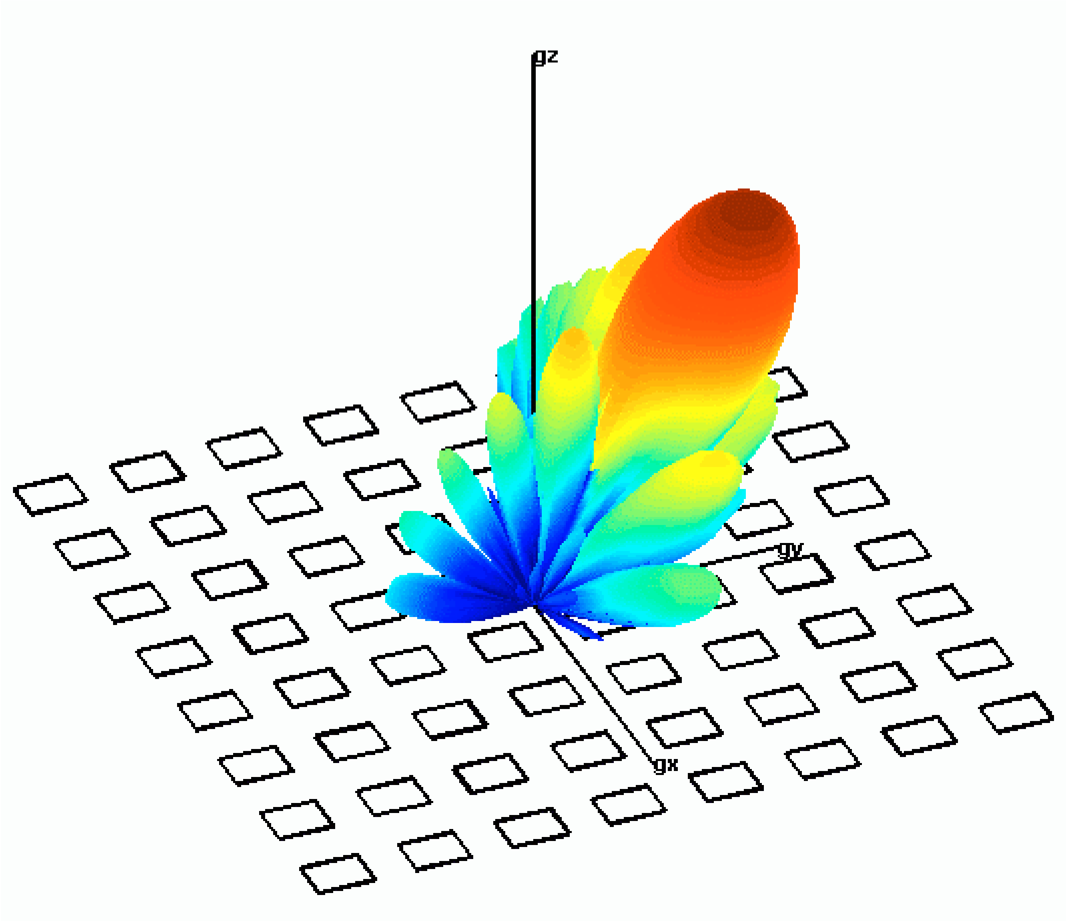
Array Calc V2.4

User Guide



CONTENTS

1. INTRODUCTION

This user guide is intended to help the user get the most out of the ArrayCalc toolbox by describing the toolbox structure and command formats and by working through a couple of tutorial examples. Although ArrayCalc is a versatile tool, it is based on a relatively simple process of vector summation and new users are recommended to read the 'Theory of Operation' document, to fully understand the limitations and advantages of this approach. For those who are pushed for time, there follows a brief description of the most important aspects.

The toolbox uses a graphical method to compute the array patterns. Individual array elements are placed in arbitrary 3D locations and orientations using a global co-ordinate system. A sphere centred on the global axis origin is the surface over which the array patterns are calculated. Calculating the distance and direction from each element to the appropriate points on the surface and summing the field contributions produces the patterns. The distance values give the phase component and the direction values give the amplitude component (by defining which part of the element patterns to use).

The radiating element models give only the total field information; the polarisation vector is organised to be in line with the local x-axis of the element. During the calculations, this information is resolved into vertical and horizontal components and allows pattern data to include polarisation information.

The main limitations of this approach are :

- All elements in the array must be of the same type.
- Mutual coupling is not taken into account.
- No voltages, currents or impedance information is available.
- Only far-field patterns can be calculated.

Despite these limitations ArrayCalc can provide a useful insight into the potential performance of an array, before committing to more detailed modelling or prototyping. ArrayCalc has been written the true spirit of 'toolbox' in mind and although much can be achieved with a few of the higher level commands, the user is encouraged to experiment with low level 'nuts and bolts', for maximum flexibility.

The first section of the guide deals with toolbox structure and command hierarchy; the middle section is a detailed command reference and the last section contains the tutorial examples.

2 INSTALLATION

Requirements : PC running Matlab 5.1 or later.

To add the ArrayCalc V2.4 toolbox to Matlab :

Unzip the files, resulting in a directory structure that should look like this :

```
Directories : ArrayCalc
                  |___ Beam_synthesis
                  |___ Documentation
                  |___ Element_indexing
                  |___ Element_models
                  |___ Examples
                  |___ Geometry_construction
                  |___ Plotting_visualisation
                  |___ Subroutines
                  |___ Validation
```

Ideally the ArrayCalc directory is placed in the Matlab toolbox directory as C:\Matlab\toolbox\ArrayCalc.

If not then the validation examples val1-6 & ex3a will not run, the path to the NEC output files will need to be changed in LoadNecPat1.m in the Validation directory. Type help LoadNecPat1 for more information. Apart from these provisos, the exact location is entirely up to the user.

The next step is to add the ArrayCalc directory, together with all the sub-directories to the Matlab search path. Exactly how this is done will depend on what version of Matlab you are running. For older versions (e.g. V5.1) you will have to add each path individually, on later versions (e.g. V6) you can just select the ArrayCalc folder using the ‘add with sub-folders’ option.

If the ArrayCalc root folder is ‘visible’ to Matlab then typing ‘help arraycalc’ should print the contents.m file to the screen. Typing ‘exlist’ at the prompt will print out a list of examples and if all is well you should be able to run the examples by typing the filenames at the prompt e.g. ‘ex1’

ArrayCalc has been written using basic Matlab 5.1 (1997) with the idea that it will run on as many subsequent versions of Matlab as possible. Due to the constant development of the Matlab environment I believe some compatibility issues have arisen with later versions. Thus far (2010), these only appear to result in warning messages, mostly notifying the user of newer command formats. Assuming there are no fatal errors, the warnings can be ignored and the screen clutter can be reduced by using the ‘warning off’ command.

3 TOOLBOX STRUCTURE

The ArrayCalc toolbox has evolved from a requirement to be able to plot radiation patterns from arbitrarily arranged, arbitrarily excited arrays of radiating elements. The toolbox structure was largely defined by identifying the minimum information needed to describe the problem, and then looking at simplest way to pass that information between the various processing functions. The diagram in figure 3-1 below shows how the commands in the various directories interact with each other. The light green boxes indicate the typical program flow for user scripts.

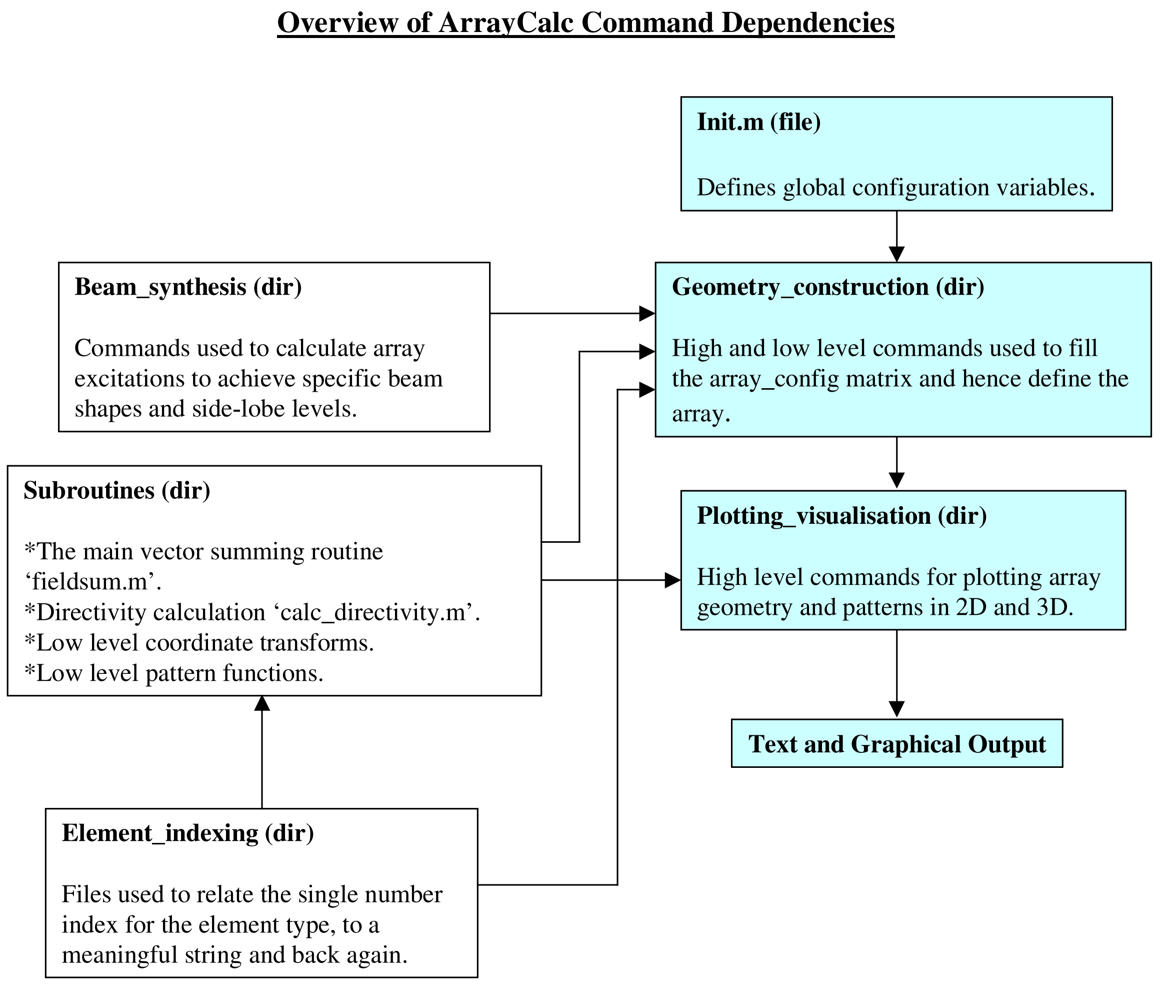

Overview of ArrayCalc Command Dependencies
Init.m (file)
Defines global configuration variables.
Beam_synthesis (dir)
Commands used to calculate array excitations to achieve specific beam shapes and side-lobe levels.
Geometry_construction (dir)
High and low level commands used to fill the array_config matrix and hence define the array.
Subroutines (dir)
*The main vector summing routine 'fieldsum.m'.
*Directivity calculation 'calc_directivity.m'.
*Low level coordinate transforms.
*Low level pattern functions.
Plotting_visualisation (dir)
High level commands for plotting array geometry and patterns in 2D and 3D.
Text and Graphical Output
Element_indexing (dir)
Files used to relate the single number index for the element type, to a meaningful string and back again.

Figure 3-1 ArrayCalc command dependencies

The following sections of this guide detail the principal components for any user script, namely :

- Initialisation
- Geometry Construction
- Plotting and Visualisation

3.1 THE INIT.M FILE

The first call in any ArrayCalc script should be init.m as this defines a series of global configuration variables of the form : varname_config=[param1, [param2…etc]]

Although initialised in init.m, all of the configuration variables can be changed as required within the user script.

3.1.1 Array_config

Starting with a generalised array description; we need to know the 3D position and orientation of each element, what type of elements they are and finally the excitation for each of the elements (phase & amplitude). Since Matlab works with arrays it was logical to use a multi-dimensional matrix to store the array description.

For an N-element array there is a 3x5xN matrix called “array_config” describing the orientation, position, excitation and element type; the matrix is configured as shown below.

Each of the N elements has an entry : L M N Xoff Amp
O P Q Yoff Pha
R S T Zoff Eltype

Where : L M N Xoff
O P Q Yoff is the 3D rotation matrix and offset in (meters)
R S T Zoff

Amp	Element amplitude (linear volts)
Pha	Element phase (radians)
Eltype	Element type (integer) 0,1,2…representing which model to use.

Note : array_config is initialised with the command array_config=-ones(3,5,1). The –ve sign lets the other functions know that the array is empty and the next element to be added will be the first. Unless otherwise stated, geometry construction functions append the new geometry to the existing array description.

3.1.2 Freq_config and Velocity_config

The freq_config and velocity_config variables set the frequency (Hz) and wave velocity (m/s) respectively, at which the array is to be analysed. Although array geometries and element models are often defined in terms of wavelength fractions, ArrayCalc uses physical distances in its internal calculations. Defining the frequency and velocity allows easy conversion to physical dimensions and the possibility to model different propagation mediums e.g. sound in air or water.

3.1.3 Range_config

This variable defines the radius of the spherical surface on which the patterns are evaluated. The default value is 999m and will be adequate for many applications, however there are certain situations where it may be necessary to change the value.

To ensure the pattern calculations represent the far-field the value should satisfy the relation range_config > 2*D^2/Lambda, where D is the maximum array dimension in meters. I have found that the maximum value for range_config is around 2e6 (m). This corresponds to an aperture of 1000 Lambda and hence the maximum antenna aperture that can be analysed. Beyond 2e6(m) and the internal trig calculations begin to break down.

It is possible to look at the pseudo near-field of an array by reducing the value of range_config. This is obviously not the true near-field but it does show the effects of quadratic phase error, as would be seen if an array was measured on a far-field test range that was not large enough.

3.1.4 Direct_config and Normd_config

The direct_config variable is used to store the result of the calc_directivity function and represents the directivity of the array in dB relative to isotropic (dBi). The directivity is calculated using numerical integration.

