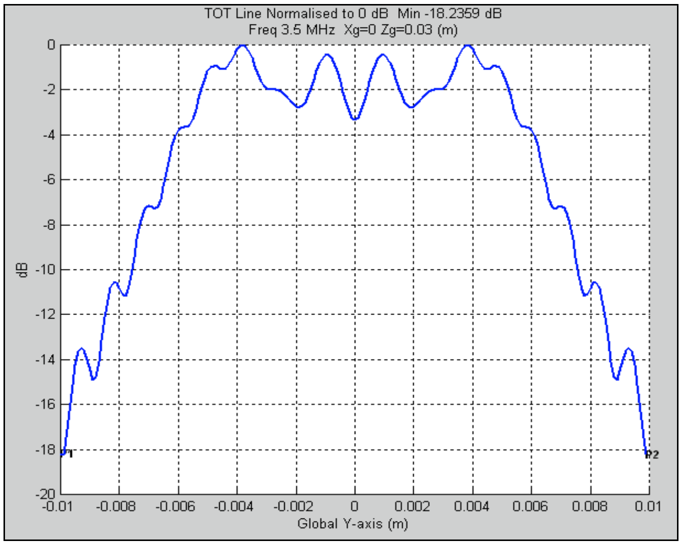


Figure 5-6 Focused



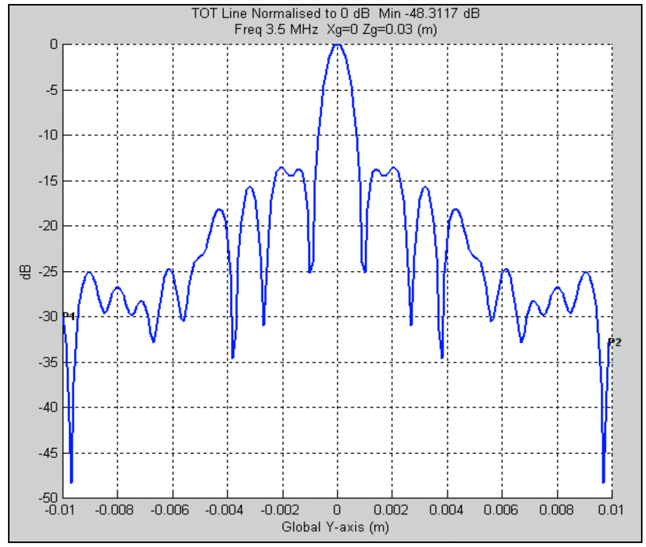
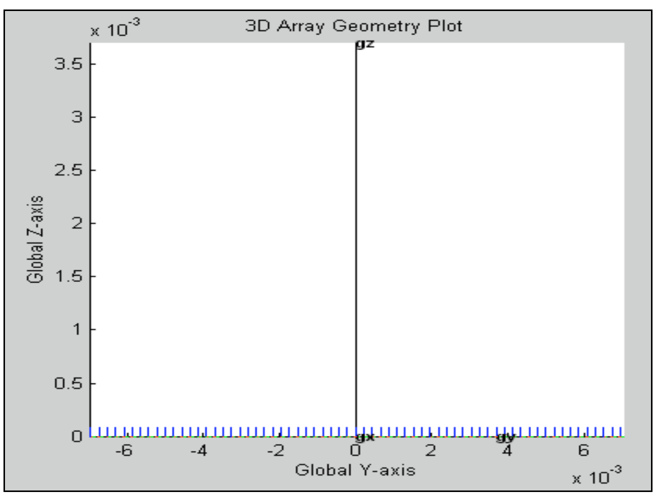


Figure 5-7 Beam cross-section at 30mm un-focused

Figure 5-8 Focused



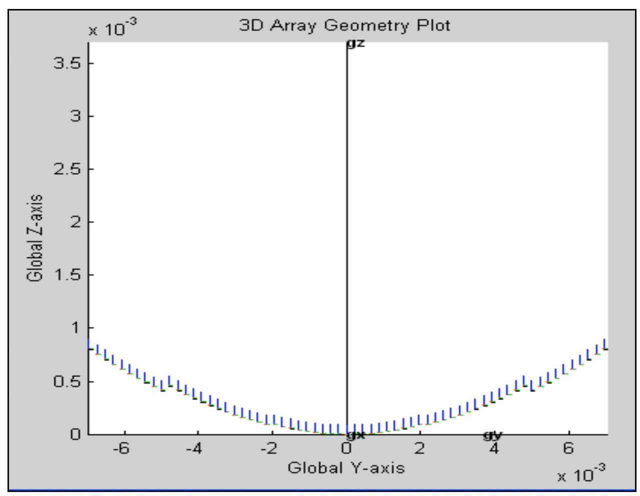


Figure 5-9 Array geometry un-focused

Figure 5-10 Focused

The benefits of focusing the array include an increased beam intensity in the region of interest, together with a better defined beam profile in cross-section. There are now a wide variety of ultrasound probes available on the market, many are highly optimised for particular diagnostic and therapeutic applications.