The normd_config variable stores the maximum pattern value and is used to normalise the plots, when required.

3.1.5 Phaseq_config

This variable defines the number of binary digits (bits) to use when scanning the array using the squint_array function. The requested squint angle is not used directly but quantised in steps of 360/(2^n) Deg, where n=phaseq_config. The default setting is 16 giving a step size of 0.0055 Deg

By changing this variable it is possible to assess the effect digital type phase shifters on array performance.

3.1.6 Element Configurations

Although the “array_config” matrix is a compact way of describing the array configuration, the element description is limited to a single index number. To fully describe the elements, each one has its own configuration variable. The variables are single vectors of the basic element parameters e.g. patchr_config=[Er,W,L,h].

The element configuration variables are used in the element models to set the model parameters. This allows realistic element patterns to be calculated, based on a simplified physical description of the element. A full description of the element models can be found in the ‘Theory of Operation’ document. All elements with the exception of the helix are linearly polarised in line with the element’s local x-axis.

Note that although initialised in the init.m file, the user can change any of the element configuration variables in isolation at any time. Once defined, this will be the element ‘design’ referred to by the string parameters in the geometry construction commands, see below for the list.

STRING	VALUE IN array_config
'iso'	0
'patchr'	1
'patchc'	2
'dipole'	3
'dipoleg'	4
'helix'	5
'aprect'	6
'apcirc'	7
'wgr'	8
'wgc'	9
'dish'	10
'interp'	11
'user1'	12

In order to make user scripts easier to read and to “decode” the Eltype index number used in array_config, a series of indexing functions are used. (see Element_indexing sub dir)

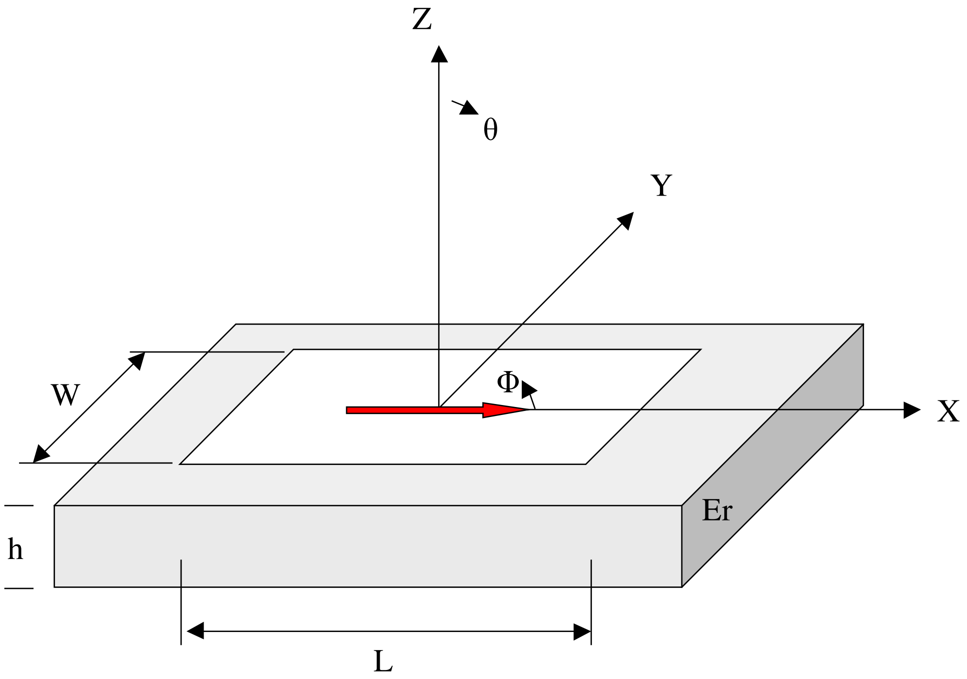
Eltcode : Takes a text string (e.g. ‘patchr’) and returns an index number (e.g. 1).
Namecode : Takes an index number (e.g. 1) and returns a text string (e.g. ‘patchr’).

Sumcode : Takes an index number and calls the appropriate element model function.

Geocode : Takes an index number and returns the graphics co-ordinates used to draw the elements.

Patchr_config

Configuration for rectangular microstrip patch.



Define patch directly using : patchr_config=[Er,W,L,h];

Er : dielectric constant

W : Width (meters)

L : Length (meters)

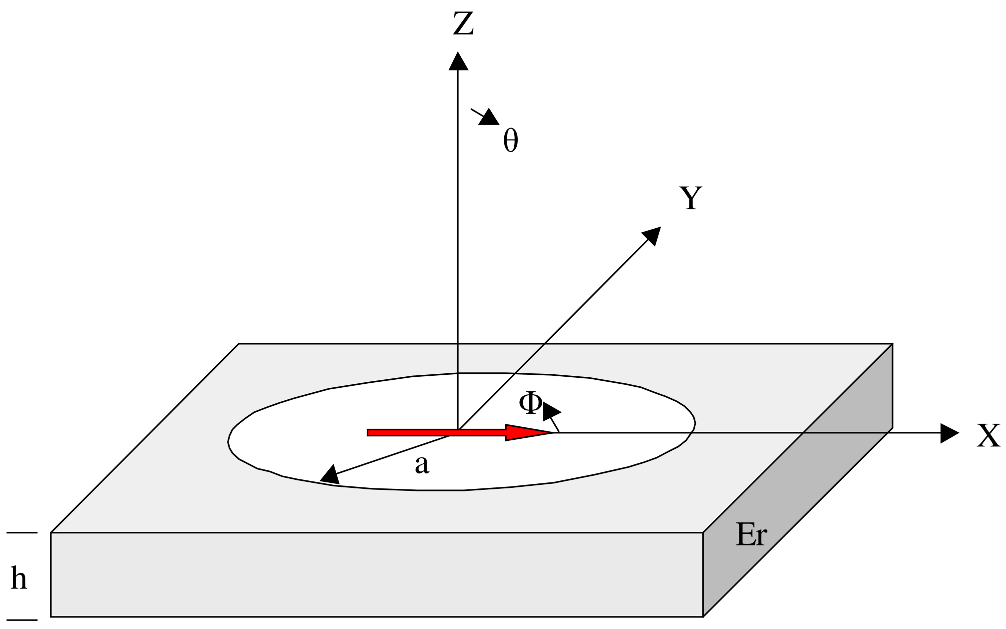
h : Substrate thickness (meters)

Or use the design rectangular patch function, which calculates the optimum width (W) and length (L) for the patch, given the dielectric constant (Er), substrate thickness (h) and frequency (Freq) in Hertz :

patchr_config=design_patchr(Er,h,Freq)

Patchc_config

Configuration for circular microstrip patch



Define patch directly using :   patchc_config=[Er,a,h];

Er : Dielectric constant

a : Patch radius (meters)

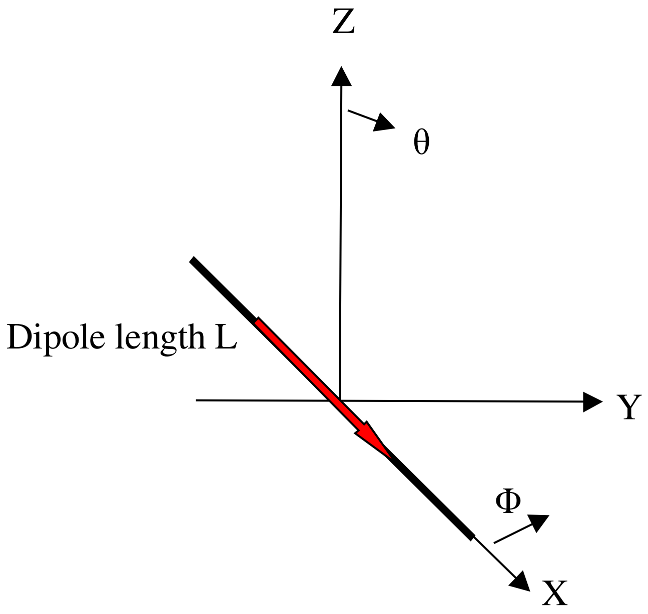
h : Substrate thickness (meters)

Or use the design circular patch function, that calculates the optimum radius (a) for the patch, given the dielectric constant (Er), substrate thickness (h) and frequency (Freq) in Hertz :

patchc_config=design_patchc(Er,h,Freq)

Dipole_config

Configuration for dipole.

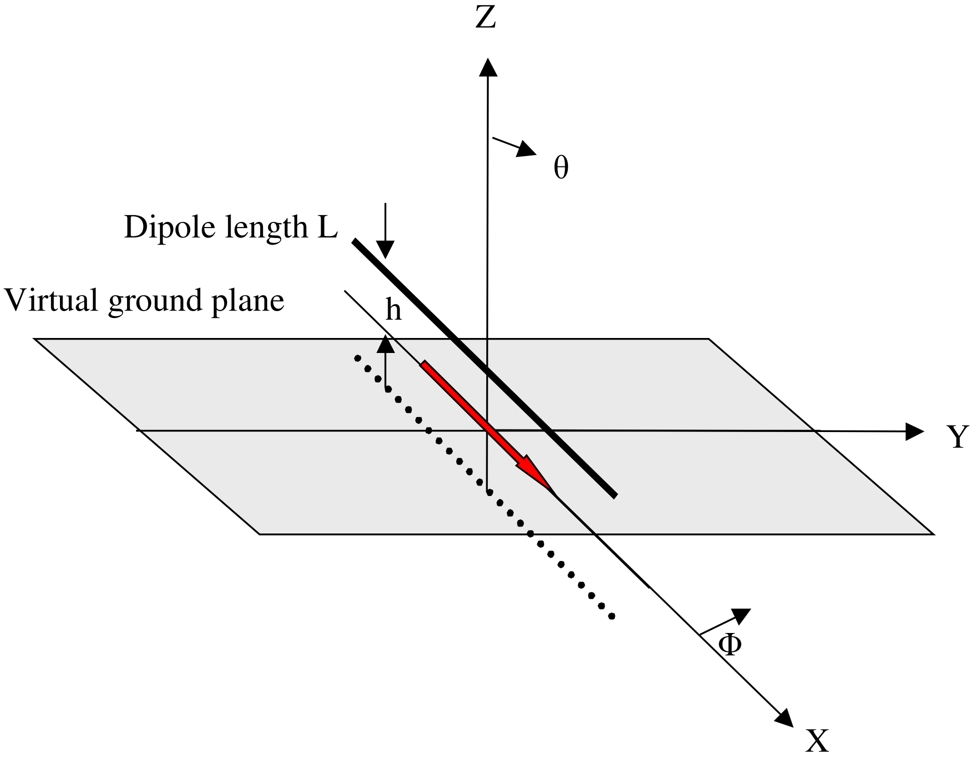


Define dipole using : dipole_config=[L];

L : length (meters)

Dipoleg_config

Configuration for dipole over ground plane.



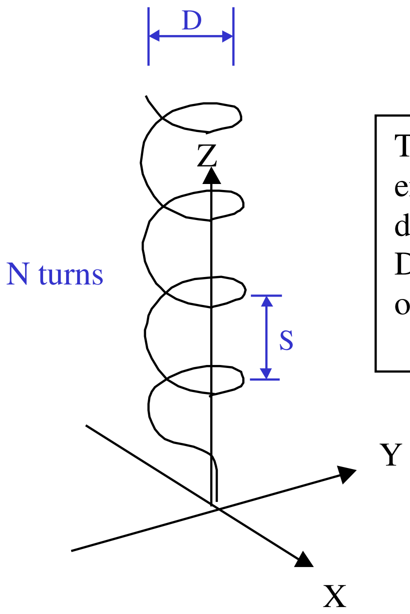
Define dipole over ground using : dipoleg_config=[L,h]

L : length (meters)

h : height above ground plane (meters)

Helix_config

Configuration for helix.



The helix model assumes conditions for endfire operation with increased directivity. This means that the diameter D is defined by default, given N,S and the operating frequeny.

Define helix using : helix_config=[N,S]

N : Number of turns

S : Turn spacing (meters)

Or use the design helix function, which calculates the optimum turn spacing given the number of turns (N) and operating frequency (Freq) in Hertz.

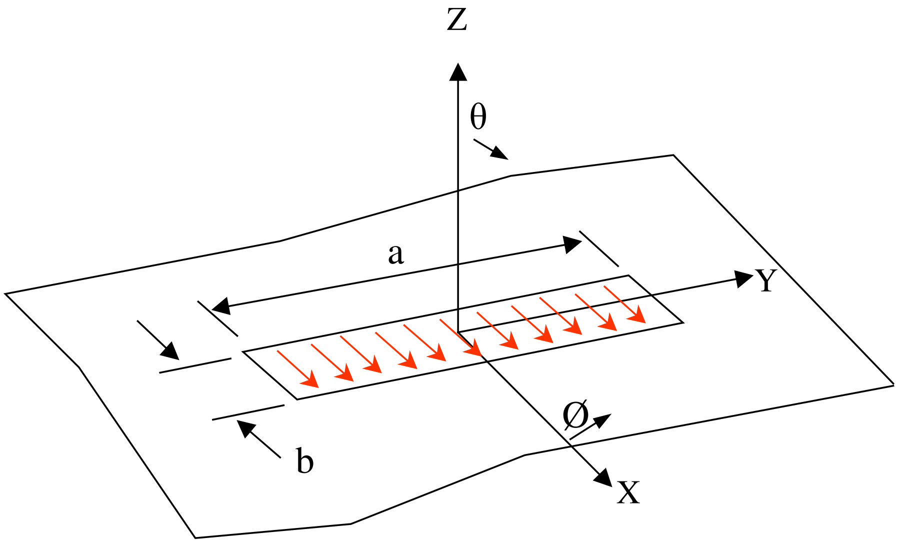
helix_config=design_helix(N,Freq)

Note :- The polarisation of the helix is inherently circular and ArrayCalc deals with this by setting the Circular Polarisation (CP) flag in the model. This has the effect setting the Horizontal and Vertical polarisation components equal to each other and –3dB down on the Total field pattern.