The plots in figures 5-5 to 5-8 are scaled in dB power relative. As with the EM field parameters, the use of the directivity calculation and an efficiency value for the array will allow the calculation and plotting of absolute field parameters. See the table in figure 5-11 for Acoustic equivalents of EM parameters, *indicates those used in ArrayCalc. The table in figure 5-12 shows the acoustic properties of some different materials. Have a look at the lecture notes by Ben Cox [5] for some good background information on acoustic theory.

EM Parameter (Units)	Equivaluent Acoustic Parameter (Units)
Charge (Coulombs)……………………………	Particle displacement (m)
Current (Amps)…………………………………	Particle velocity ($m.s^{-1}$)
Voltage (Volts)…………………………………	Acoustic pressure ($N.m^{-2}$, $Kg.m^{-1}.s^{-2}$, Pascals)
Voltage field ($V.m^{-1}$) *…………………………	Acoustic pressure change per metre ($Kg.m^{-2}. s^{-2}$)
Power (Watts) *………………………………..	Sound Power (Watts)
Power flux density ($W.m^{-2}$) *…………………	Acoustic Intensity ($W.m^{-2}$)
Impedance (Ohms) *………………………….	Acoustic Impedance ($Kg.m^{-2.}s^{-1}$, Rayleighs)

For info :- Acoustic impedance = *PoCo*
Where : *Po* = material density ($Kg.m^{-3}$) and *Co* = speed of propagation ($m.s^{-1}$)

Figure 5-11 Table showing acoustic equivalents of EM parameters

Material	Velocity (m/s)	Acoustic Impedance (Rayleighs)
Air (20degC)	343	413
Fresh Water (20degC)	1482	1.48e6
Sea Water (25degC)	1531	1.569e6
Aluminium 6262-T9	6380	17.41e6
Stainless Steel 347	5790	45.7e6

Figure 5-12 Table of acoustic properties of different materials

6. PHASE AND POWER CONVENTIONS

Phase Convention

In terms of differences between ArrayCalc v2.5 and previous versions, the phase convention for defining the element excitations is probably the most important.

All versions of ArrayCalc use the standard e^{-jwt} / e^{-jkr} convention for wave propagation. This means phase is –ve for increasing time or distance. The difference between v2.5 and previous versions is in the definition of the element excitations. In this version the element excitations conform to the same definition, so a phased delayed element will also have –ve phase. This is a little counter-intuitive if you are used to designing arrays with fixed feed networks, because to retard the phase you need to add transmission line length. This was why in previous versions of ArrayCalc +ve phase meant adding delay.

However, in the interests of making it easier to compare results with other packages, the (–ve phase) = (phase delay) convention has been adopted. All ArrayCalc's internal calculations and commands such as squint_array and focus_array, that calculate phase values, have been modified. Therefore, models developed on earlier versions that use these commands, should work without changes. However, models that use standard Matlab code to calculate phase excitations may require the sign of element excitation phases to be changed. All ArrayCalc examples have been modified as required.

Power Convention

A difference that does remain between ArrayCalc and some full-wave solvers, is the way in which input power is treated. In ArrayCalc, the global variable arraypwr_config represents the actual input power into the array in Watts. So, if an array has an input power of 100W and is 100% efficient , it will radiate 100W.

By contrast, in some full-wave packages a 'port' or 'probe' type excitation is used. The excitations are defined as a source current or voltage into/across an input impedance that has been calculated for that particular antenna. Some packages request a characteristic impedance value for the source (e.g. 50ohm) and take into account the mismatch-loss into the antenna. Other packages e.g. NEC2 assume a conjugate match, whatever the antenna's input impedance.

In either case, the power source has finite impedance, so even when conjugate matched, the input power is calculated as :

Power Input = (Iin * conj(Iin) * Re(Zin)) / 2

For voltage sources : Iin = Vin / Zin

Power is divided equally between the source and the load (the antenna in this case).

In other words, if you have a (50ohm + 0j) port impedance, you need to define a port current of (1A*sqrt(2) + 0j) to get an actual input power of 50W to the antenna.

In some cases, such as 4NEC2 [6], a pre-processing option allows the user to set the actual input power to the antenna and the necessary calculations are done for you.

Basically, what ever package you are using, check to see that the input power or port excitations are doing what you think are.

REFERENCES

[1] "Antenna Theory Analysis and Design" 2nd edition by Constantine A. Balanis. Published Wiley ISBN 0-471-59268-4

[2] "Microstrip Antennas" by I.J. Bahl and P. Bartia. Published Artech House `ISBN 0-89006-098-3

[3] "Advances in Micristrip Printed Antennas" Lee and Chen (Chapter5)

[4] "An Overview of Electroacoustic Transducers by Donald P. Massa" DPM_Overview_of_Electroacoustic_Transducers.pdf

[5] "Acoustics for Ultrasound Imaging Ben Cox January 2013" USlecturenotes_Jan2013.pdf

[6] "4NEC2" by Arie Voors based on the NEC2 code.