The ‘hand’ of polarisation of the helix is (RHCP) by default. This is defined by the CPflag=1 statement at the end of the helix.m file in the Element_Models directory. Use CPflag=-1 for LHCP. The element model has no X-polar component so plotting LHCP for a RHCP helix will give a ‘zero pattern’ i.e. –inf dB for all angles.

Aprect_config

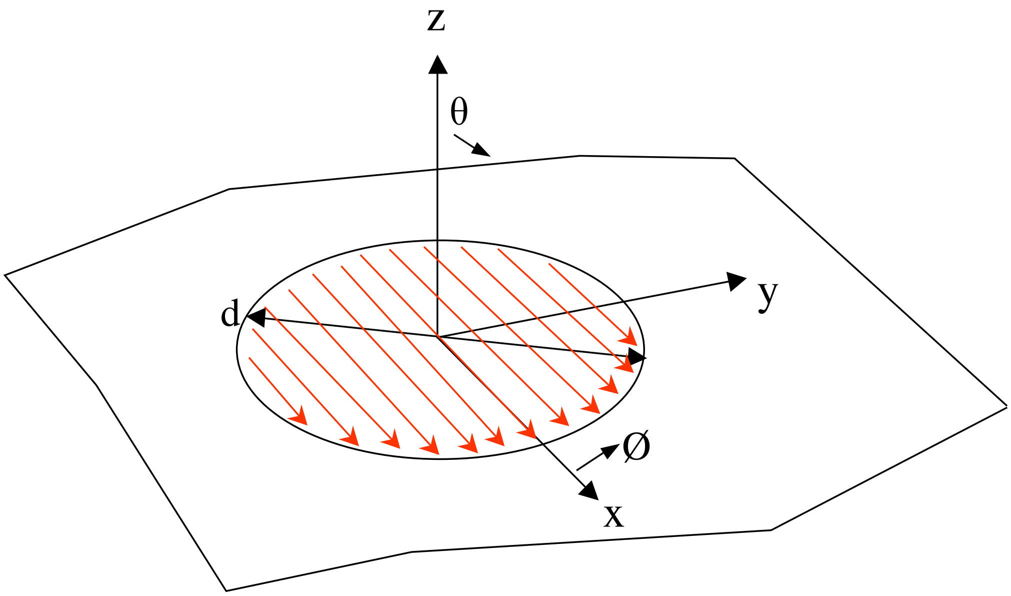
Configuration for rectangular aperture.



Define aperture using : aprect_config=[a,b] a : length (meters) b : width (meters)

Apcirc_config

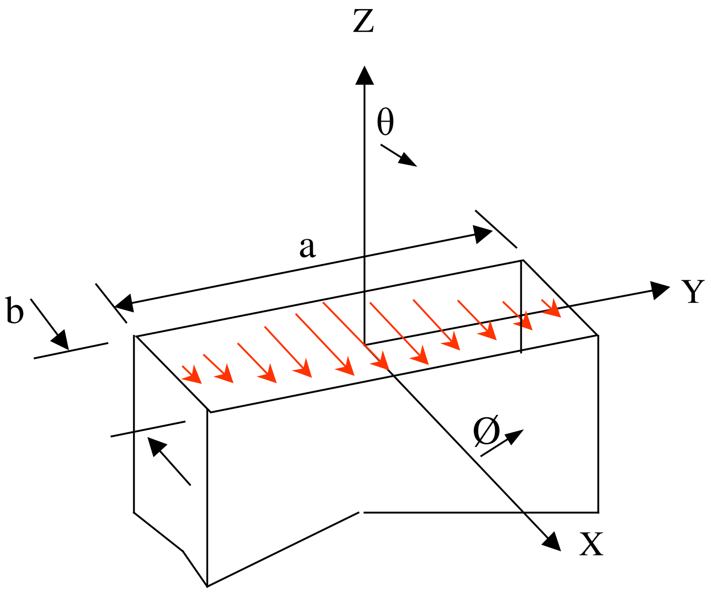
Configure for circular aperture.



Define aperture using : apcirc_config=[d] d : diameter (meters)

Wgr_config

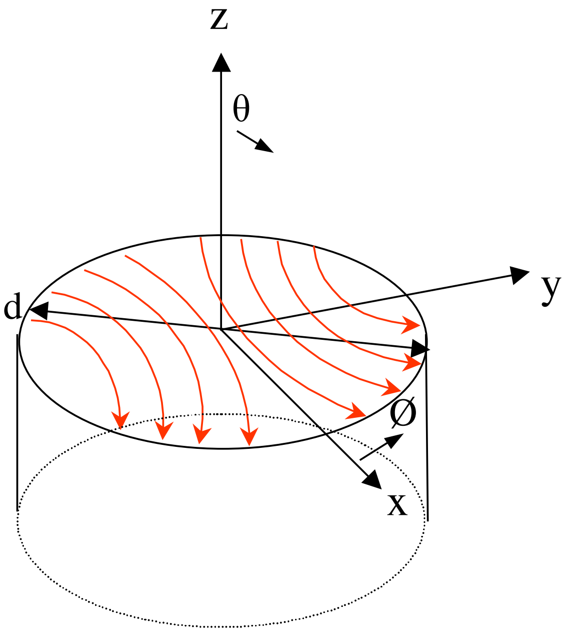
Configuration for rectangular waveguide.



Define waveguide using : wgr_config=[a,b] a : length (meters) b : width (meters)

Wgc_config

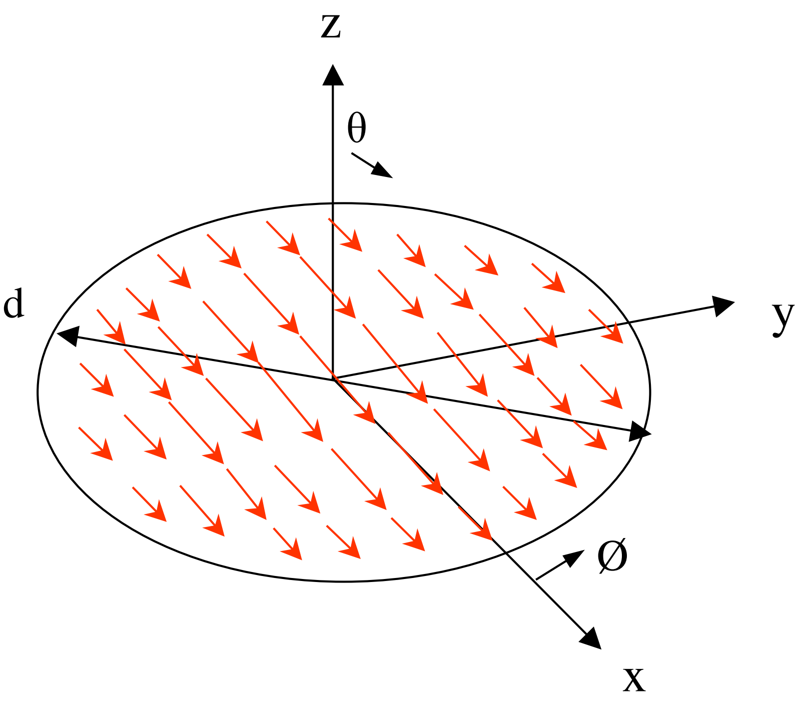
Configuration for circular waveguide.



Define waveguide using : wgc_config=[d] d : diameter (meters)

Dish_config

Configuration for parabolic dish.



Difine parabolic dish using : dish_config=[d,n,t]

d : diameter (meters)

n : taper factor (unitless). Defines rate of taper towards dish edge, typically 2.0

t : taper (dB). Power reduction at dish edge relative to centre, typically 10

4 COMMAND REFERENCE

This section of the user guide documents the higher level ArrayCalc commands and is subdivided into two main categories, geometry construction and plotting & visualisation. Full lists of the commands can be obtained by typing ‘help arraycalc’ at the command prompt (this will print the contents.m file to the screen). All of the commands have their own help files that can be listed by typing ‘help commandname’ at the Matlab prompt.

4.1 GEOMETRY CONSTRUCTION

The geometry construction commands are used to fill the array_config matrix, and it is this that constitutes the array description.

<u>Geometry Construction Command List</u>

Place_Element

Place a single element in specific orientation and location

Usage: place_element(n,xr,yr,zr,x,y,z,eltype,Pwr,Pha)

```
  n.......Element number (integer)
 xr......Rotation about X-axis (Deg)
 yr......Rotation about Y-axis (Deg)
 zr......Rotation about Z-axis (Deg)
 x.......X-coordinate (meters)
 y.......Y-coordinate (meters)
 z.......Z-coordinate (meters)
 eltype..Element type (string)
 Pwr.....Power (volts^2 in dB)
 Pha.....Phase (Deg)
```

Set element number n=0 to append element to existing geometry.
Valid strings for eltype are listed in the element configurations section 3.1.6.

```
      z
      |-theta   (theta 0-180 measured from z-axis)
      |/
      |_____ y
     /\
    /-phi       (phi 0-360 measured from x-axis)
   x
```

+ve rotation is defined as clock-wise looking from axis end towards
the origin of the axis set.

The xr,yr,zr rotations refer to sequential rotations about x,y and z axes. The rotation matrix is defined as :

Trot=XR*YR*ZR

Where are XR,YR and ZR are 3D rotation matrices.

If you need to use independent rotations, place the element with xr=yr=zr=0 and orientate element using : xrot_array, yrot_array or zrot_array functions.

Single_Element

Place a single element in specific location, appended to the current array configuration. Default E-plane for the element is the X-axis.

Usage: single_element(x,y,z,eltype,Pwr,Pha)

```
  x.......X-coordinate (meters)
  y.......Y-coordinate (meters)
  z.......Z-coordinate (meters)
  eltype..Element type (string)
  Pwr.....Power (volts^2 in dB)
  Pha.....Phase (Deg)
```

Valid strings for eltype are listed in the element configurations section 3.1.6.

E.g.   single_element(0,0,0,’patchr’,-10,90)

Excite_Element

Set element phase and Amplitude

Usage: excite_element(elnumber,Pwr,Pha)

elnumber..Element number
Pwr.......Pwr (volts^2 in dB)
Pha.......Phase (Deg)

E.g. excite_element(1,-10,90)

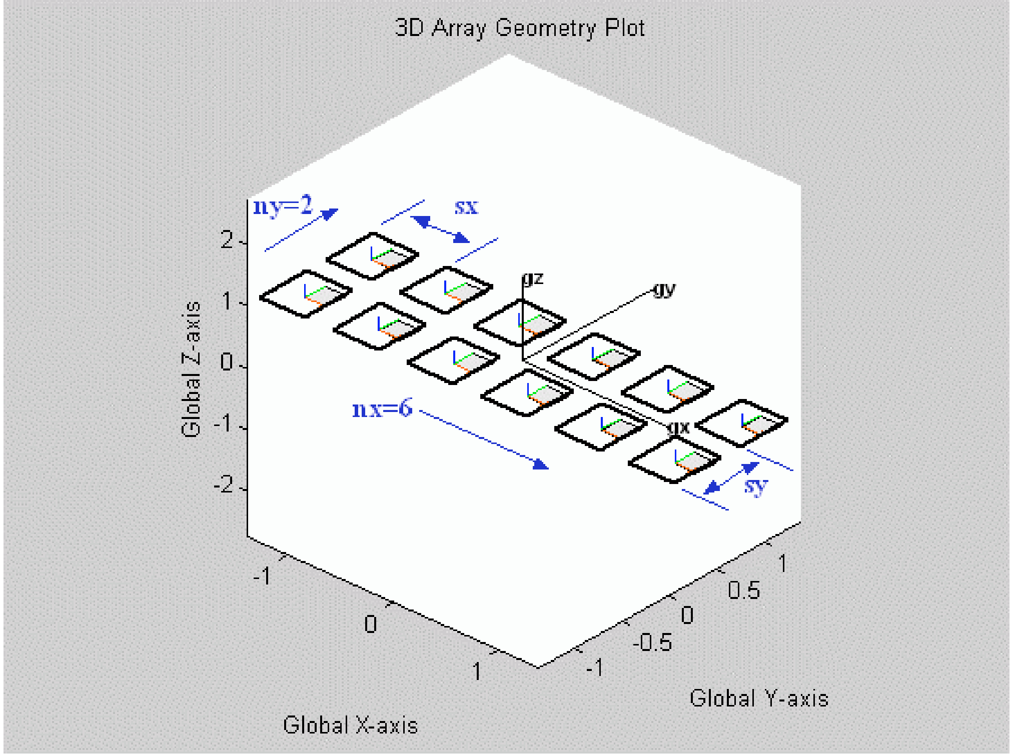
Rect_Array

Define a rectangular array

Usage: rect_array(nx,ny,sx,sy,eltype,Erot)

nx………Number of elements in x-direction
ny………Number of elements in y-direction
sx………Element spacing in x (meters)
sy………Element spacing in y (meters)
eltype…Element type (string)
Erot……E-plane angle from X-axis (Deg)

Valid strings for eltype are listed in the element configurations section 3.1.6.
E.g.   rect_array(6,2,(0.7*lambda),(0.7*lambda),'patchr',0)



Circ_array

Define a circular array using number of elements in first ring and element spacing to determine the number of elements in subsequent rings.

Usage: circ_array(nr,nrg,sr,srng,eltype,Erot,Efix)

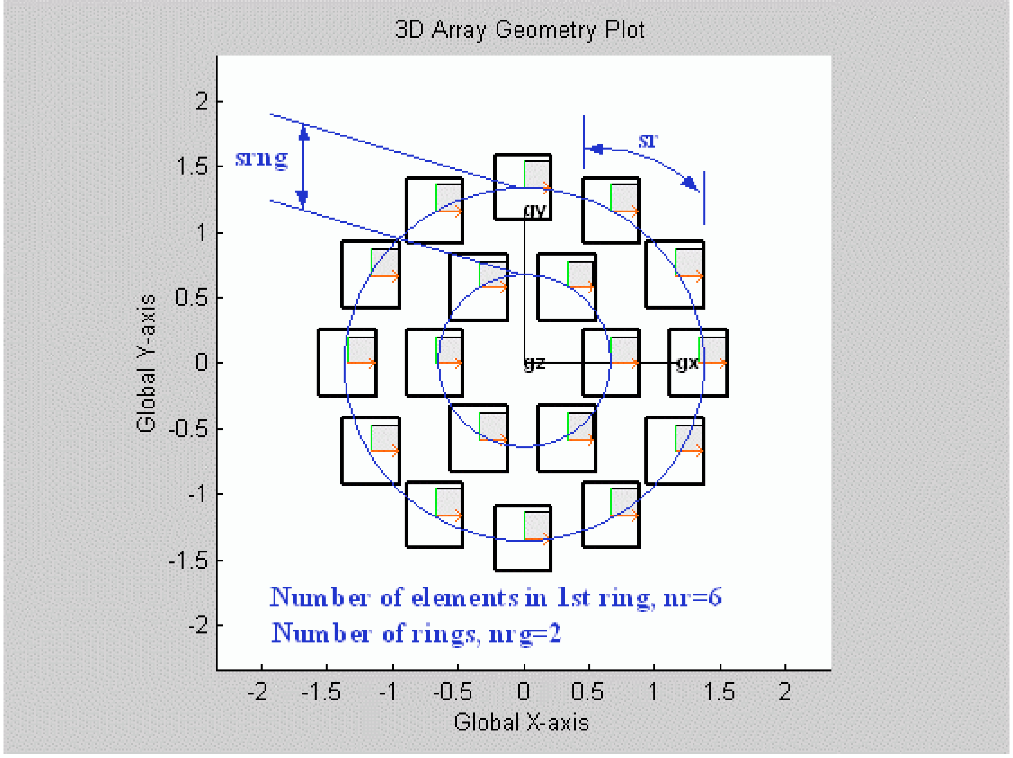
nr…….Number of elements in 1st ring (integer)
nrg……Number of rings (integer)
sr……..Element spacing around the ring (m)
srng....Spacing between rings (m)
eltype..Element type (string)
Erot…..E-plane rotation about Z-axis (Deg)
Efix…..E-plane rotation with ring angle (string)

Options for Efix are :

'yes' - Fixed E-plane rotation as defined by Erot
'no' - Rotate with ring angle, starting at Erot

Valid strings for eltype are listed in the element configurations section 3.1.6.

E.g.   circ_array(6,2,(0.7*lambda),(0.7*lambda),'patchr',0,'yes')



Cylin_Array

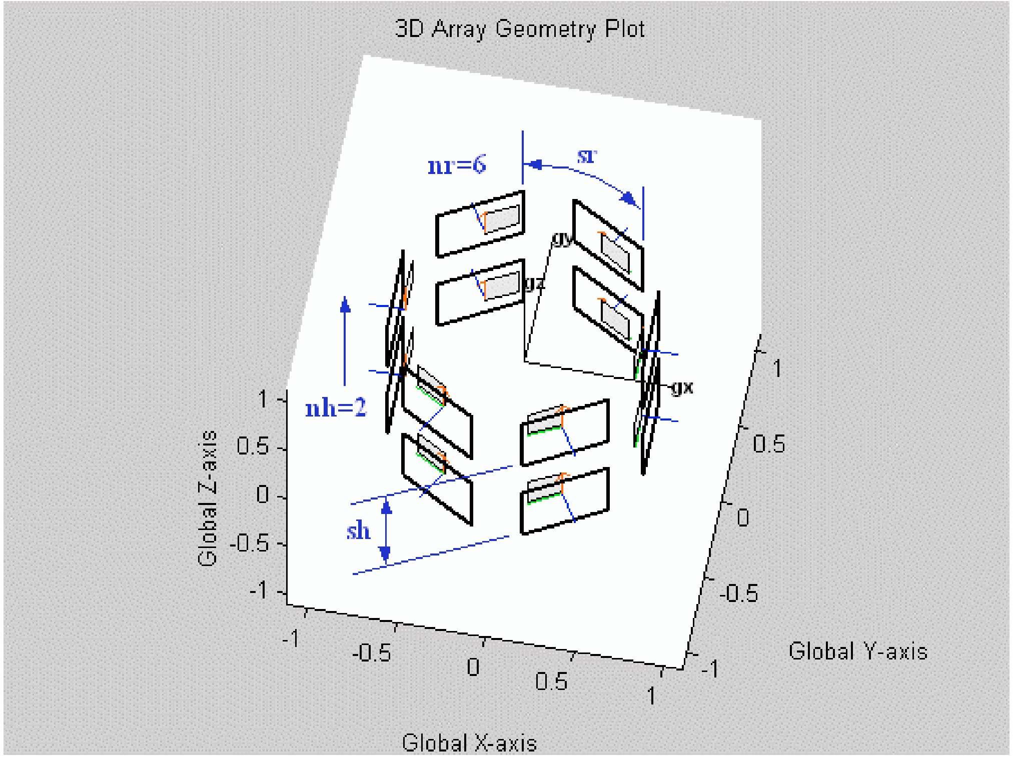
Define a cylindrical array

Usage: cylin_array(nr,nh,sr,sh,eltype,Erot)

nr…......Number of elements in each ring (integer)
nh..…....Number of rings (integer)
sr………Element spacing around the cylinder surface (m)
sh..…....Spacing between rings (m)
eltype...Element type (string)
Erot…..E-plane angle from Z-axis (deg)

Valid strings for eltype are listed in the element configurations section 3.1.6.

E.g. cylin_array(6,2,(0.7*lambda),(0.7*lambda),'patchr',0)



Rhomb_array

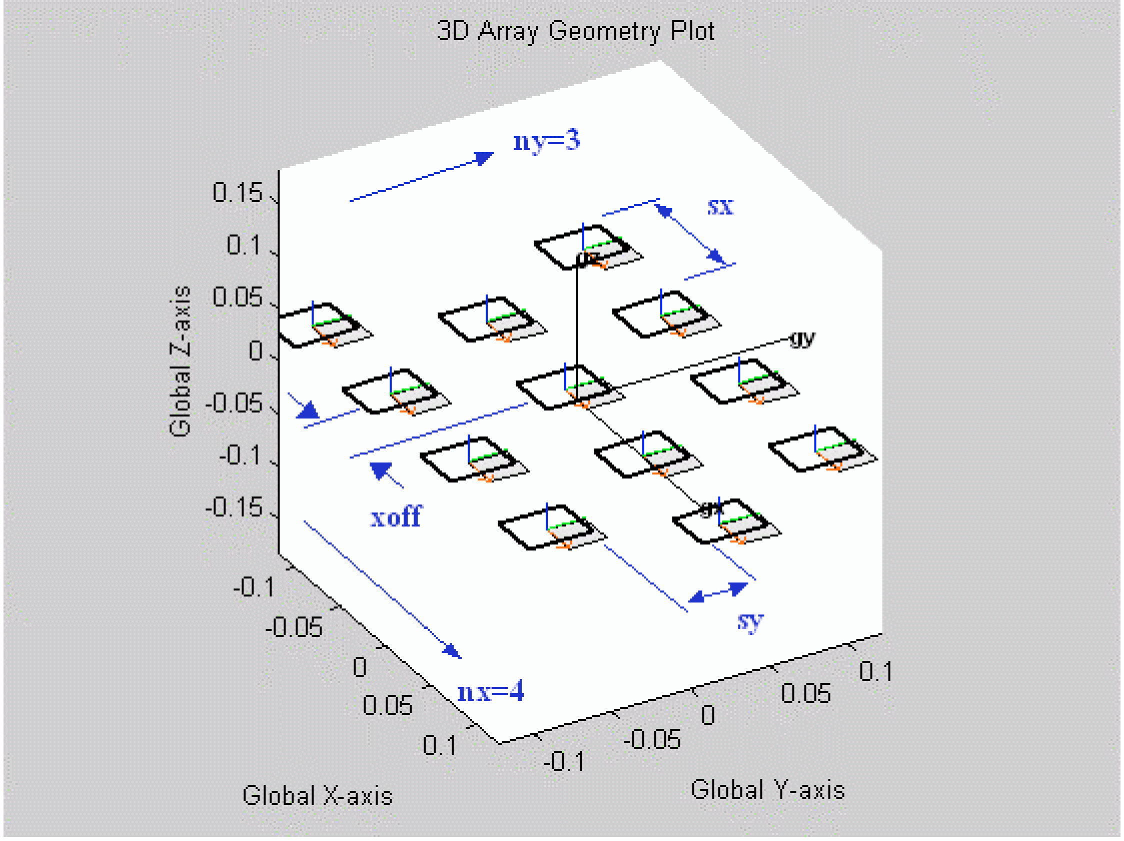
Define a rectangular array with offset alternate
rows / columns

Usage: rhomb_array(nx,ny,sx,xoff,sy,yoff,eltype,Erot)

nx…....Number of elements in x-direction
ny…....Number of elements in y-direction
sx..…..Element spacing in x (meters)
xoff…..Offset for alternate rows in x (meters)
sy..…..Element spacing in y (meters)
yoff…..Offset for alternate rows in y (meters)
eltype..Element type (string)
Erot…..E-plane angle from X-axis (Deg)

Valid strings for eltype are listed in the element configurations section 3.1.6.

```
E.g. rhomb_array(4,3,(0.7*lambda),(0.35*lambda),...
                        (0.7*lambda),(0.0*lambda),'patchr',0)
```



Squint_array

Adjusts phase excitations for each element in the array,
(as stored in array_config) such that they are equiphased
in the direction defined by spherical coords (theta,phi)

The calculated phases are quantised into steps of : 360/(2^n) Deg
The value of the number of bits (n) is stored in phaseq_config
and initialised in init.m
(For n=16 the step size is 0.0055 Deg)

Usage: squint_array(theta,phi,Elref)

```
  theta....Theta (Deg)
  phi……..Phi (Deg)
  Elref.....Element number to normalise phase to
```

```
E.g. squint_array(12,0,1)  % Squint array by 12 Deg towards X-axis
                                   in the X-Z plane

     squint_array(5,90,1)   % Squint array by 5 Deg towards Y-axis
                                   in the Y-Z plane
       z
        |-theta   (theta 0-180 measured from z-axis)
        |/
        |_____ y
       /\
      /-phi       (phi 0-360 measured from x-axis)
     x
```

Focus_array

Adjusts phase excitations for each element in the array,
(as stored in array_config) such that they are equiphased
at the distance and in the direction defined by spherical
coords (r,theta,phi)

Usage: focus_array(R,theta,phi,Elref)

```
  R........Focal distance (m)
  theta....Theta (Deg)
  phi......Phi (Deg)
  Elref....Element number to normalise phase to
```

```
E.g. focus_array(50,12,0,1) % Squint array by 12 Deg towards X-axis
                                   in the X-Z plane and set focal length to 50m
```

Taywin_array

Adjusts amplitude excitations for each element in the array, (as stored in array_config) to give monotonically decreasing sidelobe levels. Uses modified Taylor distribution.

Usage : taywin_array(R,xyr)

R....Sidlobe level for 1st sidlobe (dB)
xyr..Windowing direction (string)

xyr = 'x' to apply window along X-axis
xyr = 'y' to apply window along Y-axis
xyr = ‘xy’ to apply window along x and y-axis

xyr = 'r' to apply window as a function of distance from
the centre of the array. Use for circular arrays
or to window in x & y for rectangular arrays

Note :- For a uniform distribution the 1st sidlobe is already
13.2dB down, R should specify a value greater than this.

Also the distribution assumes isotropic sources, element directivity can further reduce sidelobe levels.

E.g.   taywin_array(20,'x') % For -20dB 1st sidelobe taper in x-direction

Move_array

Move array geometry in x,y,z, movement is relative to the current location.

Usage: move_array(x,y,z,elstart,elfinish)

x....X movement (m)
y....Y movement (m)
z....Z movement (m)

elstart...Start element number (1 to N) where N
is the number of elements in the array

elfinish..Finish element number (1 to N) where N
is the number of elements in the array.
For elfinish>N, elfinish is set equal to N

E.g.   move_array(0.1,0,0,1,4)   % Move 4-element array 0.1m along +ve X-axis

Movec_array

Translate and copy array geometry in x,y,z, new elements are added onto the end of array_config.

Usage: movec_array(x,y,z,elstart,elfinish)

x....X movement (m)
y....Y movement (m)
z....Z movement (m)

elstart...Start element number (1 to N) where N is the number of elements in the array

elfinish..Finish element number (1 to N) where N is the number of elements in the array. For elfinish>N, elfinish is set equal to N

```
E.g. movec_array(0.5,0,0,2,4)    % Copy elements 2 to 4 and move them
                                 % 0.5m along the X-axis
```

Centre_array

Moves entire array such that the average distance of all the elements from the centre of the array is zero.

Usage: centre_array

Clear_array

Re-initialises the array_config matrix, clearing all existing elements. All other variables remain un-changed.

Usage: clear_array

Xrot_array & Xrotc_array

Rotate array geometry about x-axis, rotation is relative to the current orientation. For xrotc_array the specified elements are copied and appended to array_config.

Usage : xrot_array(ang,elstart,elfinish) & xrotc_array(ang,elstart,elfinish)

```
  ang.......Rotation angle (Deg)

  elstart...Start element number (1 to N) where N
            is the number of elements in the array

  elfinish..Finish element number (1 to N) where N
            is the number of elements in the array.
            For elfinish>N, elfinish is set equal to N
```

+ve defined by RH screw rule, holding x-axis. RH = +ve, LH = -ve

```
      z
      |
      |
      |_____ y
     /
    /
   x     RH Screw towards origin
```

Yrot_array & Yrotc_array

Rotate array geometry about y-axis, rotation is relative to the current orientation. For yrotc_array the specified elements are copied and appended to array_config.

Usage : yrot_array(ang,elstart,elfinish) & yrotc_array(ang,elstart,elfinish)

```
  ang.......Rotation angle (Deg)

  elstart...Start element number (1 to N) where N
            is the number of elements in the array

  elfinish..Finish element number (1 to N) where N
            is the number of elements in the array.
            For elfinish>N, elfinish is set equal to N
```

+ve defined by RH screw rule, holding y-axis. RH = +ve, LH = -ve

```
      z
      |
      |
      |_____ y  RH Screw towards origin
     /
    /
   x
```

Zrot_array & Zrotc_array

Rotate array geometry about z-axis, rotation is relative to the current orientation. For zrotc_array the specified elements are copied and appended to array_config.

Usage : zrot_array(ang,elstart,elfinish) & zrotc_array(ang,elstart,elfinish)

ang.......Rotation angle (Deg)

elstart...Start element number (1 to N) where N
is the number of elements in the array

elfinish..Finish element number (1 to N) where N
is the number of elements in the array.
For elfinish>N, elfinish is set equal to N

+ve defined by RH screw rule, holding z-axis. RH = +ve, LH = -ve

```
     z RH Screw towards origin
     |
     |
     |_____ y
    /
   /
  x
```

Cpol_array

Circularly polarise existing array

Usage: cpol_array(Zrot,dPha,dAmp)

Zrot…...Rotation about element local Z-axis (Deg)
dPha.....Relative phase change for each element (Deg)
dAmp....Relative amplitude change for each element (dB)

This function duplicates all existing array elements and then applies relative changes to orientation, phase and amplitude as listed above.

Example Rectangular right-hand circularly polarised array

```
rect_array(8,1,0.8,0,'dipoleg',0);   % Define (8 by 1) array
taywin_array(20,'x');                 % Amplitude taper, if required
squint_array(15,0,1);                % Steer the array, if required
cpol_array(-90,90,0);                % Generate the orthogonal elements for RHCP
```

For LHCP use cpol_array(90,90,0)

4.2 PLOTTING AND VISUALISATION

The plotting and visualisation commands are used to analyse the array defined in the array_config matrix.

The high-level commands listed in this section generate a range of plots in pre-allocated figure windows. The exact nature of the plots depends on user supplied parameters such as angular range, normalisation and number of pattern cuts.

ArrayCalc Function	Description	Fignum
Plot_geom3D	(3D Geometry)	Fig 1
Plot_theta	(cartesian plots as a function of theta) (polar plots as a function of theta)	Fig 2 Fig 3
Plot_phi	(cartesian plots as a function of phi) (polar plots as a function of phi)	Fig 4 Fig 5
Plot_squint_theta	(cartesian theta plots for theta squints) (polar theta plots for theta squints)	Fig 6 Fig 7
Plot_squint_phi	(cartesian phi plots for phi squints) (polar phi plots for phi squints)	Fig 8 Fig 9
Plot_pattern3D	(3D pattern plots)	Fig 10
Plot_geom2D	(2D Geometry)	Fig 11
Plot_theta_statvar	(Effect of statistical variation on array, theta plots)	Fig 12
Plot_phi_statvar	(Effect of statistical variation on array, phi plots)	Fig 13
Plot_geopat3D	(3D pattern and 3D geometry on the same plot)	Fig 14

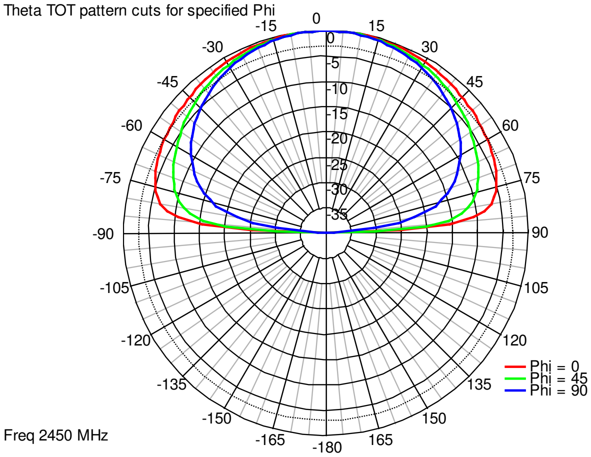
All plot types have the following polarisation options :

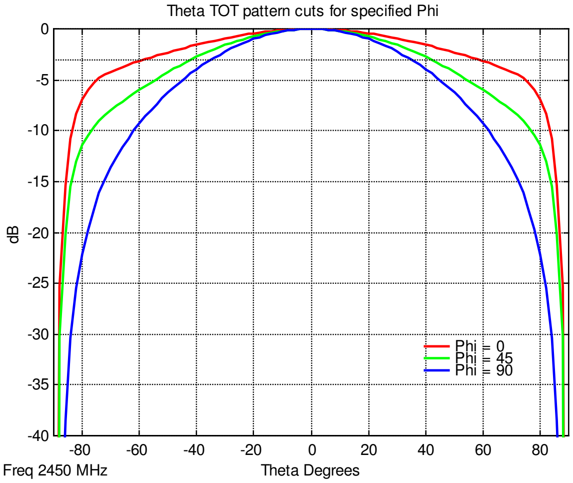
'tot' - Total E-field
'vp' - Vertical polarisation
'hp' - Horizontal polarisation
'lhcp' - Left Hand circular polarisation
'rhcp' - Right Hand circular polarisation

Axial ratio ‘ar’ is only availiable for the 3D pattern plotting functions at present.

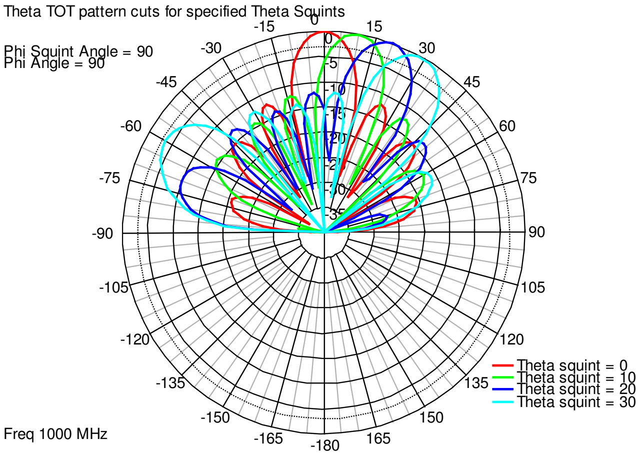
To give the user an idea what output is produced by each of the plotting and visualisation functions, there follows a list of the functions and typical output. This should enable the user to select the most appropriate function for his/her application.

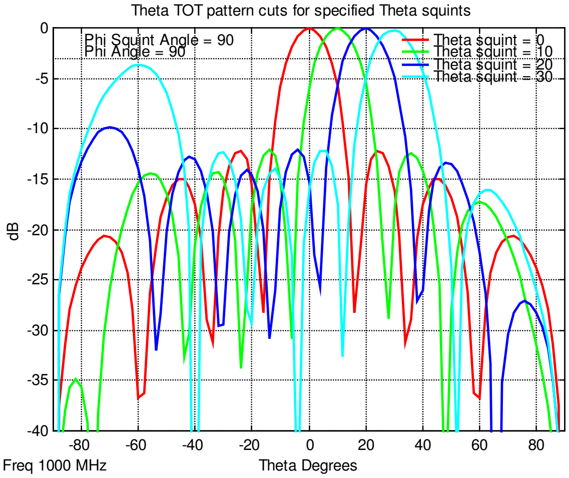
Plot_theta / **phi** (polar and cartesian plots generated)



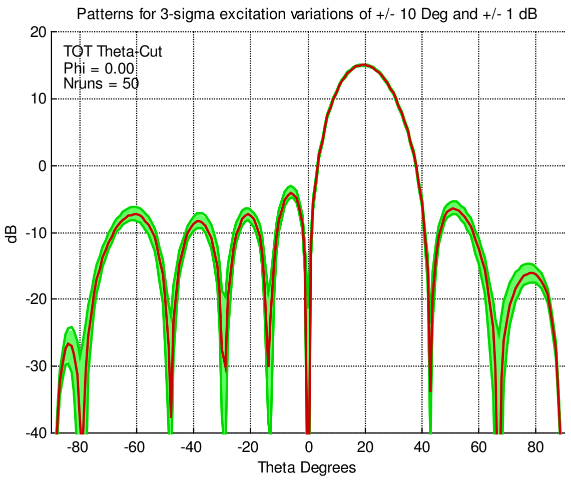


Plot_squint_theta / **phi** (polar and cartesian plots generated)

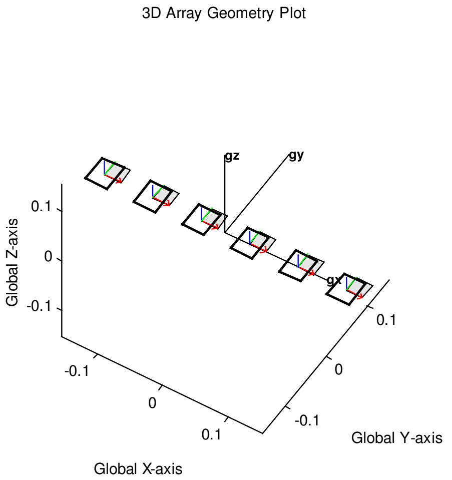




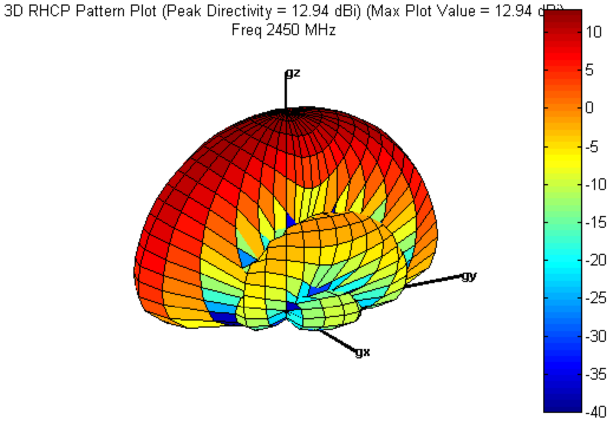
Plot_theta / phi_statvar



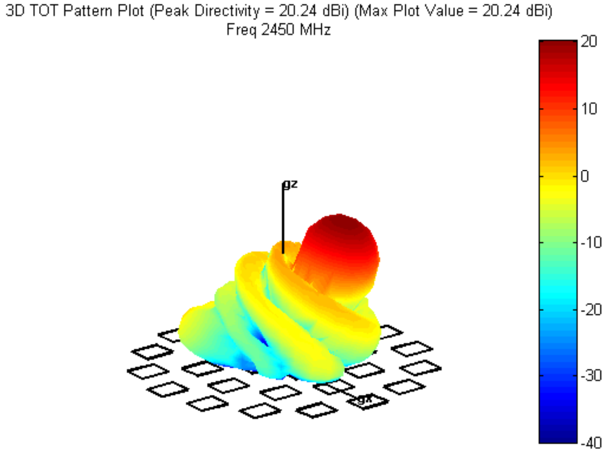
Plot_geom3D



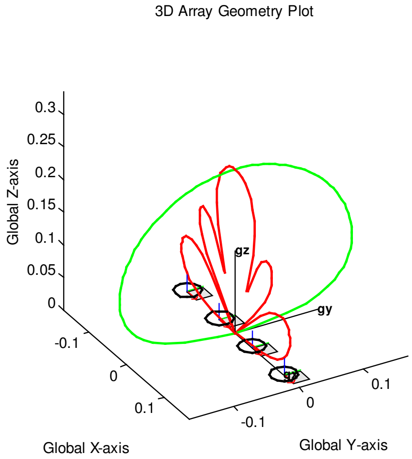
Plot_pattern3D


3D RHCP Pattern Plot (Peak Directivity = 12.94 dBi) (Max Plot Value = 12.94 dBi)
Freq 2450 MHz
gz
gy
gx
10
5
0
-5
-10
-15
-20
-25
-30
-35
-40

Plot_geopat3D


3D TOT Pattern Plot (Peak Directivity = 20.24 dBi) (Max Plot Value = 20.24 dBi)
Freq 2450 MHz
gz
gx
20
10
0
-10
-20
-30
-40

Plot_theta / phi_geo



Note

In V2.4 the functions Plot_theta, Plot_phi Plot_squint_theta, Plot_squint_phi also add pattern cuts to (figure1), the default 3D geometry figure, to aid identification.

For a separate 3D geometry plot use the plot_geom3D1 function.

Plotting and Visualisation Command List

Plot_theta

Plots pattern cuts in theta for specified values of phi.

Default figure(2) for cartesian display
Default figure(3) for polar display

```
Usage: [theta,pwrdB]=plot_theta(thetamin,thetastep,thetamax,phi_list,...
                                 polarisation,normalise)

  thetamin......Minimum value of theta (Deg)
  thetastep.....Step value for theta (Deg)
  thetamax.....Maximum value for theta (Deg)
  phi_list.……....List of Phi values for theta cut (Deg)
  polarisation..Polarisation (string)
  normalise.....Normalisation (string)
```

Options for polarisation are :
'tot' - Total E-field
'vp' - Vertical polarisation
'hp' - Horizontal polarisation
'lhcp' - Left Hand circular polarisation
'rhcp' - Right Hand circular polarisation

Options for normalise are :

'first' - Normalise all cuts to first pattern's maximum value
'each' - Normalise each pattern cut to its maximum value
'none' - Directivity (dBi), no normalisation
Note : calc_directivity must be run first !

E.g. For two -90 to +90 Deg theta cuts for phi values of 0 and 90 Deg normalised to maximum in phi=0 Deg cut use :

```
[theta,pwrdB]=plot_theta(-90,1,90,[0,90],'tot','first')
```

The returned values [theta,pwrdB] correspond to the last cut requested.

Plot_squint_theta

Plots pattern cuts in theta for specified theta-squint angles

Default figure(6) for cartesian display
Default figure(7) for polar display

```
Usage: plot_squint_theta(thetamin,thetastep,thetamax,phi,phi_squint,...
                              theta_squint_list,polarisation,normalise)
```

thetamin..............Minimum value of theta (Deg)
thetastep.............Step value for theta (Deg)
thetamax.............Maximum value for theta (Deg)

phi.......................Phi angle for theta cut (Deg)
phi_squint..............Squint value in phi direction (Deg)
theta_squint_list….List of squint values in theta direction (Deg)

polarisation...........Polarisation (string)
normalise..............Normalisation (string)

Options for polarisation are :

'tot' - Total E-field
'vp' - Vertical polarisation
'hp' - Horizontal polarisation
'lhcp' - Left Hand circular polarisation
'rhcp' - Right Hand circular polarisation

Options for normalise are :

'first' - Normalise all cuts to first pattern's maximum value
'each' - Normalise each pattern cut to its maximum value
'none' - Directivity (dBi), no normalisation
Note : calc_directivity must be run first !

E.g. For -90 to +90 Deg theta cuts for theta squints values of 0, 5 and 10 Deg normalised to maximum in theta_squint=0 Deg use :

```
plot_squint_theta(-90,1,90,0,0,[0,5,10],'tot','first')
```

```
   z
   |-theta   (theta 0-180 measured from z-axis)
   |/
   |_____ y
  /\
 /-phi      (phi 0-360 measured from x-axis)
x
```

Plot_theta_statvar

Plots pattern cuts in theta for specified value of phi and specified
3sigma variations in phase and amplitude of the element excitations.

Default figure(12) for cartesian display

Usage: [thetacut,minpat,maxpat]=plot_theta_statvar(thetamin,thetastep,thetamax,phi,...
polarisation,normalise,phavar,ampvar,Nruns)

thetamin…...Minimum value of theta (Deg)
thetastep..….Step value for theta (Deg)
thetamax......Maximum value for theta (Deg)
phi….............Phi angle for theta cut (Deg)
polarisation..Polarisation (string)
normalise.....Normalisation (string)
phavar....….Phase variation +/- (Deg)
ampvar........Amplitude variation +/- (dB)
Nruns....….Number of runs (integer)

Notes :

The phase and amplitude variations 'phavar' and 'ampvar' are applied to the array as normal distributions about a mean of zero. The variance is scaled such that the value supplied defines the +/- 3sigma of the distribution.

In other words 99.7 pcnt of the random phase values lie in the range of +/- phavar. and 99.7 pcnt of the random amplitude values lie in the range of +/- ampvar.

For each of Nruns the array's phase and amplitude excitations are randomised according to 'phavar' and 'ampvar'. The plots build to form an envelope of maximum and minimum pattern values. These are returned in minpat and maxpat, see below

Also output to the text screen is a summary of the variation statistics for each element. Nruns needs to be sufficiently large so the the 3sigma values are close the supplied values of 'phavar' and 'ampvar'

Options for polarisation are :

'tot' - Total E-field
'vp' - Vertical polarisation
'hp' - Horizontal polarisation
'lhcp' - Left Hand circular polarisation
'rhcp' - Right Hand circular polarisation

Options for normalise are :

'yes' - Normalise each pattern cut to its maximum value
'no' - Directivity (dBi), no normalisation
Note : calc_directivity must be run first !

The returned values [thetacut,minpat,maxpat] are :

thetacut - angle vector (Deg)
minpat - minimum pattern values for all variations applied
maxpat - maximum pattern values for all variations applied

Each vector is of form [1,npoints]

E.g. For a -90 to +90 Deg theta cut for a phi value of 0 Deg,
normalised to maximum in phi=0 Deg
3sigma phase variation +/- 10 Deg
3sigma amplitude variation +/- 1 dB
Run 25 times

use : [thetacut,minpat,maxpat]=plot_theta_statvar(-90,1,90,0,'tot','yes',10,1,25);

```
    z
     |-theta   (theta 0-180 measured from z-axis)
     |/
     |_____ y
    /\
   /-phi       (phi 0-360 measured from x-axis)
  x
```

Plot_phi

Plots pattern cuts in phi for specified values of theta

Default figure(4) for cartesian display
Default figure(5) for polar display

```
Usage: [phi,pwrdB]=plot_phi(phimin,phistep,phimax,theta_list,...
                         polarisation,normalise)

  phimin..........Minimum value of phi (Deg)
  phistep.........Step value for phi (Deg)
  phimax..........Maximum value for phi (Deg)
  theta_list......List of Theta values for phi cut (Deg)
  polarisation...Polarisation (string)
  normalise.......Normalisation (string)
```

Options for polarisation are :

```
'tot'     - Total E-field
'vp'      - Vertical polarisation
'hp'      - Horizontal polarisation
'lhcp'  - Left Hand circular polarisation
'rhcp'  - Right Hand circular polarisation
```

Options for normalise are :

```
'first'    - Normalise all cuts to first pattern's maximum value
'each'    - Normalise each pattern cut to its maximum value
'none'    - Directivity (dBi), no normalisation
              Note : calc_directivity must be run first !
```

```
 E.g. For two 360Deg phi cuts for theta values of 90 and 45 Deg
      normalised to maximum in theta=90 Deg cut use :
      [phi,pwrdB]=plot_phi(0,1,360,[90,45],'tot','first')

      The returned values [phi,pwrdB] correspond to the last plot requested
        z
        |-theta   (theta 0-180 measured from z-axis)
        |/
        |_____ y
       /\
      /-phi       (phi 0-360 measured from x-axis)
     x
```

Plot_squint_phi

Plots pattern cuts in phi for specified phi-squint angles

Default figure(8) for cartesian display
Default figure(9) for polar display

```
Usage: plot_squint_phi(phimin,phistep,phimax,...
                        theta,theta_squint,phi_squint_list,polarisation,normalise)

  phimin...............Minimum value of phi (Deg)
  phistep..............Step value for phi (Deg)
  phimax...............Maximum value for phi (Deg)

  theta................Theta angle for phi cut (Deg)
  theta_squint.......Squint value in theta direction (Deg)
  phi_squint_list....List of squint values in phi direction (Deg)

  polaristion..........Polarisation (string)
  normalise............Normalisation (string)
```

Options for polarisation are :

```
'tot'    - Total E-field
'vp'     - Vertical polarisation
'hp'     - Horizontal polarisation
'lhcp' - Left Hand circular polarisation
'rhcp' - Right Hand circular polarisation
```

Options for normalise are :

```
'first'   - Normalise all cuts to first pattern's maximum value
'each'  - Normalise each pattern cut to its maximum value
'none'  - Directivity (dBi), no normalisation
             Note : calc_directivity must be run first !
```

E.g. For -90 to +90 Deg phi cuts for phi squints values of 0, 5 and 10 Deg
normalised to maximum in phi_squint=0 Deg cut use :
plot_squint_phi(-90,1,90,0,0,[0,5,10],'tot','first')

```
   z
   |-theta   (theta 0-180 measured from z-axis)
   |/
   |_____ y
  /\
 /-phi      (phi 0-360 measured from x-axis)
X
```

Plot_phi_statvar

Plots pattern cuts in phi for specified value of theta and specified
3sigma variations in phase and amplitude of the element excitations.

Default figure(13) for cartesian display

```
Usage: [phicut,minpat,maxpat]=plot_phi_statvar(phimin,phistep,phimax,theta,...
                                    polarisation,normalise,phavar,ampvar,Nruns)
```

phimin.........Minimum value of phi (Deg)
phistep........Step value for phi (Deg)
phimax........Maximum value for phi (Deg)
theta...........Theta value for phi cut (Deg)
polarisation..Polarisation (string)
normalise.....Normalisation (string)
phavar.........Phase variation +/- (Deg)
ampvar........Amplitude variation +/- (dB)
Nruns..........Number of runs (integer)

Notes :

The phase and amplitude variations 'phavar' and 'ampvar' are applied to the array as normal distributions about a mean of zero. The variance is scaled such that the value supplied defines the +/- 3sigma of the distribution.

In other words 99.7 pcnt of the random phase values lie in the range of +/- phavar. and 99.7 pcnt of the random amplitude values lie in the range of +/- ampvar.

For each of Nruns the array's phase and amplitude excitations are randomised according to 'phavar' and 'ampvar'. The plots build to form an envelope of maximum and minimum pattern values. These are returned in minpat and maxpat, see below

Also output to the text screen is a summary of the variation statistics for each element. Nruns needs to be sufficiently large so the the 3sigma values are close the supplied values of 'phavar' and 'ampvar'

Options for polarisation are :

'tot'	- Total E-field
'vp'	- Vertical polarisation
'hp'	- Horizontal polarisation
'lhcp'	- Left Hand circular polarisation
'rhcp'	- Right Hand circular polarisation

Options for normalise are :

'yes'	- Normalise each pattern cut to its maximum value
'no'	- Directivity (dBi), no normalisation Note : calc_directivity must be run first !

The returned values [phicut,minpat,maxpat] are :

phicut - angle vector (Deg)
minpat - minimum pattern values for all variations applied
maxpat - maximum pattern values for all variations applied

Each vector is of form [1,npoints]

E.g. For a -90 to +90 Deg phi cut for a theta value of 90 Deg,
normalised to maximum in phi=0 Deg
3sigma phase variation +/- 10 Deg
3sigma amplitude variation +/- 1 dB
Run 25 times

use : [phicut,minpat,maxpat]=plot_phi_statvar(-90,1,90,90,'tot','yes',10,1,25);

```
   z
   |-theta   (theta 0-180 measured from z-axis)
   |/
   |_____ y
  /\
 /-phi      (phi 0-360 measured from x-axis)
x
```

Plot_geom3D

Draw local axis sets for elements and show array geometry in 3D.
Default figure(1).

Usage: plot_geom3d(gaxisflag,anotflag)

```
  gaxisflag..Flag=1 to plot global axis set, labeled gx,gy and gz
             Flag=0 no axis set

  anotflag...Flag=1 specifies that element details are added to plot
             Flag=0 no details
```

Details if requested are :

```
  Element number      - As stored in array_config
  Element amplitude   - dB power (20*log10(Volts))
  Element phase       - Degrees

 E.g.   plot_geom3d(0,0)     % No axis, Geometry only
        plot_geom3d(1,1)     % Global axis and element details
```

Plot_geom3D1

As for plot_geom3D except figure number can be specified.

Plot_geom2D

Draw local axis sets for elements and show array geometry in 2D.
Default figure(11)

Usage: plot_geom2d(gaxisflag,anotflag)

```
  gaxisflag..Flag=1 to plot global axis set, labeled gx and gy
             Flag=0 no axis set

  anotflag...Flag=1 specifies that element details are added to plot
             Flag=0 no details
```

Details if requested are :

```
  Element number      - As stored in array_config
  Element amplitude  - dB power (20*log10(Volts))
  Element phase       - Degrees

E.g.   plot_geom2d(0,0)    % No axis, Geometry only
       plot_geom2d(1,1)    % Global axis and element details
```

Plot_pattern3d

Plot 3D pattern
Default figure (10)

Usage: plot_pattern3D(deltheta,delphi,polarisation,normalise)

deltheta……..Step value of theta (Deg) Such that 180/deltheta is an integer
delphi...……..Step value for phi (Deg)   Such that 360/delphi is an integer
polarisation....Polarisation (string)
normalise.…..Normalisation (string)

Options for polarisation are :

'tot'   - Total E-field
'vp'   - Vertical polarisation
'hp'   - Horizontal polarisation
'lhcp' - Left Hand circular polarisation
'rhcp' - Right Hand circular polarisation
'ar'   - Axial Ratio

Options for normalise are :

'yes' - Normalise pattern suface to its maximum value
'no'   - Directivity (dBi), no normalisation
Note : calc_directivity must be run first !

E.g.   plot_pattern3D(10,15,'tot','no')

```
     z
     |-theta   (theta 0-180 measured from z-axis)
     |/
     |_____ y
    /\
   /-phi       (phi 0-360 measured from x-axis)
  x
```

Plot_pattern3d1

Plot 3D pattern in selected figure, suggest using figure 15 or greater, numbers 1 to 14 already assigned as defaults for other plots.

Usage: plot_pattern3D1(deltheta,delphi,polarisation,normalise,fignum)

See plot_pattern3d for parameter definitions.

Plot_geopat3D

Plot 3D pattern and array geometry.
Default figure (14)

Usage: plot_geopat3D(deltheta,delphi,polarisation,normalise,surftype,gpratio)

deltheta…......Step value of theta (Deg) Such that 180/deltheta is an integer
delphi...…....Step value for phi (Deg) Such that 360/delphi is an integer
polarisation…Polarisation (string)
normalise…...Normalisation (string)
surftype...….Surface plot type (string)
gpratio...…...Geometry to pattern ratio (float)

Options for polarisation are :
'tot' - Total E-field
'vp' - Vertical polarisation
'hp' - Horizontal polarisation
'lhcp' - Left Hand circular polarisation
'rhcp' - Right Hand circular polarisation
'ar' - Axial Ratio

Options for normalise are :
'yes' - Normalise pattern suface to its maximum value
'no' - Directivity (dBi), no normalisation
Note : calc_directivity must be run first !

Options for surftype are :
'surf' - Surface plot, interpolated shading
'mesh' - Wire mesh plot

Notes for gpratio are :

The larger the value for gpratio the larger the geometry
plot relative to the pattern plot will be.
Values are typically (1 to 5)
For a small array (2 x 2) choose a large value (4)
For a large array (8 x 8) choose a small value (2)

E.g. plot_geopat3D(10,15,'tot','no','surf',4)

Plot_geopat3D1

Plot 3D pattern and array geometry in selected figure, suggest using 15 or greater, numbers 1 to 14 already assigned as defaults for other plots.

Usage: plot_geopat3D1(deltheta,delphi,polarisation,normalise,surftype,gpratio,fignum)

E.g. plot_geopat3D1(10,15,'tot','no',’surf’,4,15)
See plot_geopat3D for parameter definitions.

Plot_theta_geo1

Plots a single pattern cut in theta for a specified value of phi.

Parameters are as for plot_theta except plots are added to the 3D geometry plot and linestyle and figure number can be specified.

```
Usage: [theta,pwrdB]=plot_theta_geo1(thetamin,thetastep,thetamax,phi_list,...
                          polarisation,normalise,linestyle,fignum)
```

E.g. For a -90 to +90 Deg theta cut for a phi value of 0 Deg
normalised to maximum, in red, on figure1, use :

```
[theta,pwrdB]=plot_theta_geo1(-90,1,90,[0],'tot','first',’r-‘,1)
```

Plot_phi_geo1

Plots a single pattern cut in phi for a specified value of theta.
Parameters are as for plot_phi except plots are added to the 3D geometry plot and linestyle and figure number can be specified.

```
Usage: [theta,pwrdB]=plot_phi_geo(phimin,phistep,phimax,theta_list,...
                          polarisation,normalise,linestyle,fignum)
```

E.g. For two 0-360 Deg phi cuts for theta values of 90 and 45 Deg
normalised to maximum in theta=90 Deg cut, use :

```
[phi,pwrdB]=plot_phi_geo(0,1,360,[90,45],'tot','first',’r-‘,1)
```

List_array

Lists array element details as stored in array_config.
Positive phase angle indicates delay i.e. phase-lag.

```
Usage: list_array(optionflag)
```

optionflag...Flag=1 lists all element data including rotation matricies
Flag=0 lists element locations and excitations only

```
E.g.   list_array(0)   % List element excitations only
```

Plegend

Put a single legend line on a plot at specified screen co-ordinates

Usage: plegend(xsc,ysc,linetype,label)

xsc.…......X screen coordinate
ysc..…....Y screen coordinate
linetype...Standard MATLAB definition (string)
label..…..Text label (string)

Screen coords : (0,1)   (1,1)

(0,0)   (1,0)

Vertical spacing for successive lines is typically 0.03

```
E.g.   plegend(0.8,0.2,'b-','Phi=0 Cut')      % (0.85,0.25) is a good position for
        plegend(0.8,0.2,'r-','Phi=90 Cut')     % polar plot legend.
```

4.3 SUBROUTINES

The subroutines directory contains ArrayCalc’s core calculation routines and lower level commands. The fieldsum and calc_directivity m-files form the heart of ArrayCalc, most of the other functions are either involved with defining the array to be analysed or post-processing the resulting data. Apart from using the calc_directivity function, it is recommended that the user gains a good understanding of the higher level commands (sections 3 and 4), before using the other m-files in this directory.

The command list below doesn’t document all the commands in the subroutines directory, but does cover those that are most likely to be used in user scripts. See the individual help files for those concerning co-ordinate transforms and rotation matrices.

Subroutines Command List

Calc_directivity

Calculate peak directivity in dBi value using numerical integration.

The result is is stored in the global variable direct_config.
The maximum pattern value is also stored in the global variable normd_config.

Usage: directivity=calc_directivity(deltheta,delphi)

```
  deltheta.....Step value of theta (Deg) Such that 180/deltheta is an integer
  delphi……..Step value for phi (Deg)  Such that 360/delphi is an integer
```

E.g. directivity=calc_directivity(10,10) :

Integration is of the form :

```
     360   180
    Int{  Int{  (E(theta,phi)*conj(E(theta,phi))*sin(theta) d(theta) d(phi)
      0     0
```

```
       z
       |-theta   (theta 0-180 measured from z-axis)
       |/
       |_____ y
      /\
     /-phi       (phi 0-360 measured from x-axis)
    x
```

Fieldsum

Summation of field contributions at location (R,th,phi)
from each element in array_config, at frequency freq_config.

Usage: Emulti=fieldsum(R,th,phi)

R....Radius of farfield point
th...Theta (radians)
phi..Phi (radians)

Returned values :

Emulti..[Etot,Evp,Ehp,Elhcp,Erhcp,AR,Tau] where :

Etot	=	Total E-field
Evp	=	Vertical E-field component (Z-axis in global coords)
Ehp	=	Horizontal E-field component (X-Y plane in global coords)
Elhcp	=	Left Hand Circular Polarisation
Erhcp	=	Right Hand Circular Polarisation
AR	=	Axial ratio (Linear)
Tau	=	Tilt angle of polarisation ellipse (radians)

Components of Emulti (Etot,Evp...etc) are of the form variable[npoints,1]

This version of fieldsum includes 1/r path loss factor for psuedo-near field calculations. i.e. It accounts for the relative distances (relative path losses) to the various array elements, as viewed by the observer. It should be noted that distances where this is significant are likely to be close to the Fraunhofer distance of 2D^2/lambda (D is the maximum aperture dimension).

ArrayCalc is intended for use as a far-field calculator, therefore the distance to the summing point (range_config) is ideally set to a value >>(2D^2/lambda).

The default value for range_config is 999(m) set in init.m

Polaxis

Draw set of polar axis.

Usage : polaxis(rmin,rmax,rstep,astep);

rmin - Polar axis centre threshold
rmax - Maximum polar radius value
rstep - Radius increment
astep - Radial increment (degrees)

Example : polaxis(-40,0,5,15)

Any value less than or equal to -40 should be plotted at the centre.
Concentric circles for -40 -30 -20 -10 0 10 and 20.
Radial lines at 36 degree increments.

The file circ.m is required to run this module.

Polplot

Plots pattern data in polar form onto polar axis set.

Usage: polplot(theta,pwrdB,mindB,linestyle,linewidth1,linewidth2);

theta.…….Theta angle on polar chart in (degrees)
pwrdBn......Normalised pattern data (dB)
mindB........Minimum dB value to be displayed (dB)

linestyle…..Standard Matlab definition (string)
linewidth1..Standard linewidth control string (string)
linewidth2..Standard linewidth (integer)

E.g. polplot(theta,pwrdB1,-40,'r-','linewidth',2);

Theta_cut

Cut in theta for single value of phi

Usage: [thetacut,Emulti]=theta_cut(thmin,thstep,thmax,phi)

thmin......Minimum theta value (Deg)
thstep......Step size (Deg)
thmax......Maximum theta value (Deg)
phi..........Phi value for pattern cut (Deg)

Returned values :

thetacut...theta values (Deg)

Emulti.....[Etot,Evp,Ehp,Elhcp,Erhcp,AR,Tau] where :

```
Etot   =  Total E-field
Evp    =  Vertical E-field component (Z-axis in global coords)
Ehp    =  Horizontal E-field component (X-Y plane in global coords)
Elhcp  =  Left Hand Circular Polarisation
Erhcp  =  Right Hand Circular Polarisation
AR     =  Axial ratio (Linear)
Tau    =  Tilt angle of polarisation ellipse (radians)
```

Theta and components of Emulti (Etot,Evp...etc) are of the form variable[npoints,1]

```
   z
    |-theta   (theta 0-180 measured from z-axis)
    |/
    |_____ y
   /\
  /-phi       (phi 0-360 measured from x-axis)
 x
```

Phi_cut

Cut in phi for single value of theta

Usage: [phicut,Emulti]=phi_cut(phimin,phistep,phimax,theta)

```
  phimin......Minimum phi value (Deg)
  phistep.....Step size (Deg)
  phimax......Maximum phi value (Deg)
  theta.......Theta value for pattern cut (Deg)
```

Returned values :

phicut..phi values (Deg)

Emulti..[Etot,Evp,Ehp,Elhcp,Erhcp,AR,Tau] where :

```
Etot  =  Total E-field
Evp   =  Vertical E-field component (Z-axis in global coords)
Ehp   =  Horizontal E-field component (X-Y plane in global coords)
Elhcp =  Left Hand Circular Polarisation
Erhcp =  Right Hand Circular Polarisation
AR    =  Axial ratio (Linear)
Tau   =  Tilt angle of polarisation ellipse (radians)
```

Phi and components of Emulti (Etot,Evp...etc) are of the
form variable[npoints,1]

```
     z
     |-theta   (theta 0-180 measured from z-axis)
     |/
     |_____ y
    /\
   /-phi       (phi 0-360 measured from x-axis)
  x
```

Calc_theta

Calculates single pattern cut data in theta for specified value of phi
no plots are generated.

```
Usage: [thetacut,pwrdBn]=calc_theta(thetamin,thetastep,thetamax,phi_val,...
                                    polarisation,normalise)
```

thetamin......Minimum value of theta (Deg)
thetastep.....Step value for theta (Deg)
thetamax......Maximum value for theta (Deg)
phi_val.......Phi value for theta cut (Deg)
polarisation..Polarisation (string)
normalise.....Normalisation (string)

Options for polarisation are :

'tot' - Total E-field
'vp' - Vertical polarisation
'hp' - Horizontal polarisation
'lhcp' - Left Hand circular polarisation
'rhcp' - Right Hand circular polarisation

Options for normalise are :

'yes' - Normalise pattern cut to its maximum value
'no' - Directivity (dBi), no normalisation
Note : calc_directivity must be run first !

Returned values :

thetacut is of the form thetacut[1,npoints] Where npoints is the size of vector thetamin:thetastep:thetamax

pwrdBn is of the form pwrdBn[1,npoints]

E.g. For a theta (-90:5:+90) cut for a phi value of 45Deg normalised to maximum use :
[thetacut,pwrdBn]=calc_theta(-90,5,90,45,'tot','yes')

```
    z
    |-theta   (theta 0-180 measured from z-axis)
    |/
    |_____ y
   /\
  /-phi       (phi 0-360 measured from x-axis)
 x
```

Calc_phi

Calculates single pattern cut data in phi for specified value of theta no plots are generated.

```
Usage: [phicut,pwrdBn]=calc_phi(phimin,phistep,phimax,theta_val,...
                                polarisation,normalise)
```

phimin........Minimum value of phi (Deg)
phistep.......Step value for phi (Deg)
phimax........Maximum value for phi (Deg)
theta_val.....Theta value for phi cut (Deg)
polarisation..Polarisation (string)
normalise.....Normalisation (string)

Options for polarisation are :

'tot' - Total E-field
'vp' - Vertical polarisation
'hp' - Horizontal polarisation
'lhcp' - Left Hand circular polarisation
'rhcp' - Right Hand circular polarisation

Options for normalise are :

'yes' - Normalise each pattern cut to its maximum value
'no' - Directivity (dBi), no normalisation
Note : calc_directivity must be run first !

Returned values :

phicut is of the form phicut[1,npoints] Where npoints is the size of vector phimin:phistep:phimax

pwrdBn is of the form pwrdBn[1,npoints]

E.g. For a (0:5:360) phi cut for a theta value of 45Deg normalised to maximum use :
[phicut,pwrdBn]=calc_phi(0,5,360,45,'tot','yes')

```
    z
    |-theta   (theta 0-180 measured from z-axis)
    |/
    |_____ y
   /\
  /-phi      (phi 0-360 measured from x-axis)
 x
```

Textsc

Places text at screen coords on current figure.
Lower Left (0,0)
Upper Right (1,1);

Usage : textsc(x,y,'text');

```
E.g.   plot(x,y);
       textsc(0.8,0.5,'text');
```

Plotsc

Draws line in colour c from (x1,y1) to (x2,y2)
Lower Left (0,0)
Upper Right (1,1);

Usage : plotsc(x1,y1,x2,y2,'colour');

```
E.g.     plotsc(0.3,0.4,0.6,0.8,'g')
```

Design_helix

Returns the helix_config parameters for endfire helix.
Helix turn circumference and turn spacing are calculated
and returned together with supplied parameter N.

Returned values are in the same format as the global helix_config variable, so can be assigned directly. The helix_config variable is of the following form [N,S].

Usage: helix_config=design_helix(N,Freq)

N.......Number of turns (integer)
Freq...Frequency (Hz)

Design is for a helix that is approximated well by the model i.e.
Number of turns, N > 3
Ratio of S/C = 0.2217 (helix pitch angle of 12.5 Deg)
C is set = (1.2)*Lambda

Note : The printed values are in easier to read units of MHz and cm

```
E.g.   helix_config=design_helix(6,1e9)
```

Design_patchr

Returns the patch_config parameters for standard lambda/2 rectangular microstrip patch. Patch length L and width W are calculated and returned together with supplied parameters Er and h.

Returned values are in the same format as the global patchr_config variable, so can be assigned directly. The patchr_config variable is of the following form [Er,W,L,h].

Usage: patchr_config=design_patchr(Er,h,Freq)

Er…...Relative dielectric constant
h….....Substrate thickness (m)
Freq...Frequency (Hz)

E.g. patchr_config=design_patchr(3.43,0.7e-3,2e9)

Design_patchc

Returns the patch_config parameters for standard lambda/2 circular microstrip patch. Patch radius a is calculated and returned together with supplied parameters Er and h.

Returned values are in the same format as the global patchc_config variable, so can be assigned directly. The patchc_config variable is of the following form [Er,a,h].

Usage: patchc_config=design_patchc(Er,h,Freq)

Er…...Relative dielectric constant
h…....Substrate thickness (m)
Freq...Frequency (Hz)

E.g. patchc_config=design_patchc(2.2,1.588e-3,10e9)

5 BEAM SYNTHESIS

In order to control the beam shape and sidelobe levels produced by an array there are a number of different amplitude distributions that can be applied. The distributions allow the designer to specify the 1st sidelobe level with respect to the main beam or in the case of the Binomial distribution, eliminate sidelobes altogether.

For more specific control of the array pattern, the Fourier relationship between the array excitation and far-field pattern can be used. The desired pattern profile can be specified and the array excitations required to produce it can be calculated using a discrete Fourier transform.

The techniques for ‘designing’ array patterns are generally referred to as beam or pattern synthesis.

Modtaylor

Calculation of modified Taylor distribution for monotonically decreasing
sidelobe levels. Ref JASIK 20-9

Usage : [Lin_Volts,dB_POWER]=modtaylor(N,R)

N....Number of elements in array
R....Sidelobe ratio of 1st sidelobe w.r.t. main beam (dB)

Valid range for R is : 15 < R < 60

Chebwin1

Calculation of Chebyshev distribution for uniform
sidelobe levels.

Usage : [Lin_Volts,dB_POWER]=chebwin1(M,R)

M....Number of elements in array
R....Sidelobe ratio of 1st sidelobe w.r.t. main beam (dB)

Valid range for R is : 15 < R < 60

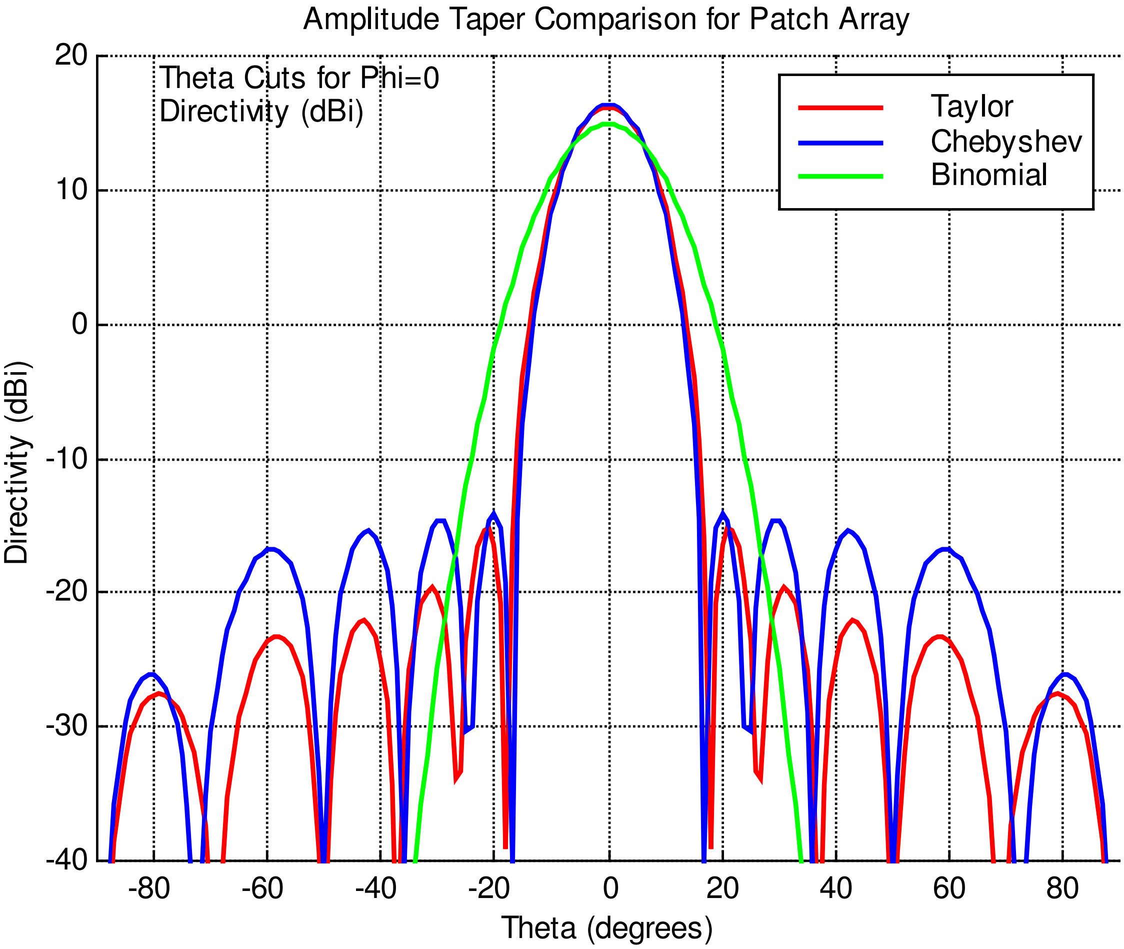
Reference RICK LYONS article on DSPrelated.com
Gives the same results as the MATLAB signal
processing function chebwin.m

Binomial1

Calculation of Binomial distribution for zero power sidelobes.

Usage : [Lin_Volts,dB_POWER]=binomial1(N)

N....Number of elements in array



Graph of the output from the Exdist example file.

Fourier1
Calculation of Fourier coefficients for specified pattern profile

```
Usage : [Lin_Volts,Phase_Rad,Theta,FnValdB]=fourier1(N,Dx,Profile)

  N.........Number of elements in array (odd or even)
  Dx........Array element spacing (m), ideally Dx=0.5*lambda
  Profile...Ideal pattern profile (Min angle=-90, Max angle=+90)


Returned Values :

  Lin_Volts...Amplitude of coefficients in (linear volts)
  Phase_Rad...Phase of coefficients (radians)
  Theta.......Theta vector for plotting [-90:1:90] (Deg)
  FnValdB.....The pattern profile defined over theta vector (dB)


To define a pattern that is -50dB between -90 and -45 deg,
                                0dB between -45 and +45 deg,
                              -50dB between +45 and +90 deg

Use: Angle=[-90 -45 -45 +45 +45  90];  % Angle data points (Deg)
     PwrdB=[-50 -50  0   0  -50 -50];    % Power data points (dB)

     Profile=[Angle;PwrdB];            % Assemble the profile matrix
     Dx=lambda*0.5;                    % Define array spacing (m)
     N=16;                             % Number of array elements

     [Lin_Volts,Phase_Rad,Theta,FnValdB]=fourier1(N,Dx,Profile);
     array_config(1,5,:)=Lin_Volts;
     array_config(2,5,:)=Phase_Rad;
```

Notes : The pattern profile is interpreted as a series of linked lines drawn between the Angle,PwrdB coordinates supplied.

This type of synthesis is generally used for flat-top sector coverage patterns. Although more complicated pattern profiles are possible, the resulting element excitations can result in very inefficient use of the array aperture. Basically many of the elements end up at very low power levels. It also requires fairly precise control of both amplitude and phase.

For the method to work, array elements must be regularly spaced and ideally 0.5*lambda apart. Less then 0.5*lambda causes the transform to map to non-visible space (complex values for theta). Larger than 0.5*lambda and grating lobes start to appear.

See C.A. Balanis 2nd Edition page 349 for more details

6 TUTORIAL EXAMPLES

In this section the user is ‘walked through’ some simple application examples, using the ArrayCalc toolbox. Type ‘exlist’ at the command prompt to see a full list of examples.

6.1 DIPOLE PATTERNS

Files : Tutorial1a.m and Tutorial1b.m

For this example we are going to define a vertically orientated half-wave dipole, operating at 1Ghz.

We will then calculate / plot the following parameters :

- Directivity (dBi)
- Azimuth and Elevation patterns

Having calculated the parameters for a single dipole, we will add a second dipole and repeat the calculations.

Single Dipole (File : Tutorial1a.m)

For the single dipole we start with the following script, note the general format of initialisation → Geometry Construction → Plotting and Visualisation.

```
% *************** Initialisation *************

close all;                 % Close all windows
clc;                       % Clear text screen

init;                      % This initialises all the configuration variables
freq_config=1e9;           % Set frequency to 1Ghz
lambda=3e8/freq_config;     % Define a variable for wavelength, this is not obligatory it's
                           % just to make the script more readable

% ********  Geometry Construction ********

dipole_config=[lambda/2]; % Configure the dipole (it requires only a single parameter of
                           % length).

single_element(0,0,0,'dipole',0,0);        % Place a single dipole at the origin, aligned by
                                           % default with the x-axis. The last 2 parameters
                                           % define 0dB amplitude and 0deg phase.

 yrot_array(90,1,1);        % Rotate the dipole 90deg around the y-axis to bring it to the
                            % vertical. The last 2 parameters refer to the start and finish
                            % elements of the array, that are to be rotated.
% The array_config matrix now contains the array description, albeit an array of one
%  element

% ******* Plotting and Visualisation ********

list_array(0);              % List out the array definition at the Matlab prompt
plot_geom3d(1,1);           % Plot array geometry in 3d, including axis, ampl and phase


% Matlab commands to fine tune the 3D view for this example

view(30,30);                % Orientate 3D view to see dipole more easily
AX=axis;                    % Store 3D axis setting in variable AX
Axis(AX/2);                 % Zoom by 2x on 3D plot

% End of fine tuning


calc_directivity(5,15);     % Calulate the directivity for the array using 5deg increments
                            %  in theta and 15deg increments in phi.

plot_theta(-180,5,180,[0],'tot','none');     % Plot the Elevation pattern, a theta pattern from
                                             % -180 to +180 in 5deg steps, for phi=0deg.
                                             %  (i.e. pattern the X-Z plane)

plot_phi(0,5,360,[90],'tot','none');        % Plot the Azimuth pattern, a phi pattern from
                                            % 0 to +360 in 5deg steps, for theta=90deg.
                                            % (i.e. pattern the X-Y plane)

plot_geopat3D(5,15,'tot','no','mesh',4);    % Plot 3D pattern as a mesh with array geometry
                                            % overlaid, to help identify individual pattern cuts.

% *************  End of Script *****************
```

Two Dipoles (File :Tutorial1b.m)

To add the second dipole, we simply add the following lines to the Geometry Construction section of the script file :

```
movec_array(0,lambda/2,0,1,1);          % Copy the 1st dipole and place the copy
                                        % lambda/2 along the x-axis. The first 3
                                        % parameters are the x,y,z offsets; the
                                        % last 2 are the start and end element numbers
                                        % to be copied.

centre_array();                         % Centre the array on the global axes
```

The tutorial examples 1a/b demonstrate a selection high-level plotting commands that are available. The user is encouraged to experiment by using the example files as a template for analysing different array designs. See section-3 of the manual for details of the different radiating elements that are available. See section-4 for the commands that are available to organise the elements into arrays.

6.2 COMPARING RESULTS

Files : Tutorial2a

Defining arrays and plotting patterns using the high-level plotting commands is fairly straightforward, hopefully as indicated in tutorial example 1a/b.

However, another likely use of ArrayCalc is to compare results for different array configurations or array excitations. To do this requires the use of the lower-level functions, in the ‘subroutines’ directory (section 5).

For example, suppose we want to compare the results from our previous two array configurations (1-dipole and 2-dipoles), on the same plot. To do this we need to generate and store the plot-data for each array configuration, then assemble the plots at the end.

One and Two Dipoles (File :Tutorial2a.m)

```
init;                   % This initialises all the configuration variables
freq_config=1e9;        % Set frequency to 1Ghz
lambda=3e8/freq_config; % Define a variable for wavelength, this is not obligatory it’s
                        % just to make the script more readable

% ******  Geometry Construction for 1-dipole configuration ******

fprintf('\n\n  1-Dipole Array Construction and Pattern Calculations\n');
fprintf('  ==============================================\n');



dipole_config=[lambda/2]; % Configure the dipole (it requires only a single parameter of
                          % length).

single_element(0,0,0,'dipole',0,0);        % Place a single dipole at the origin, aligned by
                                           % default with the x-axis. The last 2 parameters
                                           % define 0dB amplitude and 0deg phase.

yrot_array(90,1,1);                        % Rotate the dipole 90 deg around the y-axis to
                                           % bring it to the vertical
% ********** Calculate the 1-dipole pattern information ***********

list_array(0);                   % List out the array definition at the Matlab
                                           % prompt


D1Direc=calc_directivity(5,15);  % Calc and store the directivity for the array using
                                           % 5deg increments
                                           % in theta and 15deg increments in phi.


[D1Theta,D1ThetaPat]=calc_theta(-180,5,180,[0],'tot','no');
                                           % Calc the Elevation pattern and store.
                                           % -180 to +180 in 5deg steps, for phi=0deg.
                                           %  (i.e. pattern the X-Z plane)


[D1Phi,D1PhiPat]=calc_phi(0,5,360,[90],'tot','no');
                                           % Calc the Azimuth pattern and store.
                                           % 0 to +360 in 5deg steps, for theta=90deg.
                                           % (i.e. pattern the X-Y plane)

% *******************  End of 1-Dipole Calculations *******************


clear_array;      % Clear the existing array configuration


% *****  Geometry Construction for 2-dipole configuration *******


fprintf('\n\n  2-Dipole Array Construction and Pattern Calculations\n');
fprintf('  ====================================================\n');

single_element(0,0,0,'dipole',0,0);  % Place a single dipole at the origin, aligned by
                                           % default with the x-axis. The last 2 parameters
                                           % define 0dB amplitude and 0deg phase.

yrot_array(90,1,1);              % Rotate the dipole 90 deg around the y-axis to
                                           % bring it to the vertical

movec_array(0,lambda/2,0,1,1);   % Copy the 1st dipole and place the copy
                                           % lambda/2 along the x-axis


centre_array;                    % Centre the array on the global axes


list_array(0);                   % List out the array definition at the Matlab
                                           % prompt
% ******** Calculate the 2-dipoles pattern information *********


D2Direc=calc_directivity(5,15);            % Calculate and store the directivity for the array
                                           % using 5deg increments in theta and 15deg
                                           % increments in phi.

[D2Theta,D2ThetaPat]=calc_theta(-180,5,180,[0],'tot','no');
                                           % Calc the Elevation pattern and store.
                                           % -180 to +180 in 5deg steps, for phi=0deg.
                                           %  (i.e. pattern the X-Z plane)

[D2Phi,D2PhiPat]=calc_phi(0,5,360,[90],'tot','no');
                                           % Calc the Azimuth pattern and store.
                                           % 0 to +360 in 5deg steps, for theta=90deg.
                                           % (i.e. pattern the X-Y plane)

% ***************  End of 2-Dipoles Calculations *****************

% ******************  Construct the polar plots *******************

figure(3);
clf;
polaxis(-dBrange_config,15,5,15);   % Set up polar axis (min(dB) max(dB) d(dB) d(Ang))

polplot(D1Theta,D1ThetaPat,-dBrange_config,'r','LineWidth',2);     % Plot 1-dipole theta
                                                                   % (EL) pattern

polplot(D2Theta,D2ThetaPat,-dBrange_config,'b','LineWidth',2);     % Plot 2-dipoles theta
                                                                   % (EL) pattern

plegend(0.78,0.14,'r','1-Dipole EL Pat');   % 1-dipole EL pat label at screen coords
plegend(0.78,0.11,'b','2-Dipoles EL Pat'); % 2-dipoles EL pat label at screen coords

textsc(-0.10,1.00,'Theta (EL) plots');        % Title line1 at screen coords
textsc(-0.10,0.97,'Directivity (dBi)');       % Title line2 at screen coords


figure(4);
clf;
polaxis(-dBrange_config,15,5,15);   % Set up polar axis (min(dB) max(dB) d(dB) d(Ang))

polplot(D1Phi,D1PhiPat,-dBrange_config,'r','LineWidth',2);   % Plot 1-dipole phi (AZ) pat
polplot(D2Phi,D2PhiPat,-dBrange_config,'b','LineWidth',2);   % Plot 2-dipoles phi (AZ) pat

plegend(0.78,0.14,'r','1-Dipole AZ Pat');    % 1-dipole EL pat label at screen coords
plegend(0.78,0.11,'b','2-Dipoles AZ Pat');  % 2-dipoles EL pat label at screen coords


textsc(-0.10,1.00,'Phi (AZ) plots');         % Title line1 at screen coords
textsc(-0.10,0.97,'Directivity (dBi)');      % Title line2 at screen coords
```